



SIMREMOTE USER MANUAL AND ICD

SOFTWARE FOR THE SPIRENT RANGE OF SATELLITE NAVIGATION SIMULATOR PRODUCTS

PROPRIETARY INFORMATION

THE INFORMATION CONTAINED IN THIS DOCUMENT IS THE PROPERTY OF SPIRENT COMMUNICATIONS PLC. EXCEPT AS SPECIFICALLY AUTHORISED IN WRITING BY SPIRENT COMMUNICATIONS PLC, THE HOLDER OF THIS DOCUMENT SHALL KEEP ALL INFORMATION CONTAINED HEREIN CONFIDENTIAL AND SHALL PROTECT SAME IN WHOLE OR IN PART FROM DISCLOSURE AND DISSEMINATION TO ALL THIRD PARTIES TO THE SAME DEGREE IT PROTECTS ITS OWN CONFIDENTIAL INFORMATION.

COPYRIGHT © 2002 - 2010 SPIRENT COMMUNICATIONS PLC

Contents

CHAPTER 1: GENERAL	1-1
1.1 This issue	1-1
1.2 Referenced documents	1-1
1.3 Copyright notices	1-2
1.4 Document conventions	1-2
CHAPTER 2: SIMREMOTE OVERVIEW	2-1
2.1 Timer card option	2-1
2.2 Interface options	2-2
CHAPTER 3: PRINCIPLES OF OPERATION	3-1
3.1 Remote motion - Simulation iteration rate	3-1
3.2 Remote motion - hardware set-up	3-1
3.3 Remote control - timing and synchronisation	3-2
CHAPTER 4: INTRODUCTION TO REMOTE CONTROL	4-1
4.1 Selecting a remote source in SimGEN	4-1
4.1.1 <i>Command file</i>	4-2
4.1.2 <i>IEEE-488 interface</i>	4-3
4.1.3 <i>TCP/IP sockets interface</i>	4-4
4.1.4 <i>RS-232 interface</i>	4-4
4.1.5 <i>SCRAMNet interface</i>	4-4
4.2 Remote command example - running a scenario	4-6
CHAPTER 5: REMOTE COMMANDS	5-1
5.1 Introduction	5-1
5.1.1 <i>Timestamp</i>	5-12
5.1.2 <i>Returned response format</i>	5-13
5.2 Scenario commands	5-14
5.2.1 <i>SET_VEHICLE_DIAL_TYPE</i>	5-15
5.2.2 <i>SET_VEHICLE_DIAL_UNITS</i>	5-15
5.2.3 <i>SC</i>	5-16
5.2.4 <i>SC_NAME</i>	5-16
5.2.5 <i>SC_DURATION</i>	5-16
5.2.6 <i>INIT_POS</i>	5-17
5.2.7 <i>TR</i>	5-17
5.2.8 <i>RU</i>	5-18
5.2.9 <i>RU_NOWAIT</i>	5-18
5.2.10 <i>EN</i>	5-19
5.2.11 <i>RW</i>	5-19
5.2.12 <i>AR</i>	5-20
5.2.13 <i>AR_NOWAIT</i>	5-20
5.2.14 <i>NULL</i>	5-20
5.2.15 <i>LOAD_ALMANAC</i>	5-21
5.2.16 <i>WRITE_ALMANAC</i>	5-21
5.2.17 <i>LOAD_RINEX</i>	5-21
5.2.18 <i>WRITE_RINEX</i>	5-22
5.2.19 <i>LOAD_ACT</i>	5-22
5.2.20 <i>LOAD_UCD</i>	5-23
5.2.21 <i>REGISTER_EXTERNAL_ARM</i>	5-23
5.2.22 <i>DEREGISTER_EXTERNAL_ARM</i>	5-23
5.2.23 <i>EXTERNAL_ARMED</i>	5-24
5.2.24 <i>TURBO</i>	5-25
5.2.25 <i>*IDN?</i>	5-25
5.2.26 <i>GET_INS_FILE</i>	5-25
5.2.27 <i>MESSAGE</i>	5-26
5.2.28 <i>LOAD_UMT</i>	5-26
5.2.29 <i>LOAD_CONSTELLATION</i>	5-26
5.2.30 <i>LOAD_ITF</i>	5-27

5.2.31 <i>SIMULATION_ITERATION_RATE</i>	5-27
5.2.32 <i>GET_6700_HW</i>	5-28
5.3 Time commands	5-29
5.3.1 <i>TIME</i>	5-29
5.3.2 <i>TIME_STEP</i>	5-29
5.3.3 <i>GPS_TIME</i>	5-30
5.3.4 <i>UTC_TIME</i>	5-30
5.3.5 <i>STTIME</i>	5-30
5.3.6 <i>START_TIME</i>	5-30
5.3.7 <i>UTC_OFFSET</i>	5-31
5.3.8 <i>UTC_OFFSET_ACC</i>	5-31
5.3.9 <i>GPS_UTC_OFFSET</i>	5-32
5.3.10 <i>ZCNT_TOW</i>	5-32
5.3.11 <i>ZCNT_WEEK</i>	5-32
5.3.12 <i>ZCNT_WEEK_ROLLOVER</i>	5-32
5.3.13 <i>OFFSET_PROC_TIME</i>	5-33
5.3.14 <i>TIMER_PULSE</i>	5-34
5.3.15 <i>SET_TIOP</i>	5-34
5.4 Antenna Pattern commands	5-36
5.4.1 <i>ANT_PAT_NUM</i>	5-36
5.4.2 <i>SET_ANT_OFFSET</i>	5-36
5.5 Signal Power commands	5-37
5.5.2 <i>Shared parameters</i>	5-38
5.5.3 <i>POW_ON</i>	5-39
5.5.4 <i>POW_MODE</i>	5-39
5.5.5 <i>POW_LEV</i>	5-40
5.5.6 <i>REF_DBM</i>	5-41
5.5.7 <i>SIGNAL_STRENGTH</i>	5-41
5.5.8 <i>SAT_POW_OFFSET</i>	5-41
5.5.9 <i>LAAS_SIG_LEV</i>	5-42
5.6 Signal Control commands	5-43
5.6.1 <i>SWITCH_SAT</i>	5-43
5.6.2 <i>MP_SWITCH</i>	5-53
5.6.3 <i>MOD</i>	5-54
5.6.4 <i>SWITCH_LMM</i>	5-57
5.6.5 <i>FADER (embedded multipath)</i>	5-58
5.6.6 <i>CODE_LEV_OFFSET</i>	5-60
5.7 Pseudorange commands	5-61
5.7.1 <i>SVID rule</i>	5-61
5.7.2 <i>PR_RAMP</i>	5-61
5.7.3 <i>PR_ERRORS</i>	5-62
5.8 Data Streaming commands	5-63
5.8.1 <i>DS_ENABLE</i>	5-63
5.8.2 <i>DS_IP</i>	5-64
5.8.3 <i>DS_RATE</i>	5-64
5.8.4 <i>DS_STATUS</i>	5-64
5.8.5 <i>DS_SYNC</i>	5-65
5.8.6 <i>DS_VEH_MOT</i>	5-65
5.8.7 <i>DS_VEH_CMS</i>	5-65
5.8.8 <i>DS_ANT_MOT</i>	5-66
5.8.9 <i>DS_ANT_SIG</i>	5-66
5.8.10 <i>DS_INFO</i>	5-66
5.9	5-67
5.10 Motion commands (trajectory delivery)	5-67
5.10.1 <i>MOT</i>	5-67
5.10.2 <i>MOTB</i>	5-69
5.10.3 <i>AIDING_OFFSET</i>	5-70
5.11 Vehicle body axes	5-71
5.11.1 <i>Vehicle body attitude</i>	5-72
5.12 Hardware and calibration commands	5-73

5.12.1 RFOF	5-73
5.12.2 CAL	5-74
5.12.3 CAL_LEVL	5-74
5.12.4 NBLK2	5-74
5.12.5 PCAL	5-75
5.12.6 FW_CMD prefix	5-76
5.12.7 *IDN?	5-76
5.12.8 LOCAL_LOCKOUT	5-77
5.13 Data request commands	5-78
5.13.1 Vehicle Data Request	5-78
5.13.2 Antenna Data Requests	5-80
5.13.3 Signal Data Requests	5-82
5.13.4 Transmitter Data Requests	5-86
5.14 Navigation commands	5-87
5.14.2 GPS_NAV_DATA_ERR	5-88
5.14.3 GPS_NAV_DATA_MOD	5-89
5.14.4 WAGE	5-90
5.15 Navigation Data related commands	5-91
5.15.2 QZ_SYMB	5-91
5.15.3 GPS_LEGACY_NAV	5-92
5.16 Interferer commands	5-93
5.16.2 COHERENT_CW	5-94
5.16.3 NON_COHERENT_CW	5-95
5.16.4 SWEPT_CW	5-95
5.16.5 AM	5-95
5.16.6 FM	5-96
5.16.7 NOISE	5-96
5.16.8 PULSE - basic	5-97
5.16.9 PULSE - using prf and duty cycle	5-98
5.16.10 RF_ON_OFF	5-98
5.16.11 DELTA_LEVEL	5-99
5.16.12 LEVEL	5-99
5.16.13 FIXED_MODE	5-99
5.16.14 MODELLED_MODE	5-100
5.16.15 TX_ANTENNA_PATTERN	5-100
5.17 UDP commands	5-101
5.17.1 Command structure	5-101
5.17.2 MOT	5-102
5.17.3 MOTB	5-104
5.17.4 MOD	5-106
CHAPTER 6: PROPAGATION OF REMOTE MOTION DATA	6-1
6.1 Motion message formats	6-1
6.2 Application of remote motion data	6-1
6.3 High-end signal generators	6-2
6.4 Low-end signal generators	6-2
6.5 Descriptions of iteration rates	6-2
6.5.1 Iteration rate of 4 milliseconds	6-2
6.5.2 Iteration rate of 10 milliseconds	6-2
6.5.3 Iteration rate of 100 milliseconds (high-end platforms only)	6-3
6.6 Synchronisation of external motion data	6-3
6.7 Latency	6-3
6.8 Calculation of Prange control coefficients (100 ms iteration rate)	6-5
CHAPTER 7: SYNCHRONISATION IN CLOSED-LOOP MODE	7-1
7.1 General principles	7-1
7.2 Motion data alignment	7-3
7.3 Software start-up - trigger Disabled	7-4
7.4 External start pulse - Immediate trigger mode	7-6
7.5 Hardware start pulse - Delayed trigger mode	7-7

7.6 Using reduced data rates	7-8
7.7 Multiple unit configurations	7-8
7.7.1 <i>Two GSS8000 or a STR4780 and a GSS77xx / STR4760</i>	7-9
7.7.2 <i>Using the GSS4767 distribution unit</i>	7-9
7.8 GSS77XX / STR47XX timer inputs and outputs	7-10
7.8.1 <i>Timer output settings for remote operation</i>	7-11
7.8.2 <i>Using the 'sync in' input</i>	7-12
7.9 Synchronisation of GSS6560 inputs	7-13
7.9.1 <i>1PPS IN</i>	7-13
7.9.2 <i>TRIG IN - Immediate mode</i>	7-14
7.9.3 <i>TRIG IN - Delayed mode</i>	7-14
7.10 Configuring the GSS6560 for IEEE-488 remote control	7-15
7.11 Timer cable assembly (optional)	7-15
CHAPTER 8: DATA STREAMING	8-1
8.1 Introduction	8-1
8.2 Data streaming definition file	8-1
8.2.1 <i>Vehicle motion, antenna motion and signal data</i>	8-4
8.3 Data streaming client	8-5
8.3.1 <i>Implementation details</i>	8-6
8.3.2 <i>EthernetShare.h explained</i>	8-7
8.3.3 <i>Enhanced Data Streaming</i>	8-23
8.4 Implementing a client	8-28
8.4.1 <i>Implementing the supplied DLL</i>	8-28
8.4.2 <i>Putting it together</i>	8-32
8.4.3 <i>Custom client</i>	8-34
8.4.4 <i>Spirent example client</i>	8-41
8.5 Data streaming message logger utility	8-43
CHAPTER 9: ACRONYMS AND ABBREVIATIONS	9-1
CHAPTER 10: CONFIGURING REMOTE OPERATION	10-1
10.1 SimGEN controller	10-1
10.2 Optional IEEE-488 and Timer cards	10-1
10.3 Optional Timer card	10-1
10.4 Install the card	10-2
10.5 Install the drivers	10-2
10.6 Testing	10-2
10.7 Optional IEEE-488 card	10-3
10.7.1 <i>Install the drivers</i>	10-3
10.7.2 <i>Install the card</i>	10-3
10.8 Enable SimGEN for remote control	10-4
CHAPTER 11: CONFIDENCE TESTS	11-1
11.1 Install the IEEE test program on the remote PC	11-1
11.1.1 <i>Using the IEEE bus</i>	11-2
11.1.2 <i>Test using the remote motion IEEE software</i>	11-2
11.2 Options using the IEEE remote test program	11-5
11.2.1 <i>Change current scenario</i>	11-5
11.2.2 <i>Check SimGEN simulation status</i>	11-5
11.2.3 <i>Change to another trigger mode</i>	11-6
11.2.4 <i>Change motion message rate</i>	11-7
11.2.5 <i>Confidence test of the timer card</i>	11-7
11.2.6 <i>Run a remote motion demonstration</i>	11-8
11.3 Test using the remote motion TCP/IP software	11-10
11.4 Commands	11-11
11.4.1 <i>Change the RF power level settings</i>	11-12
11.4.2 <i>Stop/Start PRN code transmission</i>	11-12
11.4.3 <i>Set start location for a vehicle</i>	11-12
11.5 Log simulation progress to a remote PC	11-13

CHAPTER 12: OPTIONAL SCRAMNET CARD	12-1
12.1 Install the SCRAMNet card	12-1
12.2 Install SCRAMNet software and drivers	12-2
12.3 Confidence test 1	12-4
12.4 Confidence test 2	12-6
12.5 Confidence test 3	12-7
12.5.1 <i>Scramnet_std_remote_QPM939 test program</i>	12-8
12.6 Get SimGEN scenario status	12-9
12.7 Change scenario	12-10
12.8 Run a scenario	12-11
12.9 Run with motion	12-14
12.10 Log truth data	12-15
12.11 Other commands available	12-16
12.12 Remote commands for SCRAMNet	12-17
12.12.1 <i>MOTBIN</i>	12-17
CHAPTER 13: VMS REMOTE COMMAND COMPATIBILITY	13-1
CHAPTER 14: SPIRENT APPLICATIONS SUPPORT	14-1
INDEX	V
LIST OF FIGURES	IX
LIST OF TABLES	X

This page is intentionally blank

Chapter 1: General

Note: The term “simulator” refers to a configured arrangement of signal generators, computer controllers and ancillary items.

Spirent reserves the right to supply simulator elements that may be superficially and visually different to those shown in an interconnection drawing, but which provide equivalent or better performance.

This document defines the real-time external remote control and motion data interface for the Spirent series of multi-channel signal generators.

This document describes using SimREMOTE and includes references to an optional Timer card and optional IEEE-488 / SCRAMNet interfaces.

SimGEN uses the full set of SimREMOTE commands. Reference i) details Spirent’s SimREPLAY and SimREPLAYplus software, which uses a sub-set of the SimREMOTE commands.

1.1 This issue

Note: “SimGEN release” refers to the SimGEN release for which this user manual is valid. Earlier releases of SimGEN may have reduced functionality compared with this release. Spirent recommends you always use the latest release.

Issue	Date	SimGEN release	Description
2-16	Mar 2010	2-90	<p>Typos Changed Appendices to Chapters for Help file Changed appropriate SimGEN references to PosApp</p> <p>Bugz: 6469: RFOF command support for GSS6700 6419: MOD command 6405: Addition to START_TIME command 6402: Update rates 6376: CODE_LEV_OFFSET updated 6364: Extra details for UDP commands 6357: Spirent support details</p>

1.2 Referenced documents

- DGP00686AAA SimGEN Software User Manual.
- ANSI/IEEE Std. 488.1-1987. IEEE Standard Digital Interface for Programmable Instrumentation.
- DGP00703AAA Signal Generator Hardware User Manual.
- SCRAMNet+ Network SC150e PCI, PCM and CPCI Bus (Universal Signalling) Hardware Reference.
- SCRAMNet+ Network Software Installation Manual for x86 Platforms.
- SCRAMNet+ Network Windows DLL Reference Guide.
- MS3015 SimREMOTE Product Specification.
- DGP00736AAA GSS4765 ISS User Manual
- DGP01074AAA GSS6700 using SimREPLAY and SimREPLAYplus User Manual

1.3 Copyright notices

Microsoft and Windows are either trademarks or registered trademarks of Microsoft Corporation.

InstallShield is a registered trademark of InstallShield Software Corporation.

Alpha is a registered trademark of Hewlett Packard.

All other company/product names referenced herein are trademarks and/or service marks or registered trademarks and/or service marks of their respective holders.

1.4 Document conventions

Table 1-1 shows the format conventions used by this document.

Table 1-1: Document format conventions

Format	Description
Clear	Button name (with extra description to avoid ambiguity)
Edit	Menu item (with extra description to avoid ambiguity)
<i>SimGEN.exe</i>	Filename
Note: <i>Insert the dongle</i>	Important information
<i>"Text"</i>	Text entered from keyboard
Signal control - General	Title of dialog or window (with extra description to avoid ambiguity)
Signal Definition	Title of area within a dialog or window

Chapter 2: SimREMOTE overview

Notes:

- 1) Enhanced Data Streaming is only available to authorised users. To use Enhanced Data Streaming, you must insert the dongle, supplied by Spirent to authorised users, in a USB port on the PC before you start SimREMOTE.*
- 2) Except where SimGEN takes control and motion data directly from a local data file; Spirent recommends you slave the motion generator to a digital time reference for accurate synchronisation of the remote motion data.*
- 3) You can use any standard interface or any optional interface to inject remote control commands or trajectory/motion data.*
- 4) Spirent normally supplies the SimREMOTE PC. However, instead of the SimREMOTE PC, you can use any system that uses one of the interfaces shown. Specific requirements, such as remote motion, may need specific hardware.*

SimREMOTE is a package of Remote Control facilities that greatly enhances the flexibility of SimGEN. Except where specified this document refers to SimGEN using SimREMOTE commands.

SimREPLAY and SimREPLAYplus can use a sub-set of the SimREMOTE commands, using the same syntax, see reference i) for details of the supported commands.

Depending on the scale of your testing, these facilities can meet every need from simple remote control through to full integration with your test equipment and systems.

The basic remote control features enable effective management of the testing process:

- a) Scripted test sequences ensure precise execution of tests.
- b) Automated test sequences to maximise utilisation.
- c) SimGEN can be slaved to a central test controller/motion generator.
- d) Scalable, with basic features operated from local script files.
- e) Integrated with remote units using TCP/IP.
- f) Can use optional IEEE-488 and SCRAMNet interfaces.
- g) Can use an optional Timer card for synchronisation with other test equipment

Adding remote motion control enhances the basic remote control features:

- a) Minimum requirement is position and velocity on three axes.
- b) Better fidelity when
- c) acceleration and jerk supplied.
- d) Automatic interpolation where motion data rate does not match SimGEN rate.
- e) Cubic spline applied for accurate interpolation.
- f) Gaps in motion data tolerated.
- g) Injection of customised motion for multiple vehicles.
- h) Allows Open- and Closed-loop motion control.

2.1 Timer card option

As delivered, SimREMOTE operates asynchronously. An optional Timer card allows synchronous operation. In synchronous operation, the motion generation software interrogates the Timer card to get current simulation time and injects motion data at the appropriate time. A suitable Timer card is included in the SimREMOTE Option packs.

2.2 Interface options

As delivered, SimREMOTE provides the following interfaces:

- a) TCP/IP (Sockets) either over a network or locally at the PC.
- b) RS232 serial link.
- c) Local disk command file.

Using optional hardware, SimREMOTE can use the following interfaces:

- a) IEEE-488 bus.
- b) SCRAMNet shared memory interface card.

Details of the SimREMOTE Options available are given in reference g), which is available from Spirent on request.

Figure 2-1 shows a schematic of the standard SimREMOTE (giving asynchronous operation with TCP/IP, RS232 and command line interfaces); together with the optional Timer card (giving synchronous operation) and optional IEEE-488 / SCRAMNet interface cards (giving a wide choice of interfaces).

Figure 2-1: SimREMOTE synchronisation and interface options

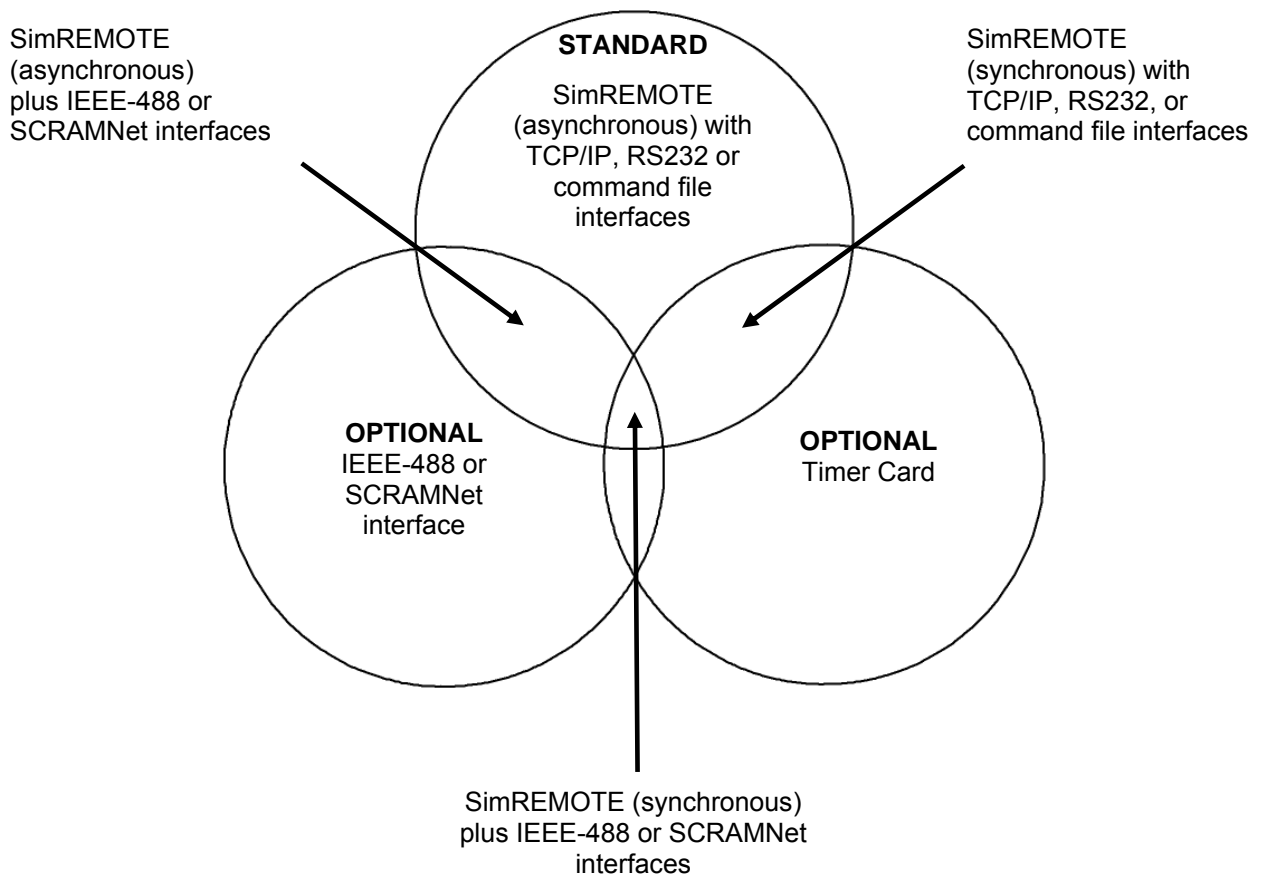


Figure 2-2 shows all the interconnections between the signal generator, the SimREMOTE PC and the SimGEN PC.

Figure 2-2: SimREMOTE - interconnection diagram

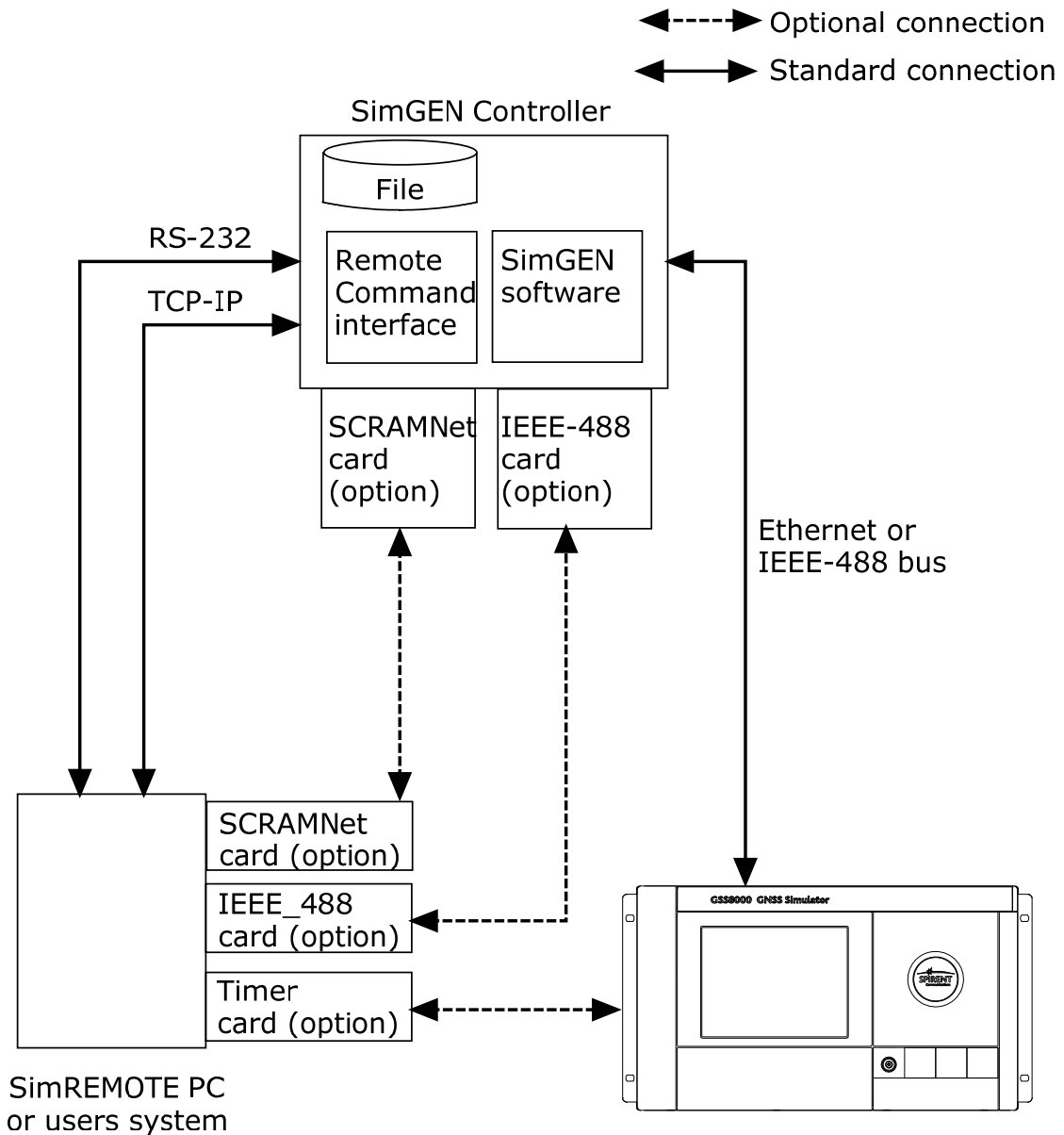


Table 7-6 gives details of the optional Timer Cable Assembly used to connect the SimREMOTE PC system to the signal generator.

This page is intentionally blank

Chapter 3: Principles of operation

SimGEN generates the simulated RF signal dynamically in real-time without the need to carry out pre-processing of vehicle motion or satellite signals. This makes your system easier to use and more productive; and makes it possible for your system to respond to and follow dynamic vehicle motion generated by another system.

The SimREMOTE package provides all the facilities for another system to operate SimGEN and generate RF signals. This is remote operation.

SimREMOTE supports multiple-vehicle and multiple-antenna modes of operation.

The remote option lets you use another computer to control SimGEN and:

- a) Select an alternative scenario,
- b) Select trigger mode and start / stop execution of the scenario,
- c) Amend the simulation start time and date,
- d) Amend the selection of the currently simulated satellites,
- e) Modify power levels and introduce pseudorange errors or multipath,
- f) Select an alternative antenna pattern,
- g) Drive the motion of multiple simulated vehicles.

3.1 Remote motion - Simulation iteration rate

SimREMOTE uses a remote computer system to simulate vehicle motion. The remote computer system continuously updates the host SimGEN PC with motion data while a simulation is in progress. SimGEN periodically samples this motion data in real-time to calculate the pseudoranges needed by the signal generator. Changes in vehicle motion are reflected with low latency in the RF output of the signal generator. Latencies less than 10 ms are achievable, depending on the performance of the particular remote system PC and the signal generator configuration.

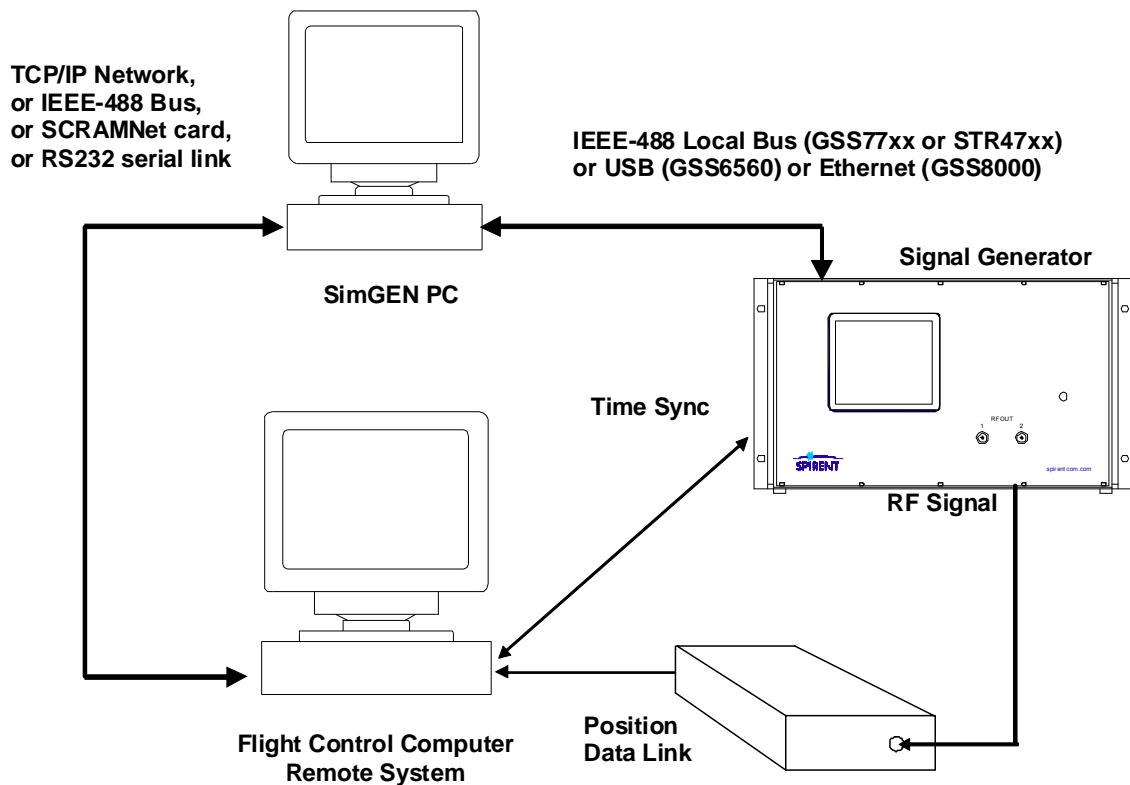
The period at which SimGEN is set to sample the supplied motion data and iterate its motion models and pseudorange calculations is termed the **Simulation iteration rate**. This rate is specified as a time in ms with “every” being implied. The simulated time within SimGEN thus advances in discrete **time steps**.

SimGEN also sends updated pseudorange trajectory data to the signal generator hardware unit at the same rate. However, the embedded software of the signal generator typically uses a faster rate to update the hardware control registers.

3.2 Remote motion - hardware set-up

The remote system sends motion data messages to the host PC using a TCP/IP network, an optional SCRAMNet interface card or an optional dedicated IEEE-488 bus. In the case of an IEEE-488 link, the remote system acts as the bus controller and the SimGEN PC as a talker-listener. This document refers to the remote system as the ‘controller’.

Figure 3-1 shows a typical closed-loop system.

Figure 3-1: Typical closed-loop system configuration (such as a flight simulator)

For correct operation of the remote option in a closed-loop mode, it is essential to synchronise the remote system with the signal generator, see section 3.3.

SimGEN allows considerable flexibility in the supply of motion data. Data rates corresponding to intervals from 4 to 1000 ms are accepted. SimGEN is also tolerant of variations in the data rate and occasional missing messages. The motion data rate need not match the SimGEN simulation iteration rate / time step. However, optimum performance occurs when the motion data matches the SimGEN time step epochs. Ideally, this is with one motion data message per time step and with timestamps aligned to the time steps and transmitted sufficiently in advance so they can be utilised to compute the control data through a time step. Section 6.2 covers this in detail and section 6.3 covers timing.

3.3 Remote control - timing and synchronisation

Precise timing and synchronisation of the remote system requires hardware components to monitor and react to the timing signals output by the signal generator. If precise timing is not required and in particular if the requirement is control only without injection of vehicle motion then the extra hardware is not required. Approximate timing can be determined by interrogating the SimGEN PC with the 'TIME' command (see section 5.3.1) and using the remote computer's own clock to track the progress of the scenario time. The 'NULL' command (see section 4.4.2.12) will respond with the current status code value for SimGEN. The transition to the 'running' state can be observed by polling with the 'NULL' command and looking for the change in the status code. If the remote computer's time is noted when this is observed, the computer's clock can be used to derive the approximate scenario 'time into run'. Control commands may then be applied at the desired times.

The latency inherent in the communications between the remote system and SimGEN mean that the time derived by this method may lag the signal generator's time by as much as half-a-second. To a degree, this is overcome by specifying a timestamp on the control commands, as SimGEN will invoke the commands at the scenario time specified. However, it will be necessary for the remote system to send the commands substantially in advance of the target time to compensate for the

lag. This means that it is unlikely that satisfactory results will be achieved with motion data injection.

An alternative technique is to use the Data Streaming facility to provide notification of changes in the SimGEN PC status. The 'status' message (see section 5.7.4) will provide notification that the scenario has started. The 'sync' message (see section 5.7.5) can provide a 'time signal' at one-second intervals. This technique results in a shorter lag than can be obtained by polling with the remote commands. A remote control program using this technique will need to be designed as an event driven system possibly with multiple threads and thus is more demanding to implement. Spirenta demonstration program using this technique.

This page is intentionally blank

Chapter 4: Introduction to remote control

Notes:

- 1) *The command syntax is common to all source types.*
- 2) *You cannot remotely set “on” a command that is set “off” at the start of a scenario. To ensure you can subsequently set a command “on” and “off” using remote commands, ensure you have first set the command “on” in the scenario.*
- 3) *Spirent recommends you do not control the same vehicle with different remote sources.*
- 4) *If you use multiple remote sources, such as **File** and **BSD Sockets**, you must carefully control the command synchronisation between the two sources.*

This Chapter defines the remote control commands supported by SimGEN, and SimREPLAY and SimREPLAYplus software. You can use a PC, configured as described in section 3.2, to send these commands to SimGEN; or you can use a command file stored in the SimGEN PC, see the SimGEN User Manual, reference a). You can create different types of command file, depending on whether you need the file associated with a specific scenario (the *.ucd file) or not (the *.act file).

If you only want to change motion parameters, Spirent recommends you use a User motion file (the *.umt file), see reference a).

SimREMOTE remote control facilities can be broken down into two categories:

Direct Control - remotely configuring and controlling a scenario without using the SimGEN interface. Examples of Direct Control are scenario selection, scenario start/stop, satellite power level variation and the logging of requested data.

Trajectory Delivery - a special case of remote operation that uses vehicle motion data you provide, instead of SimGEN's built-in vehicle motion models.

Section 4.2 details how to configure SimGEN for remote operation, together with descriptions on all supported remote source types.

Section Chapter 5: gives the complete remote control command set. Appendix E: gives a comparison of the legacy SimGEN for VMS commands with the Windows commands.

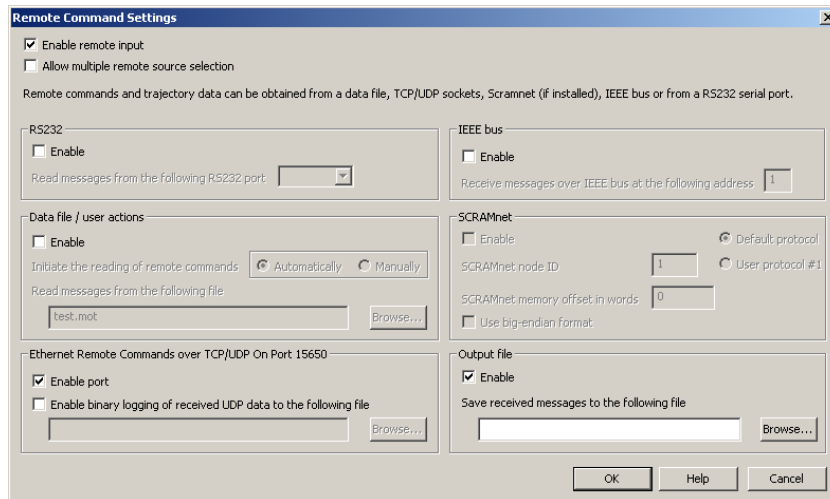
4.1 Selecting a remote source in SimGEN

Notes:

- 1) *Only use manual mode when reading remote commands from a file. In ‘automatic’ mode the external program initiates control. In either mode, a remote run command, ‘RU’ (see section 5.2.8) is required to start the scenario. If a run command is not present, you must start the run by clicking **Run** on the SimGEN toolbar.*
- 2) *When SimGEN is under remote control, it is undesirable that message pop-ups occur, as these messages require acknowledgement before the application resumes. You can suppress these pop-ups using **Options - Message Reporting - Disable message popups**.*

This section amplifies the instructions given in section B.5 and details the remote sources supported by SimGEN.

Use **Options-Remote Command Settings** to select a remote command source, see Figure 4-1.

Figure 4-1: Remote Command Settings dialog

Select **Enable remote input** to activate remote control.

You can select one or more sources. For each source, you must complete the associated items to enable that source.

For **Data file - user actions**:

Select **Initiate the reading of remote – Automatically**, to begin reading the remote commands immediately you close the **Remote Command Settings** dialog.

Select **Initiate the reading of remote – Manually** to begin reading the remote commands when you click the **Remote** button on the SimGEN toolbar.

4.1.1 Command file

Using the syntax described in section 4.2, a series of commands may be placed in an ASCII file (with a file extension *.act) to define a test sequence that you can run repeatedly or unattended. To do this:

- Write the commands given in section 4.2 to a file.
- Save the file with the file extension *.act.
- Select the menu **Options - Remote command settings**.
- Select **Enable remote input**, **Data file / user actions - Enable** and **Initiate the reading of remote - manually**.
- Click **Browse...** and navigate to the command file you created.
- Click **OK**.
- Exit **Remote Command Settings** by clicking **OK**.
- To execute the command file, click on the remote control icon on the toolbar.

4.1.1.1 Commented example command file

(Comments are **bolded** and should **NOT** be placed in the file):

```
SC,C:\scenarios\example\example
.sim
RU
0 00:05:00,EN,1,1
SC,C:\scenarios\example2\exampl
```

Select "example.sim" in the folder
example

Run it

Halt run after 5 minutes and rewind
scenario

Select "example2.sim" in the folder

```
e2.sim
RU
0 00:05:00,EN,2,1
```

example2

Run it

Halt run after 5 minutes, rewind scenario and rewind command file and repeat (that is, loop forever running the 2 scenarios alternately)

4.1.2 IEEE-488 interface

“High-end” systems (GSS47xx / 77xx) require a second IEEE-488 card for remote control. These systems use the first IEEE-488 card to control the signal generator.

The second card option uses a National Instruments IEEE-488 card, which must be installed in the PC running the SimGEN software (PCMCIA version for a laptop, PCI version for a desktop) together with the National Instruments NI-488.2 driver software.

If Spirent has not pre-installed this upgrade, follow the installation instructions supplied by the IEEE-488 card manufacturer and see section B.4.

With **Remote Commands Settings-IEEE bus** selected, see Figure 4-1, the SimGEN software effectively becomes a talker-listener instrument (at address 1 by default). An external computer equipped with an IEEE-488 interface and acting as a Bus Controller may control it by issuing any of the commands listed in section 4.3.

Currently SimGEN does not use the SRQ mechanism to flag errors. Fatal-level software errors in SimGEN are returned to the remote user appended to the xml message returned when an IEEE-488 read is performed, see section 5.1.2.)

Each time a command is received, simulation status data is written to the serial poll response byte. You can access this data by performing a Serial Poll on the SimGEN device address. In addition the Bus Controller may read a serial poll status byte from SimGEN by performing a serial poll operation. This status byte contains the simulation state. Table 4-1 shows bits 0 to 3 of the simulation status.

Table 4-1: Simulation status bit description

Bits 0 to 3	Description	Decimal message number
0000	No scenario specified	0
0001	Loading	1
0010	Ready	2
0011	Arming	3
0100	Armed	4
0101	Running	5
0110	Paused	6
0111	Ended	7

Bit 4 set = “Scenario is incomplete” (not all source files specified, will not run)

Bit 5 set = “Incompatible hardware” (declared hardware not compatible with specified scenario)

Bit 6 is specified in the IEEE-488 standard as the SRQ flag, but this is currently not used and the SimGEN software will never raise a service request (SRQ)

Bit 7 not used at present.

To assist you in writing your own control program, or incorporating SimGEN control into an existing program, the source code for a simple example program, `port_ieee_test.exe` is included on the SimGEN distribution CD, in the *Remote Motion IEEE* folder.

4.1.3 TCP/IP sockets interface

Note: *The client program must connect to port 15650.*

“High-end” systems using the GSS80000 series of signal generators can use this interface without additional cards or software. For single unit systems connect the Ethernet cable from the remote PC to the spare Ethernet port on the SimGEN controller. Do not connect the remote PC to the SimGEN controller port (labelled ETH-1-PC), this port is used by the signal generator.

This option allows either another program running on the same PC or on a remote PC connected using Ethernet and TCP/IP to control the SimGEN software.

Select **Remote Command Settings-BSD sockets** see Figure 4-1.

The SimGEN software acts as a server and the remote program as a client. SimGEN must be active and **Remote Command Settings-BSD sockets** selected before the client attempts to connect.

Once connected, you can issue all the commands specified in section Chapter 5:

A read command must be executed after each individual command write by the client. This will return simulation status information and the last fatal error message (if there is one) in XML format. Section 5.1.2 describes the error message format. The status byte contains the simulation state and is identical to the status byte for the IEEE-488 interface see section 4.1.2.

To assist you in writing a control program or incorporating SimGEN control into an existing program the source code for a simple example program is included on the SimGEN distribution CD. This code is installed with SimGEN and can be found in the folder Remote Motion TCPIP.

4.1.4 RS-232 interface

Notes:

- 1) *Spirent recommends using the RS-232 interface with a high baud-rate.*
- 2) *Do not use the RS-232 interface when you need high data throughput.*
- 3) *SimGEN can only open COM ports up to and including COM9.*

Select **Remote Command Settings-RS-232** and specify an RS-232 port, see Figure 4-1. Setting the RS-232 port characteristics is described in the SimGEN User Manual, see reference a).

Operation in this mode is identical to that as described in section 4.1.3.

4.1.5 SCRAMNet interface

Note: *SimGEN does not support “burst mode”.*

For this option, an optional SCRAMNet card must be installed in the PC running the SimGEN software see Appendix D: and a SCRAMNet card must be installed on the Remote system.

Select **Remote Command Settings-SCRAMNet** and select **Default protocol**.

Set the **SCRAMNet node ID** and specify the **SCRAMNet memory offset**.

Select **Use big-endian format** to swap bytes of data written to SCRAMNet. This is required in networks in which other nodes have big-endian processors, or are operating in big-endian format.

The default protocol defines two areas of memory, one for SimGEN commands to be written to, and one for SimGEN to write responses into. Each of these is controlled by two flags: “busy” and “new”. “Busy” is set while the memory is being written to, when the write operation has finished “busy” is unset and “new” is set to indicate that there is new data available. While the data is being read, “busy” is set. When a read operation is completed, “busy” is unset and “new” is unset to indicate that this data has been read. By default, the “command” area of the memory is near the beginning of the SCRAMNet memory. 4000 bytes of space are allowed for commands and the “response” area follows immediately after. It is possible to shift both of these areas further into SCRAMNet memory by specifying the SCRAMNet memory offset. You must allow sufficient space for SimGEN’s responses and for the SimGEN data buffer if logging data to SCRAMNet, as described in section 4.1.5.1.

Table 4-2: Memory map for default SCRAMNet protocol

Address, bytes	Data
68	New flag (long int)
72	Busy flag (long int)
76	Command (A null-terminated string starting at this address. 4000 bytes have been allowed for this so even the longest commands can be accommodated.)
4084	New flag (long int)
4088	Busy flag (long int)
4092	Response (A null-terminated string starting at this address.)
8184	Truth data read index (long int) Indicates the position where SimGEN will write the next truth data message
8188	Truth data write index (long int) Indicates the position of the next data to be read by the remote application.
8192	Truth data. Buffer size = 1000 messages where each message is 512 bytes

To assist you to write a control program, or incorporate SimGEN control into an existing program, Spirent have included the source code for a simple example program on the SimGEN distribution CD. This code is installed with SimGEN and can be found in the folder Scramnet/ Std Remote. When creating a control program, remember SimGEN utilises longword mode and be sure to use that mode.

4.1.5.1 Data streaming to SCRAMNet

In addition to commands and responses, you can configure SimGEN so the truth data that is broadcast using Ethernet can also be sent to SCRAMNet.

To do this, select at least one vehicle or antenna, then select **Copy to SCRAMNet** in the data streaming definition file and select **Data streaming**. After each time step interval, SimGEN may write some truth data messages to the truth data buffer and then writes the address where the next truth data will go to the truth data write index. The next time SimGEN comes to write truth data, it will use this information to know where to begin. The remote application can also use the truth data write index to ensure that it does not read past that part of the buffer that has been updated by SimGEN. The remote application can also use the truth data read index to keep track of where to start reading next time.

Chapter 8: describes the Data Streaming format.

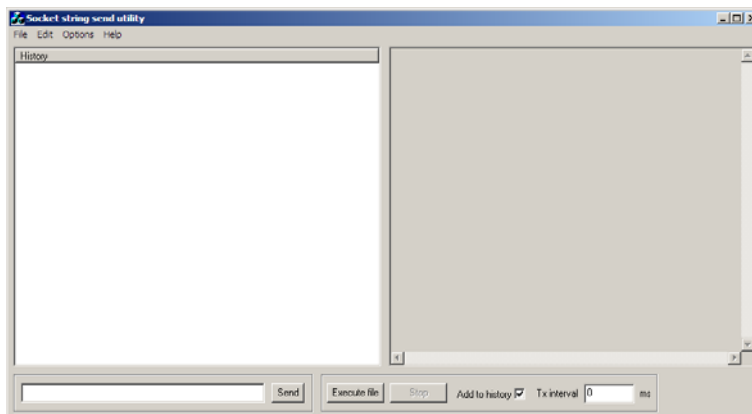
4.2 Remote command example - running a scenario

Notes:

- 1) When running in remote mode you must prevent message pop-ups by selecting **Options-Message reporting-Disable message popups**.
- 2) You can send remote commands using the SimGEN controller without connecting hardware to it. In this case, do not select **Options-General options-Display a warning before running without hardware**.
- 3) You must correctly format each remote command. Table 5-3 gives references to each command and the section specifying the command format.

You can use the **Socket string send** utility to send any of the commands detailed in Chapter 5: to your signal generator. Figure 4-2 shows the utility.

Figure 4-2: Socket string send utility dialog



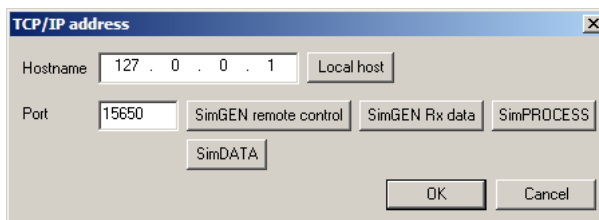
Select **Add to history** to add the commands you type in the **Send** text area to a text (*.txt) file. Save the history file to the current scenario folder (by default), or you can choose the file location, using **File-Save History**. If you have selected **Add to history**, each time you close the utility you will be asked if you want to save the history.

Load a History file using **File-Load History**. Run this History file by clicking **Execute File**. The commands in the History file will be sent at the interval (in milliseconds) you enter in the **Tx interval** text box.

Click **Stop** to stop sending the History file.

You can set commonly used IP addresses using **Options-TCP/IP address...**, see Figure 4-3.

Figure 4-3: TCP/IP address



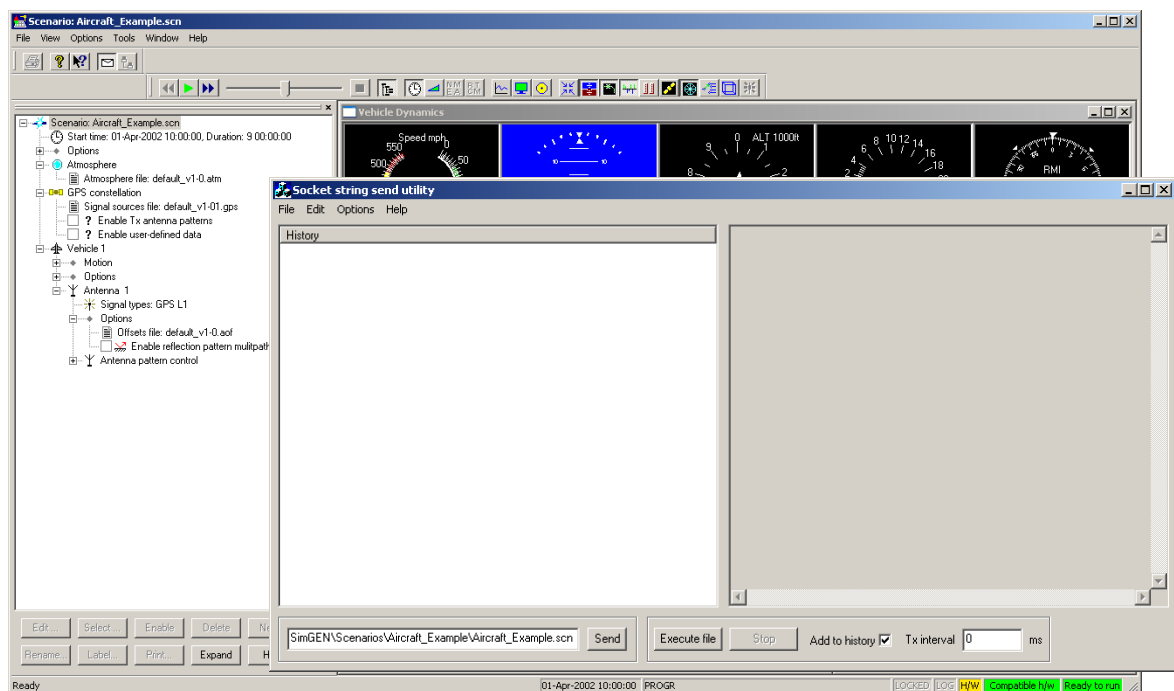
Select **Options-Output timings in log file** to record read write timings in the log file.

To run a scenario using remote commands:

- a) Start SimGEN.
- b) Close any scenario that may automatically start.
- c) Select Options-Remote Command Settings-Enable remote input.

- d) Select Options-Remote Command Settings-Ethernet Remote Commands over TCP/UDP On Port 15650-Enable port.
- e) Select Tools-General Utilities-Socket String Send.
- f) Choose a scenario you want to run. This example uses the scenario Aircraft_Example.scn located in the Aircraft_Example folder in the default Scenarios folder,
C:\Program Files\Spirent Communications\PosApp\Scenarios\Aircraft_Example\
- g) To run this scenario, in the Send area of the Socket string send dialog, type:
"SC,C:\Program Files\Spirent Communications\PosApp\Scenarios\Aircraft_Example\Aircraft_Example.scn". Section 5.2.3 gives details of the SC command.
- h) Click Send to select but not run the scenario, see Figure 4-4.

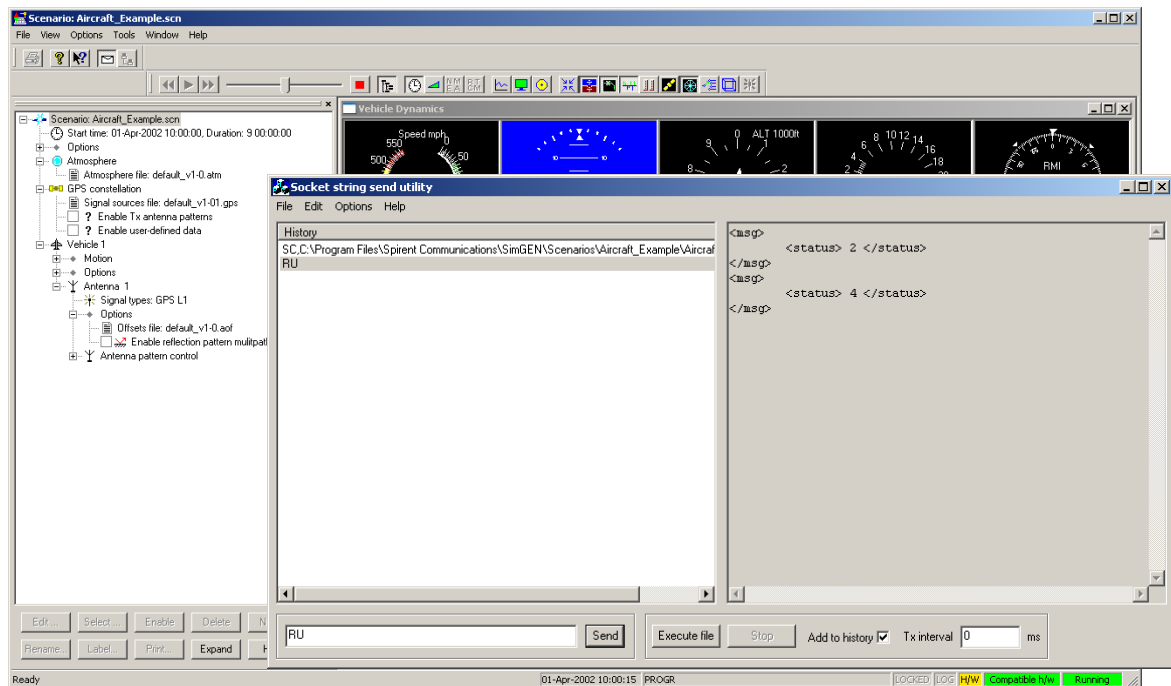
Figure 4-4 Socket string send utility



- i) The bottom right-hand corner of the SimGEN status bar shows Ready to run.
- j) A message appears in the right-hand control area of the Socket string send dialog:

```
<msg>
  <status> 2 <\status>
</msg>
```

 Table 4-1 shows that message status 2 indicates "Ready".
- k) Delete any text in the Send text area.
- l) Type "RU" and click Send to run the scenario, see Figure 4-5.

Figure 4-5: Using Socket string send utility – Run

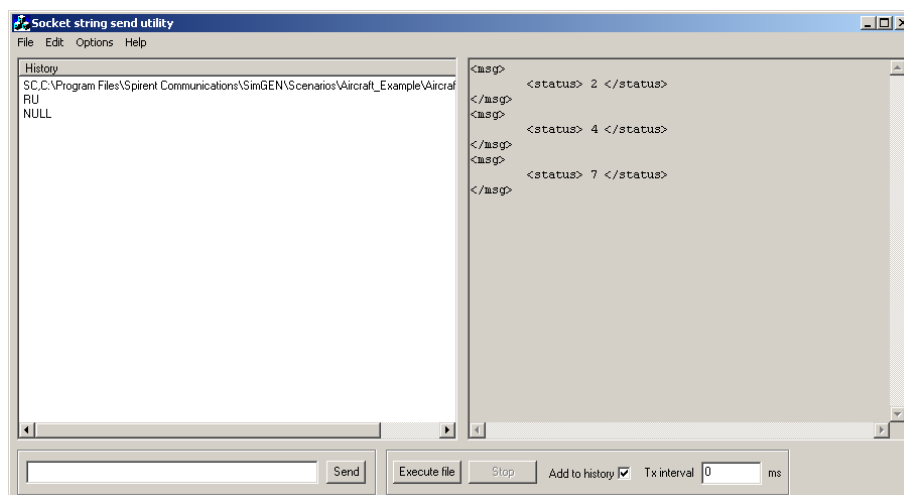
- m) The bottom right-hand corner of the SimGEN status bar shows Running.
- n) A new message appears in the right-hand control area of the **Socket string send** dialog:

```
<msg>
  <status> 4 </status>
</msg>
```

Table 4-1 shows that message status 4 indicates “Armed”.

Note: The status bar message represents the scenario status immediately after clicking **Send**. You may not see the bottom right-hand corner briefly display “Armed”, before the scenario starts running.

- o) You can check the status of the scenario at any time by typing “NULL” into the **Send text** box and clicking **Send**, see Figure 4-6.

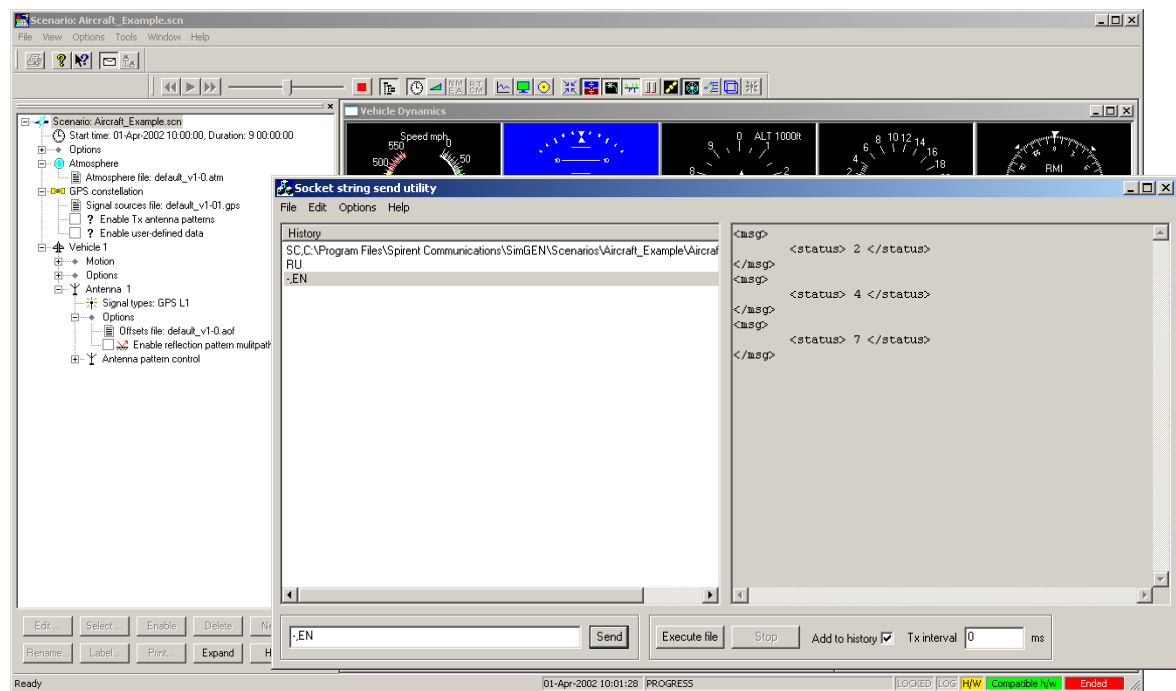
Figure 4-6: Using Socket string send utility – Null

- p) A new message appears in the right-hand control area of the Socket string send dialog:
- ```
<msg>
 <status> 5 </status>
</msg>
```

Table 4-1 shows that message status 5 indicates “Running”.

- q) To end the scenario immediately, type “-,EN” (see section 5.2.10) in the **Send text** area and click **Send**, see Figure 4-7.

Figure 4-7: Using Socket string send utility - End



- r) The bottom right-hand corner of the SimGEN status bar shows Ended.
- s) A new message appears in the right-hand control area of the **Socket string send** dialog:
- ```
<msg>
    <status> 7 </status>
</msg>
```

Table 4-1 shows that message status 7 indicates “Ended”.

This page is intentionally blank

Chapter 5: Remote commands

Notes:

1) From SimGEN version 2-41 onwards, remote commands are case insensitive. For example, *mot*, *Mot*, *moT* or *MOT* works equally well.

2) The command syntax applies whichever interface (such as IEEE-488, TCP/IP, RS-232, SCRAMNet) you use.

5.1 Introduction

The following sections detail the remote commands. For readability, categories are used to group commands of a similar type, for example commands associated with scenario setup, see Table 5-1.

Table 5-1 Index of remote commands

Remote command set	See
Scenario commands	Table 5-3
Time commands	Table 5-4
Antenna Pattern commands	Table 5-5
Signal Power commands	Table 5-6
Signal Control commands	Table 5-7
Pseudorange commands	Table 5-8
Data Streaming command	Table 5-9
Motion commands	Table 5-10
Hardware and Calibration commands	Table 5-11
Vehicle Data Request commands	Table 5-12
Antenna Data Request commands	Table 5-13
Signal Data Request commands	Table 5-14
Transmitter Data Request commands	Table 5-15
Navigation commands	Table 5-16
Navigation Data related commands	Table 5-17
Interferer commands	Table 5-18

Before you continue, Spirent recommends you review Table 5-2, which covers the remote command syntax.

Table 5-2 Remote command syntax

Symbol	Description	Example
< >	Category name	<number>
,	Delimiter between categories	VEH_X_POS,v1
Δ	Space in command	<d>Δ<hh>:<mm>:<ss>
[]	Optional parameter	[<d>Δ]<hh>:<mm>:<ss>
	'OR'	<timeFormat> <actionImmediately>
::=	Defined	ss::= 0-59
" "	Literal	"1" type 1
{ }	Zero or more times	<number> <string>{,<number> <string>}
-	Range	0-23
()	Descriptive text	(defaults to 0 if not used)

Table 5-3 Scenario commands

Command	Description	Section
Scenario commands		5.2
SET_VEHICLE_DIAL_TYPE	Sets the type of vehicle dial SimGEN uses.	5.2.1
SET_VEHICLE_DIAL_UNITS	Sets the units of the vehicle speedometer(s)	5.2.2
SC	Select a scenario	5.2.3
SC_NAME	Returns currently selected scenario name	5.2.4
SC_DURATION	Returns currently selected scenario duration	5.2.5
INIT_POS	Set a vehicles initial position	5.2.6
TR	Set trigger mode	5.2.7
RU	Run a scenario	5.2.8
RU_NOWAIT	Runs scenario without waiting for ARMED state.	5.2.9
EN	End a running scenario	5.2.10
RW	Rewind an ended scenario	5.2.11
AR	Arms the scenario	5.2.12
AR_NOWAIT	Makes remote control available during arming	5.2.13
NULL	Returns the status of the scenario	5.2.14
LOAD_ALMANAC	Loads a YUMA format almanac file	5.2.15
WRITE_ALMANAC	Writes a YUMA format almanac file	5.2.16
LOAD_RINEX	Loads a RINEX format file	5.2.17
WRITE_RINEX	Writes a RINEX format file	5.2.18
LOAD_ACT	Loads a user action file	5.2.19
LOAD_UCD	Loads a user command file	5.2.20
REGISTER_EXTERNAL_ARM	Registers an external arm source.	5.2.21
DEREGISTER_EXTERNAL_ARM	De-registers a registered external arm source	5.2.22
EXTERNAL_ARMED	Tells SimGEN an external source has armed	5.2.23
TURBO	Enables Turbo mode	5.2.24
*IDN?	Return the version of SimGEN	5.2.25
GET_INS_FILE	Returns the XML file of each vehicle number	5.2.26
MESSAGE	Shows a message in System Messages	5.2.27
LOAD_UMT	Loads a user motion file	5.2.28
LOAD_CONSTELLATION	Loads a constellation file	5.2.29
LOAD_ITF	Loads an interferer file	5.2.30
SIMULATION_ITERATION_RATE	Sets the simulation iteration rate	5.2.31
GET_6700_HW	Queries capability of GSS6700 signal generators	5.2.32

Table 5-4 Time commands

Command	Description	Section
Time commands		5.3
TIME	Retrieves the current time into run when the command is received.	5.3.1
TIME_STEP	Retrieves the time into run of the current time step	5.3.2
GPS_TIME	Retrieves the number of seconds from GPS time zero.	5.3.3
UTC_TIME	Retrieves the scenario time as an ASCII string.	5.3.4
STTIME	Retrieves the start time of the current scenario	5.3.5
START_TIME	Sets the start time and duration of the current scenario	5.3.6
UTC_OFFSET	Retrieves GPS to UTC offset	5.3.7
UTC_OFFSET_ACC	Set the UTC offset acceleration	5.3.8
GPS_UTC_OFFSET	Retrieves GPS to UTC offset, at a given time	5.3.9
ZCNT_TOW	Retrieves the z-count TOW in seconds	5.3.10
ZCNT_WEEK	Retrieves the z-count week number	5.3.11
ZCNT_WEEK_ROLLOVER	Retrieves the z-count week rollover number	5.3.12
OFFSET_PROC_TIME	Applies a fixed delay to signal control data processing.	5.3.13
TIMER_PULSE	Sends a single pulse from an optional timer card	5.3.14
SET_TIOP	Reads or sets the timer output.	5.3.15

Table 5-5 Antenna Pattern commands

Command	Description	Section
Antenna Pattern commands		5.4
ANT_PAT_NUM	Sets the antenna pattern to use at the given time	5.4.1
SET_ANT_OFFSET	Sets the antenna offset to use at the given time	5.4.2

Table 5-6 Signal Power commands

Command	Description	Section
Signal Power commands		5.5
POW_ON	Set the power on or off by Channel or SVID	5.5.3
POW_MODE	Obsolete command - see POW_LEV	5.5.4
POW_LEV	Set the power level by Channel or SVID	5.5.5
REF_DBM	Sets the reference label	5.5.6
SIGNAL_STRENGTH	Sets signal into fixed or modelled mode	5.5.7
SAT_POW_OFFSET	Offsets the RF Power level of a given SVID.	5.5.8
LAAS_SIG_LEV	Set the LAAS signal level	5.5.9

Table 5-7 Signal Control commands

Command	Description	Section
Signal Control commands		5.6
SWITCH_SAT	Defines the state of signals for visible satellites.	5.6.2
MP_SWITCH	Used with MOD command to define multipath signals	5.6.3
MOD	Defines level, code and carrier offset data for an SVID	5.6.4
LMM_SELECT	Select Land Mobile Multipath Model attributes	5.6.5
FADER (embedded multipath)	Add up to 4 multipath signals (sub-channels) per Channel	5.6.6
CODE_LEV_OFFSET	Applies signal level control to each component (C/A, P, Y)	5.6.7

Table 5-8 Pseudorange commands

Command	Description	Section
Pseudorange commands		5.7
PR_RAMP	Pseudorange ramp	5.7.2
PR_ERRORS	Pseudorange errors	5.7.3

Table 5-9 Data Streaming commands

Command	Description	Section
Data Streaming commands		5.8
DS_ENABLE	Enable or disables Data Streaming	5.8.1
DS_IP	Specifies the Data Streaming I.P. address	5.8.2
DS_RATE	Specifies the Data Streaming update rate	5.8.3
DS_STATUS	Enables the transmission of status messages	5.8.4
DS_SYNC	Enables the transmission of time synchronisation messages	5.8.5
DS_VEH_MOT	Enables the transmission of vehicle motion messages	5.8.6
DS_VEH_CMS	Enable transmission of car motion sensor data messages.	5.8.7
DS_ANT_MOT	Enables the transmission of antenna motion messages	5.8.8
DS_ANT_SIG	Enables the transmission of signal data messages	5.8.9
DS_INFO	Returns data streaming capabilities of the running SimGEN	5.8.10

Table 5-10 Motion commands

Command	Description	Section
Motion commands		5.9
MOT	Vehicle motion using ECEF axes with respect to WGS84	5.10.1
MOTB	Vehicle motion using Lat/Long/Height with respect to WGS84	5.10.2
AIDING_OFFSET	Sets the aiding offset for a given vehicle	5.10.3

Table 5-11 Hardware and Calibration commands

Command	Description	Section
Hardware and Calibration commands		5.12
RFOF	Returns the serial number and rear connector offset setting.	5.12.1
CAL	Turns on calibration mode.	5.12.2
CAL_LEVEL	Adjusts the output level at the front panel.	5.12.3
NBLK2	Specifies an alternative hardware signal control block	5.12.4
PCAL	Performs self-calibration	5.12.5
FW_CMD	Use to pass commands to firmware	5.12.6
*IDN?	Gets ASCII equipment identification	5.12.7
LOCAL_LOCKOUT	Disables SimGEN user interface	5.12.8

Table 5-12 Vehicle Data Request commands

Command	Description	Section
Vehicle Data Request commands		5.13.1
VEH_X_POS	Returns the X position of the vehicle	
VEH_Y_POS	Returns the Y position of the vehicle	
VEH_Z_POS	Returns the Z position of the vehicle	
VEH_X_VEL	Returns the X velocity of the vehicle	
VEH_Y_VEL	Returns the Y velocity of the vehicle	
VEH_Z_VEL	Returns the Z velocity of the vehicle	
VEH_N_VEL	Returns the north velocity of the vehicle	
VEH_E_VEL	Returns the east velocity of the vehicle	
VEH_D_VEL	Returns the down velocity of the vehicle	
VEH_X_ANG_VEL	Returns the x angle velocity of the vehicle	
VEH_Y_ANG_VEL	Returns the y angle velocity of the vehicle	
VEH_Z_ANG_VEL	Returns the z angle velocity of the vehicle	
VEH_X_ACC	Returns the x acceleration of the vehicle	
VEH_Y_ACC	Returns the y acceleration of the vehicle	
VEH_Z_ACC	Returns the z acceleration of the vehicle	
VEH_X_JERK	Returns the x jerk of the vehicle	
VEH_Y_JERK	Returns the y jerk of the vehicle	
VEH_Z_JERK	Returns the z jerk of the vehicle	
VEH_LAT	Returns the latitude of the vehicle	
VEH_LONG	Returns the longitude of the vehicle	
VEH_HEIGHT	Returns the height of the vehicle (GPS Ellipsoid)	
VEH_HEADING	Returns the heading of the vehicle	
VEH_ELEVATION	Returns the elevation of the vehicle	
VEH_BANK	Returns the bank of the vehicle	
VEH_MSL_HEIGHT	Returns the meters above sea level height of the vehicle	
VEH_MSL_OFFSET	Returns the offset between MSL_HEIGHT and HEIGHT	
VEH_SPEED	Returns the speed of the vehicle	
VEH_ALL	Returns all of the above in a list	
VEH_GROUND_SPEED	Return the ground speed of the vehicle	

Table 5-13 Antenna Data Request commands

Command	Description	Section
Antenna Data Request commands		5.13.2
ANT_X_POS	Returns the x position of the antenna	
ANT_Y_POS	Returns the y position of the antenna	
ANT_Z_POS	Returns the z position of the antenna	
ANT_X_VEL	Returns the x velocity of the antenna	
ANT_Y_VEL	Returns the y velocity of the antenna	
ANT_Z_VEL	Returns the z velocity of the antenna	
ANT_X_ACC	Returns the x acceleration of the antenna	
ANT_Y_ACC	Returns the y acceleration of the antenna	
ANT_Z_ACC	Returns the z acceleration of the antenna	
ANT_DOP	Returns the DOP of the antenna	
DOP_TYPE	Returns the DOP type of the antenna	
ANT_LAT	Returns the latitude of the antenna	
ANT_LONG	Returns the longitude of the antenna	
ANT_HEIGHT	Returns the height of the antenna	
ANT_ALL	Returns all position information	
ANT_GROUND_SPEED	Returns the ground speed of the antenna	
ANT_TERR_OBS_ANGLE	Returns the angle below which satellites will be obscured (rad)	
ANT_TERR_OBS_DISTANCE	Returns the distance from the antenna to the obscuring obstacle	
ANT_TERR_OBS_HEIGHT	Returns the height used to calculate the obscuration angle (either height above antenna if height is relative to vehicle's height is selected, otherwise it is just obstacle height)	

Table 5-14 Signal Data Request commands

Command	Description	Section
Signal Data Request commands		5.13.3
SIG_X_POSN	Returns the x position of the satellite when signal sent	
SIG_Y_POSN	Returns the y position of the satellite when signal sent	
SIG_Z_POSN	Returns the z position of the satellite when signal sent	
SIG_PRANGE	Returns the pseudo range of the signal (at time T the distance between the antenna and the satellite at time T-1)	
SIG_TXID	Returns the SVID of the signals transmitter	
SIG_ELEV	Returns the elevation of the signal	
SIG_AZIM	Returns the azimuth of the signal	
SIG_RANGE	Returns the range of the signal (at time T the distance between the satellite and the antenna)	
SIG_LEVEL	Returns the signal level	
SIG_TROPO_DELAY	Returns the troposphere delay of the signal	
SIG_IONO_DELAY	Returns the ionosphere delay of the signal	
SIG_PR_RATE	Returns the pseudorange rate of the signal	
SIG_PRR_RATE	Returns the pseudorange and pseudorange rate	
SIG_PRANGE_ERROR	Returns the pseudorange error	
SIG_MPI	Returns the multipath index of a satellite (note this command should only be directed to a Channel)	
SIG_ALL	Returns all	
SIG_MP_TYPE	Returns multipath settings	

Table 5-15 Transmitter Data Request commands

Command	Description	Section
Transmitter Data Request commands		5.13.4
TX_ECEF_X	Returns the x position of the transmitter	
TX_ECEF_Y	Returns the y position of the transmitter	
TX_ECEF_Z	Returns the z position of the transmitter	
TX_ECEF_X_VEL	Returns the velocity of the transmitter in the x axes	
TX_ECEF_Y_VEL	Returns the velocity of the transmitter in the y axes	
TX_ECEF_Z_VEL	Returns the velocity of the transmitter in the z axes	
TX_LAT	Returns the latitude of the transmitter	
TX_LONG	Returns the longitude of the transmitter	
TX_HEIGHT	Returns the height of the transmitter	
TX_POS	Returns the ECEF for X,Y,Z	
TX_SVID	Returns the SVID of the transmitter	
TX_SIG_ID	Returns the signal id of the transmitter, such as GPS or SBAS	

Table 5-16 Navigation commands

Command	Description	Section
Navigation commands		5.14
GPS_NAV_DATA_ERR	Remote input and insertion into action queue of GPS nav data errors already scheduled in current scenario	5.14.2
GPS_NAV_DATA_MOD	Remote input and insertion into action queue of GPS nav data modifications already scheduled in current scenario	5.14.3
WAGE	Enables / disables WAGE for the whole constellation.	5.14.4

Table 5-17 Navigation Data related commands

Command	Description	Section
Navigation Data related commands		5.15
QZ_SYMB	Sends Quasi-Zenith navigation data	5.15.2
GPS_LEGACY_NAV	Sends GPS legacy navigation data	5.15.3

Table 5-18 Interferer commands

Command	Description	Section
Interferer commands		5.16
COHERENT_CW	Generates a coherent CW signal	5.16.2
NON_COHERENT_CW	Generates a non-coherent CW signal	5.16.3
SWEPT_CW	Generates a swept CW signal	5.16.4
AM	Generates an AM signal	5.16.5
FM	Generates an FM signal	5.16.6
NOISE	Generates white Gaussian noise	5.16.7
PULSE - basic	Overlays the current modulation with a pulsing signal defined using period and width	5.16.8
PULSE - using prf and duty cycle	Overlays the current modulation with a pulsing signal defined using prf and duty cycle	5.16.9
RF_ON_OFF	Specifies when the interference source is on	5.16.10
DELTA_LEVEL	Specifies an increment in signal level	5.16.11
LEVEL	Changes the absolute signal level	5.16.12
FIXED_MODE	Fixes current signal at a constant level	5.16.13
MODELLED_MODE	Fixes geographical position of interferer	5.16.14
TX_ANTENNA_PATTERN	Applies a transmit antenna pattern	5.16.15

5.1.1 Timestamp

Note:

1) SimREMOTE supports this timestamp syntax: *ssss.ss,<command>...*, the time directly defined in floating point seconds. For example “01:02:03.4,MOT...” is equivalent to “3723.4,MOT...”

2) SimREMOTE will not accept or process further commands until the current command is actioned. Specifying a command with a timestamp in the future will block recognition of further commands until that time

Description	<p>The timestamp field prefixes all timed commands. The timestamp specifies the time into run to action the command.</p> <p>Use the literal '-' instead of a time format to action the command as soon as it is received.</p>
Format	<pre>timestamp ::= <timeFormat> <actionImmediately></pre>
Where	<pre>timeFormat ::= [<d>Δ]<hh>:<mm>:<ss>[.<ms>]</pre> <p>Where:</p> <pre>d ::= 0-23 (defaults to 0 if not used) hh ::= 0-23 mm ::= 0-59 ss ::= 0-59 ms ::= 0-999 (in 1 millisecond steps, defaults to 0 if not used.)</pre> <pre>actionImmediately ::= "-"</pre>
Example(s)	<pre>-,VEH_X_POS,v1 1 12:05:10.100,VEH_X_POS,v1 12:05:10,VEH_X_POS,v1</pre>

5.1.2 Returned response format

Description	Every command will cause a response to be sent. All responses to a remote command take the form of an XML data set.
Format	<pre> response ::= "<msg>" "<status>" <statValue> "<\status>" ["<data>"<dataValue>"<\data>"] ["<error>"<errorValue>"<\error>"] ["<fatal>"<fatalValue>"<\fatalValue>"] "<\msg>" </pre>
Where	<pre> statValue ::= (hex number, see status definitions in section 4.1.2) dataValue ::= <number> <string>{,<number> <string>} errorValue ::= <string> fatalValue ::= <string> number ::= (any number value expressed in scientific format) string ::= (alphanumeric characters) </pre>
Returns	<pre> <msg> <status> 2 <\status> <data> 8 <\data> <\msg> <msg> <status> 2 <\status> <error> Channel number is out of range <\error> <\msg> </pre>

5.2 Scenario commands

Command	Description
Scenario commands	
SET_VEHICLE_DIAL_TYPE	Sets the type of vehicle dial SimGEN uses.
SET_VEHICLE_DIAL_UNITS	Sets the units of the vehicle speedometer(s)
SC	Select a scenario
SC_NAME	Returns currently selected scenario name
SC_DURATION	Returns currently selected scenario duration
INIT_POS	Set a vehicles initial position
TR	Set trigger mode
RU	Run a scenario
RU_NOWAIT	Runs scenario without waiting for ARMED state.
EN	End a running scenario
RW	Rewind an ended scenario
AR	Arms the scenario
AR_NOWAIT	Makes remote control available during arming
NULL	Returns the status of the scenario
LOAD_ALMANAC	Loads a YUMA format almanac file
WRITE_ALMANAC	Writes a YUMA format almanac file
LOAD_RINEX	Loads a RINEX format file
WRITE_RINEX	Writes a RINEX format file
LOAD_ACT	Loads a user action file
LOAD_UCD	Loads a user command file
REGISTER_EXTERNAL_ARM	Registers an external arm source.
DEREGISTER_EXTERNAL_ARM	De-registers a registered external arm source
EXTERNAL_ARMED	Tells SimGEN an external source has armed
TURBO	Enables Turbo mode
*IDN?	Return the version of SimGEN
GET_INS_FILE	Returns the XML file of each vehicle number
MESSAGE	Shows a message in System Messages
LOAD_UMT	Loads a user motion file
LOAD_CONSTELLATION	Loads a constellation file
LOAD_ITF	Loads an interferer file
SIMULATION_ITERATION_RATE	Sets the simulation iteration rate
GET_6700_HW	Queries capability of GSS6700 signal generators

5.2.1 SET_VEHICLE_DIAL_TYPE

Description	Sets the type of vehicle dial displayed in the SimGEN main window. Must be sent after the scenario has either been loaded or placed in the user command file
Format	SET_VEHICLE_DIAL_TYPE,<veh_ant>,<type>,<max_velocity>
Where	veh_ant ::= <vehicle_id>" "<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 type ::= "AIR" "LAND" "SEA" "SPACE" max_velocity ::= maximum velocity to be shown on vehicle dial, in m/s (with conversion for the dial units set in SET_VEHICLE_DIAL_UNITS)
Returns	status see returned response format, section 5.1.2
Example	Set type of vehicle dial to Land, with a maximum velocity of 15 m.s ⁻¹ SET_VEHICLE_DIAL_TYPE,v1_a1,LAND,15

5.2.2 SET_VEHICLE_DIAL_UNITS

Description	Sets the type of units displayed in SET_VEHICLE_DIAL_TYPE
Format	SET_VEHICLE_DIAL_UNITS,<veh_ant>,<units_id>,<dial_id>
Where	veh_ant ::= <vehicle_id>" "<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 units_id ::= "m/s" "kph" "knots" "ft/s" "ft/min" "mph" dial_id ::= "spd1" (left-hand speedometer) "spd2" (right-hand speedometer) (where there are two speedometers)
Returns	status see returned response format, section 5.1.2
Example	Set the left-hand speedometer for vehicle 1, antenna 1 to read kph SET_VEHICLE_DIAL_UNITS,v1_a1,kph,spd1

5.2.3 SC

Description	<p>Loads a scenario file (*.scn) from the given path.</p> <p>If a filename is not given, the command looks in the current directory for a scenario file.</p> <p>To correctly specify a scenario, you must give the absolute path to the scenario file.</p> <p>Must be sent before a scenario has been run.</p>
Format	SC[,<file_name_and_path>]
Where	File_name_and_path ::= (absolute path and file name)
Returns	see returned response format, section 5.1.2
Example	<p>Load the <i>Aircraft_Example.scn</i> scenario from the Aircraft_example scenario folder:</p> <pre>SC,C:\Program Files\Spirent Communications\SimGEN \Scenarios\Aircraft_Example\Aircraft_Example.scn</pre>

5.2.4 SC_NAME

Description	Returns the name of the currently loaded scenario.
Format	SC_NAME
Returns	Scenario filename see returned response format, section 5.1.2

5.2.5 SC_DURATION

Description	Returns the duration of the currently loaded scenario.
Format	SC_DURATION
Returns	Scenario duration see returned response format, section 5.1.2

5.2.6 INIT_POS

Description	<p>Overrides the current simulation initial position (static scenarios only).</p> <p>Only valid for static and remote vehicles (but note that remote vehicles can be moving).</p> <p>Not a valid command for vehicle models with motion.</p> <p>Must be sent before running a scenario, but after an SC command. You must always set the timestamp of this command to "-" or 0Δ00:00:00.</p>
Format	<code><timestamp>, INIT_POS, <veh_mot>, <latitude>, <longitude>, <height></code>
Where	<pre>timestamp ::= see section 5.1.1 veh_mot ::= <vehicle_id>"_"<motion_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 motion_id ::= "m"<m_number> m_number ::= "1" (motion model number always 1) latitude ::= latitude, degrees, +ve = North longitude ::= longitude, degrees, +ve = East height ::= height, metres</pre>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Now, set the initial position of vehicle 1, motion model number 1, to be 10.0° N, 20.0° E and 30.0 m high:</p> <pre>-, INIT_POS, v1_m1, 10.0, 20.0, 30.0</pre> <p>At the start of the scenario, set the initial position of vehicle 2, motion model number 1, to be 30.0° S, 20.0° E and 10.0 m high:</p> <pre>0 00:00:00, INIT_POS, v2_m1, -30.0, 20.0, 10.0</pre>

5.2.7 TR

Description	<p>Sets the trigger mode.</p> <p>You must send this command BEFORE an RU command</p>
Format	<code>TR, <mode></code>
Where	<pre>mode ::= "0" (software trigger) "1" (exit trigger immediately) "2" (exit trigger on edge of next 1PPS)</pre>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Set software trigger mode:</p> <pre>TR, 0</pre> <p>Exit trigger mode on edge of next 1PPS:</p> <pre>TR, 2</pre>

5.2.8 RU

Description	<p>Run the currently selected scenario (requires an external pulse to start in Immediate or Delayed trigger modes).</p> <p>In the ARMED state and with TR,0 (software trigger), the scenario runs on the next 1PPS.</p> <p>If an AR (arm) command has not been received, the RU command performs an AR (arm) command first,</p> <p>Note: <i>In this case the time to start is delayed by an indeterminate number of seconds.</i></p>
Format	RU
Returns	status see returned response format, section 5.1.2
Example(s)	RU

5.2.9 RU_NOWAIT

Description	<p>Arms and runs the currently selected scenario (requires an external pulse to start in Immediate or Delayed trigger modes).</p> <p>The RU command waits until the ARMED state is achieved before returning with a response.</p> <p>The RU_NOWAIT command returns immediately and does not wait for the ARMED state.</p> <p>Use RU_NOWAIT when an external armed source is used. See Figure 4-7 for more details.</p>
Format	RU_NOWAIT
Returns	status see returned response format, section 5.1.2
Example(s)	RU_NOWAIT

5.2.10 EN

Description	End the currently running scenario.
Format	<code><timestamp>,EN[,<ending_type>[,<log_option>]]</code>
Where	<p><code>timestamp</code> ::= see general notes on commands</p> <p><code>ending_type</code> ::= "0" (Stop scenario and leave at ENDED state. Default) "1" (Stop scenario and rewind to INITIALISED state. Ready to run again) "2" (Stop scenario and rewind to INITIALISED state. Rewind remote command file and repeat command sequence in it. Only applies to remote commands from file)</p> <p><code>log_option</code> ::= "0" (If logging is enabled do not produce log files) "1" (If logging is enabled produce log files. Default)</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>End current scenario now: <code>-,EN</code></p> <p>End current scenario at 24 hours into run: <code>1 00:00:00,EN:</code></p> <p>End current scenario now. Rewind scenario: <code>-,EN,1</code></p> <p>End current scenario now. Do not rewind, do not save log file: <code>-,EN,0,0</code></p>

5.2.11 RW

Description	Rewinds a scenario, ready to run again, after using the EN command with <code>ending_type</code> set to 0.
Format	<code>RW</code>
Returns	status see returned response format, section 5.1.2
Example(s)	<code>RW</code>

5.2.12 AR

Description	<p>Notes:</p> <ol style="list-style-type: none"> 1) Send this command before an RU command. 2) SimGEN must enter the armed state before you can send the RU command. 3) If you use other commands during arming or use remote communication, read the details below - you may need to use AR_NOWAIT, section 5.2.13. In all other cases, use the AR command. <p>The AR command blocks other commands and will not return a result until the arming phase is complete. It also blocks remote communication with SimGEN during the arming phase. This means external arming sources that need to get data from SimGEN at the arming time; or communicate to SimGEN they are armed, cannot do so because the AR command blocks communication with SimGEN.</p> <p>Arms the current scenario.</p>
Format	AR
Returns	Status see returned response format, section 5.1.2
Example(s)	AR

5.2.13 AR_NOWAIT

Description	<p>Notes:</p> <ol style="list-style-type: none"> 1) Send this command before an RU or RU_NOWAIT command. 2) SimGEN must enter the armed state before you can send the RU or RU_NOWAIT commands. 3) If you use other commands during arming or use remote communication, use AR_NOWAIT. In all other cases, use the AR command, section 5.2.12. <p>The AR_NOWAIT command returns immediately to the user of the command and then starts to arm the scenario. It does not block commands or remote communication, like the AR command.</p> <p>To determine when SimGEN reaches the armed state, send NULL commands, section 5.2.14; or monitor data-streaming, section 5.8.</p> <p>When you have confirmed SimGEN is armed, you can use the RU command, section 5.2.8 or the RU_NOWAIT command, section 5.2.9 to run the scenario.</p>
Format	AR_NOWAIT
Returns	status see returned response format, section 5.1.2
Example(s)	AR_NOWAIT

5.2.14 NULL

Description	Returns status information (no other function)
Format	NULL
Returns	status see returned response format, section 5.1.2
Example(s)	NULL

5.2.15 LOAD_ALMANAC

Description	Loads SimGEN with a YUMA format almanac file you specify.
Format	LOAD_ALMANAC,<file_name_and_path>,<rollover_week_number>,<network>
Where	<p>file_name_and_path ::= the full path to the YUMA Almanac file</p> <p>rollover_week_number ::= [0 - 5]</p> <p>[network ::= "GPS" "GLONASS" "Galileo" (If omitted, default network is GPS)]</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Load the Almanac file <i>yuma1.txt</i> in the Aircraft_Example scenario folder, using rollover week number 1 , for network GPS</p> <pre>LOAD_ALMANAC,C:\Program Files\Spirent Communications\ SimGEN\Scenarios\Aircraft_Example\yumal.txt,1,GPS</pre>

5.2.16 WRITE_ALMANAC

Description	Writes a YUMA format almanac file to a location you specify.
Format	WRITE_ALMANAC,<file_name_and_path>,<network>
Where	<p>file_name_and_path ::= the full path where you want to write the Almanac file.</p> <p>[network ::= "GPS" "GLONASS" "Galileo" (If omitted, default network is GPS)]</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Write the YUMA format almanac file <i>yuma1.txt</i> in the Aircraft_Example scenario folder, for network GPS</p> <pre>WRITE_ALMANAC,C:\Program Files\Spirent Communications\ SimGEN\Scenarios\Aircraft_Example\yumal.txt,GPS</pre>

5.2.17 LOAD_RINEX

Description	Loads SimGEN with a RINEX format file you specify.
Format	LOAD_RINEX,<file_name_and_path>,<network>
Where	<p>file_name_and_path ::= the full path to the RINEX file.</p> <p>[network ::= "GPS" "Galileo" (If omitted default network is GPS)]</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Load the RINEX format file <i>rinex1.txt</i> in the Aircraft_Example scenario folder for the Galileo network</p> <pre>LOAD_RINEX,C:\Program Files\Spirent Communications\ SimGEN\Scenarios\Aircraft_Example\rinex1.txt,GALILEO</pre>

5.2.18 WRITE_RINEX

Description	Writes a RINEX format file to a location you specify.
Format	WRITE_RINEX,<file_name_and_path>,<network>,<version>
Where	<p>file_name_and_path ::= the full path where you want to write the RINEX file.</p> <p>[network ::= "GPS" "Galileo" (If omitted default network is GPS)]</p> <p>[version ::= "2.10" "3.00" SimGEN supports both RINEX v2.10 and v3.00. For Galileo, you must specify v3.00. RINEX v2.10 does not support GALILEO. (If omitted, default version is 2.10)]</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Write the RINEX format file <i>rinex1.txt</i> in the Aircraft_Example scenario folder for the GPS network using RINEX v2.10</p> <pre>WRITE_RINEX,C:\Program Files\Spirent Communications\ SimGEN\Scenarios\Aircraft_Example\rinex1.txt,GPS,2.10</pre>

5.2.19 LOAD_ACT

Description	<p>Loads a user actions (*.act) file into the SimGEN scenario.</p> <p>If a path is not given, the command looks in the current directory for a user actions file.</p> <p>Note: <i>Ignored if scenario is running</i></p>
Format	LOAD_ACT,<file_name_and_path>
Where	file_name_and_path ::= absolute path and filename
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Load the user actions file <i>user_actions.act</i> from the current scenario folder:</p> <pre>LOAD_ACT,user_actions.act</pre> <p>Load the user actions file <i>user_actions.act</i> located in the <i>Aircraft_Example</i> folder into the SimGEN scenario options.</p> <pre>LOAD_ACT,C:\Program Files\Spirent Communications \SimGEN\Scenarios\Aircraft_Example\user_actions.act</pre>

5.2.20 LOAD_UCD

Description	<p>Loads a user command (*.ucd) file into the SimGEN scenario.</p> <p>If a path is not given, the command looks in the current directory for a user command file.</p> <p>Note: <i>Ignored if scenario is running</i></p>
Format	LOAD_UCD,<file_name_and_path>
Where	file_name_and_path ::= absolute path and filename
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Load the user command file <i>user_commands.ucd</i> from the current scenario folder:</p> <pre>LOAD_UCD,user_commands.ucd</pre> <p>Load the user command file <i>user_command.ucd</i> located in the <i>Aircraft_Example</i> folder into the SimGEN scenario options.</p> <pre>LOAD_UCD,C:\Program Files\Spirent Communications \SimGEN\Scenarios\Aircraft_Example\user_command.ucd</pre>

5.2.21 REGISTER_EXTERNAL_ARM

Description	<p>Registers an external arm source.</p> <p>Allows an external source to prevent SimGEN entering the ARMED state until the external source has informed SimGEN it has armed using the EXTERNAL_ARMED command.</p> <p>See Figure 4-7 for more details of the arming sequence.</p>
Format	REGISTER_EXTERNAL_ARM
Returns	status see returned response format, section 5.1.2
Example(s)	REGISTER_EXTERNAL_ARM

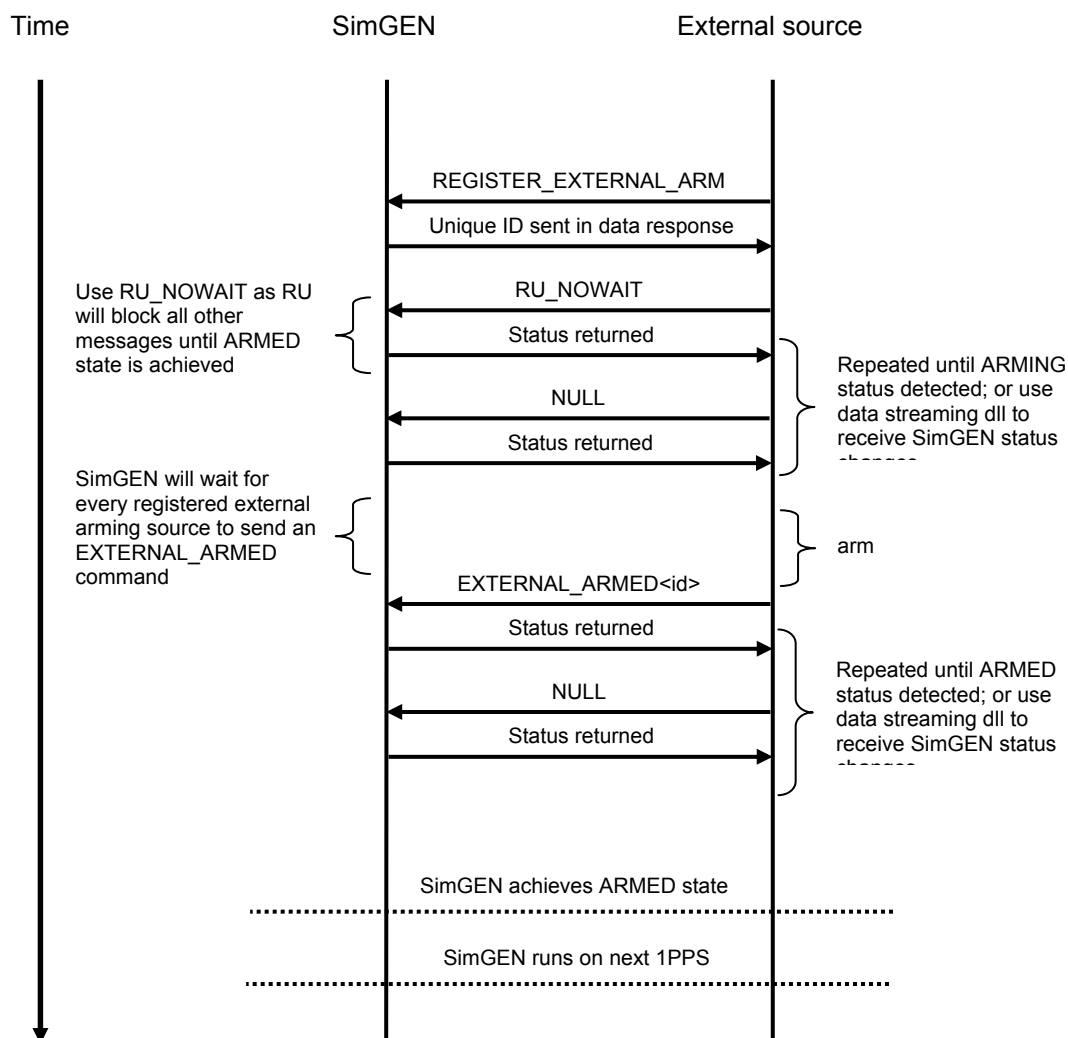
5.2.22 DEREGISTER_EXTERNAL_ARM

Description	<p>De-registers an external arm source</p> <p>Informs SimGEN the external armed source no longer requires SimGEN to wait before going into the ARMED state. See Figure 4-7 for more details.</p>
Format	DEREGISTER_EXTERNAL_ARM,<id>
Where	id ::= unique id returned using the REGISTER_EXTERNAL_ARM command
Returns	status see returned response format, section 5.1.2
Example(s)	DEREGISTER_EXTERNAL_ARM,1

5.2.23 EXTERNAL_ARMED

Description	<p>Notifies SimGEN the external source has armed.</p> <p>Notifies SimGEN on starting to run it cannot reach the ARMED state until the EXTERNAL_ARMED command has been received with the correct id.</p> <p>Once all registered external armed sources have armed and informed SimGEN, SimGEN moves into the ARMED state just after the next 1PPS.</p> <p>SimGEN runs on the next 1PPS after achieving the ARMED state.</p> <p>See Figure 4-7 for more details.</p>
Format	EXTERNAL_ARMED,<id>
Where	id ::= unique id returned using the REGISTER_EXTERNAL_ARM command
Returns	status see returned response format, section 5.1.2
Example(s)	EXTERNAL_ARMED,1

Figure 5-1: External arm source message sequence



5.2.24 TURBO

Description	<p>Enables Turbo mode. Turbo mode can only be used when you have selected Options-General Options-No Hardware.</p> <p>Note: The actual Turbo speed depends on your scenario and the available processing power of your PC. When the Turbo slider is set to maximum (fully to the right) the Turbo speed is as fast as your system can achieve. In Turbo mode, the System Messages window displays Current Rate (elapsed minutes in the scenario) for the speed (Turbo slider position) every minute of real time.</p>
Format	TURBO,<state>,<speed>
Where	<p>state := "0" (off) "1" (on)</p> <p>speed ::= 0 - 100 as a percentage of the Turbo slider.</p> <p>Where: 0 is fully to the left 100 is fully to the right.</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Set Turbo mode on, using speed 50 (slider at half-way setting):</p> <p>TURBO,1,50</p>

5.2.25 *IDN?

Description	Returns the identity of SimGEN software. Also see section 5.12.7 (these two *IDN? commands are NOT identical).
Format	*IDN?
Returns	status see returned response format, section 5.1.2 and message data is: <company>,<executable>,0,<version>
Example	<data> Spirent, SimGEN for Windows,0,V1.0 </data>

5.2.26 GET_INS_FILE

Description	Return the XML data of the inertial sensor settings of a vehicle
Format	<timestamp>,<GET_INS_FILE>,<vehicle_id>
Where	<p>timestamp ::= see section 5.1.1</p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p>
Returns	Status see returned response format, section 5.1.2
Example(s)	<p>Now, get the inertial sensor settings of vehicle 1:</p> <p>-,GET_INS_FILE,v1</p>

5.2.27 MESSAGE

Description	Sends a message to the SimGEN message file from a remote source
Format	MESSAGE,<remote_name>,<level>,<message>
Where	<pre>remote_name ::= "any string" level ::= "Hardware" "Debug" "Bite_info" "Info" "Warning" "Error" "Bite_error" "Fatal" message ::= "any string"</pre>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Send the message "This is a Test Message" from remote name "xyz" at the Debug level:</p> <pre>MESSAGE,xyz,Debug,This is a Test Message</pre>

5.2.28 LOAD_UMT

Description	<p>Loads a user motion (*.umt) file. If a path is not given, the command looks in the current directory for a user motion file.</p> <p>Note: Ignored if scenario is running</p>
Format	LOAD_UMT,<vehicle_id>,<file_name_and_path>
Where	<pre>vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 file_name_and_path ::= absolute path and filename</pre>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>For vehicle 1, load the Aircraft_Example.umt file from the Aircraft_example scenario folder:</p> <pre>LOAD_UMT,v1,C:\Program Files\Spirent Communications \SimGEN\Scenarios\Aircraft_Example\Aircraft_Example.umt</pre>

5.2.29 LOAD_CONSTELLATION

Description	<p>Loads a constellation file. The extension of the constellation file must match the constellation you select. If a path is not given, the command looks in the current directory for a constellation file.</p> <p>Note: Ignored if scenario is running</p>
Format	LOAD_CONSTELLATION,<constellation>,< file_name_and_path>
Where	<pre>constellation ::= GPS "WAAS" "EGNOS" "MSAS" "GLONASS" "GALILEO" file_name_and_path ::= absolute path and filename</pre>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Load the GPS constellation file <i>foo.gps</i> from the current scenario folder:</p> <pre>LOAD_CONSTELLATION,GPS,foo.gps</pre> <p>Load the Galileo constellation file <i>foo.gleo</i> from the Aircraft_Example scenario folder:</p> <pre>LOAD_CONSTELLATION,GALILEO,C:\Program Files\Spirent Communications\SimGEN\Scenarios\Aircraft_Example\foo.gleo</pre>

5.2.30 LOAD_ITF

Description	<p>Loads an interference file.</p> <p>If a path is not given, the command looks in the current directory for an interference file.</p> <p>Note: <i>Ignored if scenario is running</i></p>
Format	LOAD_ITF,<veh_ant>,<file_name_and_path>
Where	<p>veh_ant ::= <vehicle_id>" "<antenna_id></p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p>antenna_id ::= "a"<a_number></p> <p>a_number ::= antenna number starting from 1</p> <p>file_name_and_path ::= absolute path and filename</p>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Load the vehicle 1, antenna 1 interference file <i>foo.itf</i> from the current scenario folder:</p> <pre>LOAD_ITF,v1_a1,foo.itf</pre> <p>Load the vehicle 1, antenna 1 interference file <i>foo.itf</i> from the Aircraft_Example scenario folder:</p> <pre>LOAD_ITF,v1_a1,C:\Program Files\Spirent Communications\SimGEN\Scenarios\Aircraft_Example\foo.itf</pre>

5.2.31 SIMULATION_ITERATION_RATE

Description	<p>Sets the simulation iteration rate (milliseconds) a scenario will use.</p> <p>As the command always returns the simulation iteration rate, you can query the simulation iteration rate a scenario will use by excluding the <rate> parameter.</p> <p>If the rate you enter is out of range, SimGEN returns an error and does not change the simulation iteration rate</p> <p>Note: <i>You can only send this command when SimGEN is in the "ready state", that is, with a scenario loaded but not running.</i></p>
Format	SIMULATION_ITERATION_RATE[,<rate>]
Where	<p>rate ::= "4" "10" "100"</p> <p>(rates in milliseconds)</p> <p>Note: <i><rate> is set by your simulator and you may not be able to use each rate available from this command. If you set a rate your simulator cannot use, SimGEN returns an error and does not change the rate.</i></p>
Returns	status see returned response format, section 5.1.2

Examples

Send a simulation iteration rate query to the current scenario (that is using a 100 ms iteration rate):

```
SIMULATION_ITERATION_RATE
```

Response:

```
<msg>
  <status> 2 </status>
  <data> 100 </data>
</msg>
```

Set the current scenario to use 10 ms simulation iteration rate:

```
SIMULATION_ITERATION_RATE,10
```

Response:

```
<msg>
  <status> 2 </status>
  <data> 10 </data>
</msg>
```

5.2.32 GET_6700_HW

Note: Use this command only with GSS6700 signal generators

Description	Queries capability of all GSS6700 signal generators connected to the SimGEN controller Note: If a GSS6700 signal generator is not connected to the SimGEN controller or you have selected Control Options-No hardware ("dummy run") the response is: "Failed to read 6700 hardware information."
Format	GET_6700_HW
Response	<serial_number>,<capability[:capability..]> [,<serial_number>,<capability..>]
Where	serial_number ::= serial number of signal generator capability ::= constellation
Example	Query all 6700 signal generators connected to the SimGEN controller 01201045,GPS:GLONASS:GALILEO

5.3 Time commands

Command	Description
Time commands	
TIME	Retrieves the current time into run when the command is received.
TIME_STEP	Retrieves the time into run of the current time step
GPS_TIME	Retrieves the number of seconds from GPS time zero.
UTC_TIME	Retrieves the scenario time as an ASCII string.
STTIME	Retrieves the start time of the current scenario
START_TIME	Sets the start time and duration of the current scenario
UTC_OFFSET	Retrieves GPS to UTC offset
UTC_OFFSET_ACC	Set the UTC offset acceleration
GPS_UTC_OFFSET	Retrieves GPS to UTC offset, at a given time
ZCNT_TOW	Retrieves the z-count TOW in seconds
ZCNT_WEEK	Retrieves the z-count week number
ZCNT_WEEK_ROLLOVER	Retrieves the z-count week rollover number
OFFSET_PROC_TIME	Applies a fixed delay to signal control data processing.
TIMER_PULSE	Sends a single pulse from an optional timer card
SET_TIOP	Reads or sets the timer output.

5.3.1 TIME

Description	Retrieves the current time into run when the command is received.
Format	<timestamp>,TIME
Where	timestamp ::= see section 5.1.1
Returns	Status see returned response format, section 5.1.2 Data is time into run in seconds, to nearest ms. Returns zero when in ARMED state
Example(s)	Now, return current time into run: -,TIME

5.3.2 TIME_STEP

Description	Retrieves the time into run of the current time step, in 4, 5, 10 or 100 ms steps (depending on simulation iteration rate)
Format	<timestamp>,TIME_STEP
Where	timestamp ::= see section 5.1.1
Returns	Status see returned response format, section 5.1.2 Data is time step in seconds (floating point number) for example 31.120. Returns zero when in ARMED state
Example(s)	Now, return time into run of current time step: -,TIME_STEP

5.3.3 GPS_TIME

Description	Returns the number of seconds from GPS time zero, (6 th January 1980)
Format	<timestamp>,GPS_TIME
Where	timestamp ::= see section 5.1.1
Returns	status see returned response format, section 5.1.2 data (time in integer seconds)
Example(s)	Now, return number of seconds from GPS time zero: -,GPS_TIME

5.3.4 UTC_TIME

Description	Returns the current scenario time as an ASCII string.
Format	<timestamp>,UTC_TIME
Where	timestamp ::= see section 5.1.1
Returns	status see returned response format, section 5.1.2 data (date and time as an ASCII string, format dd mmm yyyy hh:mm:ss) For example: 01 Jan 2004 00:00:50
Example(s)	Now, return current scenario time: -,UTC_TIME

5.3.5 STTIME

Description	Retrieves the start time of the current scenario
Format	STTIME
Returns	status see returned response format, section 5.1.2 data (data format is <date>Δ<time> see section 5.3.6)
Example(s)	STTIME

5.3.6 START_TIME

Notes:

- 1) Send after an SC command and before running a scenario.
- 2) Scenario start time must start on a 6-second boundary. For example, 12:00:06 or 12:00:18 is OK but 12:00:15 is not.
- 3) Reference a) details sat_locas Maintain satellite locations. Using the sat_loc option will overwrite the signal sources files or every constellation your scenario uses with the new satellite orbital data.

Description	Over-rides current simulation start time, date and duration.
Format	START_TIME,<date>Δ<time>[,<duration>][,<sat_loc>]
Where	<pre> date ::= <dd>"-"<mmm>"-"<yyyy> dd ::= (day of the month) mmm ::= "JAN" "FEB" "MAR" "APR" "MAY" "JUN" "JUL" "AUG" "SEP" "OCT" "NOV" "DEC" (month) yyyy ::= year time ::= <hh>:<mm>:<ss> d ::= 0-23 (number of days) hh ::= 0-23 (hour) mm ::= 0-59 (min) ss ::= 0-59 (seconds) duration ::= [<d>Δ]<hh>:<mm>:<ss> (default is 9 days) sat_loc ::= "0" don't maintain satellite location "1" maintain satellite location </pre>
Returns	status see returned response format, section 5.1.2
Example(s)	<p>Set the scenario to start on 1st January 2003 at 10:00am, scenario to run for 5 days and 2 hours:</p> <pre>START_TIME,01-JAN-2003 10:00:00,5 02:00:00</pre> <p>Set the scenario to start on 12th December 2002 at 1:00pm, scenario to run for 1 hour :</p> <pre>START_TIME,12-DEC-2002 13:00:00,01:00:00</pre>

5.3.7 UTC_OFFSET

Description	Retrieves GPS to UTC offset (as defined in GPS constellation file) in integer seconds
Format	UTC_OFFSET
Returns	<p>status see returned response format, section 5.1.2</p> <p>data offset in integer seconds</p>
Example	UTC_OFFSET

5.3.8 UTC_OFFSET_ACC

Description	Set the UTC_offset_acceleration used by SimGEN (Signal sources-General-A2: UTC offset acceleration:)
Format	UTC_OFFSET_ACC,<network>,<UTC_offset_acceleration>
Where	<pre> network ::= "GPS" UTC_offset_acceleration ::= Value of A2 range -1E-6 to 1E-6 μs/s </pre>
Returns	<p>status see returned response format, section 5.1.2</p> <p>data offset in seconds</p>
Example	<p>Set GPS UTC_Offset_Acceleration parameter of 1E-6 μs per s:</p> <pre>UTC_OFFSET_ACC,GPS,1E-6</pre>

5.3.9 GPS.UTC_OFFSET

Description	Retrieves GPS to UTC offset, at a given time into the run, in decimal seconds
Format	<timestamp>,GPS.UTC_OFFSET
Where	timestamp ::= see section 5.1.1
Returns	status see returned response format, section 5.1.2 data offset in decimal, double precision, seconds
Example	Now, get GPS to UTC offset: -,GPS.UTC_OFFSET

5.3.10 ZCNT_TOW

Description	Retrieves the GPS Time Of Week in seconds
Format	<timestamp>,ZCNT_TOW
Where	timestamp ::= see section 5.1.1
Returns	status see returned response format, section 5.1.2 data current GPS TOW in integer seconds
Example	Now, get GPS Time Of Week: -,ZCNT_TOW

5.3.11 ZCNT_WEEK

Description	Retrieves the GPS week number
Format	<timestamp>,ZCNT_WEEK
Where	timestamp ::= see section 5.1.1
Returns	Status see returned response format, section 5.1.2 data Current GPS TOW week in integer seconds
Example	Now, get the GPS week number: -,ZCNT_WEEK

5.3.12 ZCNT_WEEK_ROLLOVER

Description	Retrieves the number of times GPS week number has rolled over.
Format	<timestamp>,ZCNT_WEEK_ROLLOVER
Where	timestamp ::= see section 5.1.1
Returns	Status see returned response format, section 5.1.2 data number of rollovers
Example	Now, get the number of times the GPS week number has rolled over: -,ZCNT_WEEK_ROLLOVER

5.3.13 OFFSET_PROC_TIME

Note:

Output debug messages are generated every minute for:

1) Comparison of the motion-data timestamp against the time of message receipt (true run-time). Outputs mean/max/min over the 1 minute period, (changes to 10 minute period after 30 minutes). If negative the data has been received early.

2) Comparison of the timestamp for available motion data against the application time for signal generation control data (motion data latency).

Outputs mean/max/min over the 1 minute period.

The signal generation data controls the RF signals over the hardware period T to $(T+\Delta T)$ where $\Delta T = 10$ or 100 ms.

In this context, a latency of 0 means that motion data was available at time T but not at $(T+\Delta T)$. In these circumstances the motion over the update interval is propagated from that at T . If this is negative, the motion data is available in advance; if positive the motion data must be propagated from an earlier time than T .

Description	<p>Applies a fixed delay to signal control data processing to minimise latency. See section 7.2.</p> <p>Not applicable with 4 ms iteration rates.</p> <p>Send before the start of the scenario</p> <p>Resets at the end of each run</p>
Format	OFFSET_PROC_TIME,<n>
Where	<p>n ::= number of milliseconds delay you want to apply. Typically, 5 or 6 ms for a 10 ms time step. Depends on the complexity of the simulation system.</p> <p>Range ::= 0 to timestep value (simulation iteration rate)</p> <p>A check on data consistency is performed for each received motion message. This tests the position and velocity of the latest message with that predicted by the previous message. The maximum error in position and velocity (RSS values) are output every minute. If these exceed a threshold of 0.1 m or 1 m.s⁻¹ the message level is upgraded to a warning.</p>
Example	<p>Now apply a fixed data processing delay of 6 ms:</p> <p>OFFSET_PROC_TIME, 6</p>

5.3.14 TIMER_PULSE

Notes:

- 1) The pulse from the optional timer card is positive going and has a width of around 50 μ s.
- 2) The pulse output time occurs approximately 3 to 5 ms late.
- 3) A pulse at time $t = 0$ occurs during the scenario load / arming phase.
- 4) If you use a simulation iteration rate of 100 ms, a *TIMER_PULSE* command at 40 ms occurs at 100 ms (approximately); however, if you are using a simulation iteration rate of 10 ms the command occurs at 40 ms.
- 5) If you have set *TIMER_PULSE* to out put pulses at 40, 50, 60, 70 ms and you are using a simulation iteration rate of 100 ms, the pulses occur after $t = 100$ ms, as close together as the software will allow.

Description	Sends a single pulse from an optional hardware timer card
Format	<timestamp>,TIMER_PULSE
Where	timestamp ::= see section 5.1.1
Returns	Status see returned response format, section 5.1.2 If the optional timer card is not fitted, the return is "Timer pulse not sent"
Example	10:30:45,TIMER_PULSE

5.3.15 SET_TIOP

Notes:

- 1) You must send the *SET_TIOP* command before you send the *ARM* command. *SET_TIOP* is actioned as part of the *ARMING* sequence.
- 2) Currently for GSS8000, 7xxx and 5060 / 6560 signal generators only.

Description	Reads or sets the timer output.
Format	SET_TIOP,<ser_no>,<timer>,<prefix>,<mode>,<frequency>
Where	<p>ser_no ::= "0" (for master box or only one signal generator) serial number of auxiliary signal generator that you want to SET_TIOP</p> <p>7xxx signal generator, select one of three timer outputs: timer ::= "1" "2" "3"</p> <p>5060/6560 signal generator: timer ::= "1"</p> <p>prefix ::= "GATED" "NOTGATED"</p> <p>mode ::= "1PPS" "UPDATE" "10PPS" "HIGH" "DISABLED" "STRIG" "ATRIG" "PLOW" "PHIGH" "XTRIG" "100PPS" "TPOP" (see section 7.8.1)</p> <p>frequency ::= Sets timer programmable frequency. Range: 100 - 5115000 Hz, in 1 Hz steps) Default: 512 Hz. Required only when using mode = TPOP</p> <p>Note: Frequency setting is lost after ac power cycling the signal generator.</p>
Returns	Returns the timer settings
Example	<p>Set timer 3 of a single 7xxx signal generator permanently high:</p> <pre>SET_TIOP,0,3,GATED,PHIGH</pre> <p>Set the single timer of a single 5060 signal generator permanently high:</p> <pre>SET_TIOP,0,1,NOTGATED,PHIGH</pre> <p>Set timer 2 of a single 7xxx signal generator to a programmable frequency of 1008 Hz:</p> <pre>SET_TIOP,0,2,GATED,TPOP,1008</pre>

Table 5-19 details the equivalent 47xx or 7xxx messages you must use when setting the 5060 or 6560 timer.

Table 5-19: 6560/5060 timer command and 47xx/7xxx equivalents

6560/5060 timer command	47xx / 7xxx equivalent timer command
HIGH	NOTGATED, PHIGH
LOW	NOTGATED, PLOW
1PPS	NOTGATED, 1PPS
GATED	GATED, 1PPS
RISE	GATED, HIGH

5.4 Antenna Pattern commands

Command	Description
Antenna Pattern commands	
ANT_PAT_NUM	Sets the antenna pattern to use at the given time
SET_ANT_OFFSET	Sets the antenna offset to use at the given time

5.4.1 ANT_PAT_NUM

Description	Sets the antenna pattern for a given time on a given vehicle and antenna. Before using this command, you must select Scenario-Vehicle-Antenna-Switching file :
Format	<timestamp>,ANT_PAT_NUM,<veh_ant>,<pattern_num>
Where	<p>timestamp ::= see section 5.1.1</p> <p>veh_ant ::= <vehicle_id>" "<antenna_id></p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p>antenna_id ::= "a"<a_number></p> <p>a_number ::= antenna number starting from 1</p> <p>pattern_num ::= 0-4</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, set vehicle 1, antenna 1 to use antenna pattern 1:</p> <pre>-,ANT_PAT_NUM,v1_a1,1</pre>

5.4.2 SET_ANT_OFFSET

Description	Sets the antenna offset for a given time on a given vehicle and antenna. Before using this command, you must select Scenario-Vehicle-Antenna-Antenna pattern control-Switching file :
Format	<timestamp>,SET_ANT_OFFSET,<veh_ant>,<x,y,z>,<heading>,<elevation>,<bank>
Where	<p>timestamp ::= see section 5.1.1</p> <p>veh_ant ::= <vehicle_id>" "<antenna_id></p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p>antenna_id ::= "a"<a_number></p> <p>a_number ::= antenna number starting from 1</p> <p>x ::= (metres, range: -500 to +500)</p> <p>y ::= (metres, range: -500 to +500)</p> <p>z ::= (metres, range: -500 to +500)</p> <p>heading ::= (degrees)"." (decimal minutes; +ve or -ve)</p> <p>elevation ::= (degrees)"." (decimal minutes; +ve or -ve)</p> <p>bank ::= (degrees)"." (decimal minutes; +ve or -ve)</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, set the antenna pattern offset for vehicle 1, antenna 1 to x = -50.3 m, y = 10 m, z = -20.75 m, heading -10.75°, elevation = 75.1°, bank = 35°</p> <pre>-,SET_ANT_OFFSET,v1_a1,-50.3,10,-20.75,-10.5,75.1,35</pre>

5.5 Signal Power commands

Command	Description
Signal Power commands	
POW_ON	Set the power on or off by Channel or SVID
POW_MODE	Obsolete command - see POW_LEV
POW_LEV	Set the power level by Channel or SVID
REF_DBM	Sets the reference label
SIGNAL_STRENGTH	Sets signal into fixed or modelled mode
SAT_POW_OFFSET	Offsets the RF Power level of a given SVID.
LAAS_SIG_LEV	Set the LAAS signal level

Choosing either Channel or SVID in a remote command achieves the same result - the changes you specify are always applied to a simulator channel. Table 5-20 outlines the differences between Channel and SVID.

Table 5-20 Choosing between Channel and SVID

Parameter	Duration of Parameter settings	Outline of SimGEN actions for remote Signal Power commands
Channel	Remainder of scenario (unless changed)	SimGEN automatically applies settings to that Channel. During the scenario, SimGEN can assign any SVID to that Channel.
SVID	While SVID is visible	At the time of command, SimGEN scans all available channels for the SVID. If the SVID is present, SimGEN applies the Channel settings.

The following sections give rules and examples for using Channel or SVID (also known as Tx Channel).

5.5.1.1 Channel settings - rules and example

Channel settings apply for the remainder of the scenario. When SimGEN moves an SVID to a new Channel, it always applies the settings of the new Channel. SimGEN does not retain SVID settings when you move the SVID between Channels.

For example: You set SVID 1 to use the RF Power Level -125 dBm on Channel 5. Later in the scenario, you force SVID 1 onto Channel 9.

SimGEN does not retain the -125 dBm RF Power Level when it moves SVID 1 from Channel 5 to Channel 9. SimGEN will apply the default RF Power Level for Channel 9 (for example, -130 dBm) to SVID 1.

A new SVID appearing on Channel 5 uses the -125 dBm RF Power Level you had set previously.

If you want to assign a specific RF Power Level to a specific SVID, Spirent recommend you use **Constellation-Signal sources file-Signal control-Signal power**, see reference a).

5.5.1.2 SVID settings - rules and examples

SimGEN applies SVID settings only while the SVID is visible. SimGEN does not retain non-default SVID settings after the SVID sets.

For example: In a scenario, SimGEN automatically assigns SVID 1 to Channel 10. During the scenario, you set a +20 dB relative RF Power Level on SVID 1. Later in the scenario, SVID 1 sets. Later still, SVID 1 rises again and SimGEN automatically assigns SVID 1 to Channel 3 (as Channel 10 is assigned to another SVID). SVID 1 uses the settings for Channel 3.

Channel 10 retains the Power Level settings from your earlier setting for SVID 1 for the duration of the scenario.

Note: This only applies for *POW_LEV*. With other commands, when an SVID starts on a new channel, it uses the default Channel settings.

If you want to set the RF Power Level of a specific SVID permanently Off, Spirent recommends you remove that SVID from the constellation by using SimGEN to deselect **Constellation-Signal sources file-Motion-Orbits-Present**, see reference a).

5.5.1.3 Run once and move on - rules and examples

SimGEN does not check for the presence of an SVID in a constellation before running a remote command. SimGEN runs each remote command once and moves to the next command. SimGEN produces an **Information** error message in the **System Messages** window when it runs a remote command for an SVID that is not present in a constellation (see reference a) for details on setting the message types shown in the **System Messages** window).

For example: The default setting for SVID 1 is RF On. During your scenario, a remote command sets SVID 1 to RF Off. However, at the time of this remote command, SVID 1 has yet to appear in the constellation. SimGEN runs the remote command, which cannot be applied to SVID 1. SVID 1 subsequently appears in the constellation with RF On (its default setting).

If you want to ensure an SVID appears in a constellation with particular (non-default) settings, Spirent recommend you determine the time the SVID appears in the constellation and apply remote commands at, or just after, this time.

5.5.2 Shared parameters

Some parameters are common across all signal power commands:

Shared Signal Power Parameters	<code>timestamp</code>	::= see section 5.1.1
	<code>veh_ant</code>	::= <vehicle_id>" "<antenna_id>
	<code>vehicle_id</code>	::= "v"<v_number>
	<code>v_number</code>	::= vehicle number starting from 1
	<code>antenna_id</code>	::= "a"<a_number>
	<code>a_number</code>	::= antenna number starting from 1
	<code><signal_type></code>	::= "EGNOS" "GALILEO" "GLONASS" "GROUND TX" "GPS" "MSAS" "WAAS"
	Note: If you do not specify <code>signal_type</code> , it is assumed to be GPS.	
	<code>id</code>	::= 0-n (Channel number) 1-n (Satellite SVID)
	<code>muti_index</code>	::= "0" (incident signal) (number of the multipath starting at 1 - only applies in SVID mode)
	<code>mode</code>	::= "0" (svid mode, use id = 1-n) "1" (channel mode, use id = 0-n)
	<code>all_flag</code>	::= "0" (apply to specified channel / SVID) "1" (apply to all channels / SVIDs)

5.5.3 POW_ON

Description	Turn RF power On or Off at a specified time.
Format	<timestamp>, POW_ON, <veh_ant>, <state>, <signal_type>, <id>, <multi_index>, <mode>, <all_flag>, <all_tx_type>
Where	<pre>state ::= "0" (off) "1" (on) all_tx_type ::= "0" (only for specified Channels/SVID of signal_type) "1" (for all Channels/SVID of signal_type)</pre> <p>See shared signal power parameters in section 5.5.2</p>
Returns	Status see returned response format, section 5.1.2
Example(s)	<p>Now, turn Off channel 0 (shown as 1 in the windows interface), for vehicle 1 antenna 1</p> <pre>-, POW_ON, v1_a1, 0, GPS, 0, 0, 1, 0</pre> <p>5 s into run, turn On (assumed GPS satellite) SVID 1 for vehicle 2 antenna 1</p> <pre>0 00:00:05, POW_ON, v2_a1, 1, 1, 0, 0, 0</pre>

5.5.4 POW_MODE

Description	<p>Changes the power mode by Channel or satellite.</p> <p>Note: The POW_MODE command is obsolete due to the “absolute” flag in the POW_LEV command.</p>
Format	<timestamp>, POW_MODE, <veh_ant>, <state>, <signal_type>, <id>, <multi_index>, <mode>, <all_flag>, <all_tx_type>
Where	<pre>state ::= "0" (absolute power) "1" (relative power) all_tx_type ::= "0" (only for specified Channels/SVID of signal_type) "1" (for all Channels/SVID of signal_type)</pre> <p>See shared signal power parameters in section 5.5.2</p>
Returns	Status see returned response format, section 5.1.2
Example	<p>Now, make all GPS channels use absolute power mode for vehicle 1 antenna 1</p> <pre>-, POW_MODE, v1_a1, 0, GPS, 0, 0, 1, 1</pre>

5.5.5 POW_LEV

Notes:

- 1) If you send an absolute POW_LEV command you have set the mode to absolute and subsequent MOD commands will have no effect.
- 2) If you send a relative POW_LEV command you have set the mode to relative. Relative POW_LEV and MOD commands will offset the power level from the originally calculated value.
- 3) Up to SimGEN 2-80, SimREMOTE would not action this command if specified with an SVID that was not being simulated at <timestamp>, even if <all_flag> was set. From SimGEN 2-81 onwards, if the <all_flag> is set SimREMOTE will action the command whatever the SVID.

Description	<p>Sets the power level by Channel or satellite.</p> <p><align> is only relevant when you set <all_flag> to 1.</p> <p>Table 5-21 details the combinations of <absolute> and <align> when <all_flag>=1</p>
Format	<timestamp>, POW_LEV, <veh_ant>, <level>, <signal_type>, <id>, <multi_index>, <mode>, <all_flag>, <absolute>, <align>, <all_tx_type>
Where	<p>level ::= power level (dB with respect to the Stanag minimum; or offset in dB, see <absolute>)</p> <p>absolute ::= "0" (relative power) "1" (absolute power)</p> <p>align ::= "0" (off) "1" (on)</p> <p>all_tx_type ::= "0" (only for specified Channels/SVID of signal_type) "1" (for all Channels/SVID of signal_type)</p> <p>See shared signal power parameters in section 5.5.2</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, set power level for vehicle 1 antenna 1 on GPS satellite SVID 23 to an absolute level of 10.5 dB</p> <pre>-, POW_LEV, v1_a1, 10.5, GPS, 23, 0, 0, 0, 1, 0</pre>

Table 5-21 Combining <absolute> and <align> when <all_flag>=1

<absolute>	<align>	Description
1	1	Sets all Channels to the value specified
1	0	Sets specified Channel/SVID to this absolute value. Offsets levels of all other Channels from their current values by the change in the specified Channel.
0	1	Offsets all the Channels by the specified amount from their levels, as calculated in the unmodified scenario.
0	0	Offsets specified Channel/SVID by the specified amount from its level as calculated in the unmodified scenario. Offsets all other Channels from their current values by the change in the specified Channel.

5.5.6 REF_DBM

Description	By default, power bar graphs and sliders display signal levels with respect to a reference level of -130 dBm. This command changes the label of the reference level to the value specified. Note: <i>This does NOT affect the generated signal level.</i>
Format	REF_DBM, <level>
Where	level ::= integer value of dBm
Returns	status see returned response format, section 5.1.2
Example	Set reference level to -150 dBm: REF_DBM, -150

5.5.7 SIGNAL_STRENGTH

Description	Sets the selected constellation to fixed or modelled signal strength
Format	SIGNAL_STRENGTH, <signal_type>, <fixed_modelled>
Where	fixed_modelled ::= "0" (modelled) "1" (fixed) See shared signal power parameters in section 5.5.2
Returns	Status see returned response format, section 5.1.2
Example	Set GPS constellation to fixed signal strength: SIGNAL_STRENGTH, GPS, 1

5.5.8 SAT_POW_OFFSET

Description	Offsets the RF Power level of a specified SVID. The new RF Power level remains with the specified SVID if it moves channels.
Format	<timestamp>, SAT_POW_OFFSET, <signal_type>, <svid>, <offset>
Where	offset ::= Power level offset, dB (added to "Global offset" in SimGENs constellation file using the signal power page) offset range: +40.0 to -60.0 dB See shared signal power parameters in section 5.5.2
Returns	status see returned response format, section 5.1.2
Example	Now, Offset the RF Power level of GPS SVID 22, level -5.5 dB: -, SAT_POW_OFFSET, GPS, 22, -5.5 Note: <i>This offset of -5.5 dB remains with SVID 22 on any channel until the end of this scenario, or a new SAT_POW_OFFSET command</i>

5.5.9 LAAS_SIG_LEV

Note: Apply this command before starting the scenario

Description	Set the power of the LAAS signal. See reference a) for more details.
Format	<timestamp>,LAAS_SIG_LEV,<fix_mod>,<ext_loss>,<rx_lev>,<rx_tx_dist>
Where	fix_mod ::= "0" (fixed) "1" (modelled) ext_loss ::= 0-9 (external loss in dB) rx_lev ::= -50 - -90 (level at receiver, dBm) rx_tx_dist ::= (receiver to transmitter distance, nautical miles) only valid when <fix_mod>="1"
Returns	status see returned response format, section 5.1.2
Example	Now, set the LAAS signal power to be modelled, using an external loss of 5 dB, a receiver power level of -60 dBm and a receiver to transmitter distance of 10 nautical miles. -,LAAS_SIG_LEV,1,5,-60,10

5.6 Signal Control commands

Command	Description
Signal Control commands	
SWITCH_SAT	Defines the state of signals for visible satellites.
MP_SWITCH	Used with MOD command to define multipath signals
MOD	Defines level, code and carrier offset data for an SVID
LMM_SELECT	Select Land Mobile Multipath Model attributes
FADER (embedded multipath)	Add up to 4 multipath signals (sub-channels) per Channel
CODE_LEV_OFFSET	Applies signal level control to each component (C/A, P, Y)

Choosing either Channel or SVID in a remote Signal Control command achieves the same result - SimGEN applies the changes you specify to an SVID. Table 5-22 outlines the differences between Channel and SVID.

Table 5-22 Choosing between Channel and SVID

Parameter	Duration of Parameter settings	Outline of SimGEN actions for remote Signal Control commands
Channel	While SVID is visible	SimGEN applies the command to the specified Channel only if it is simulating an SVID at that time. Once that original SVID leaves the Channel, SimGEN restores the default Channel conditions.
SVID	While SVID is visible	SimGEN applies the command to the specified SVID only if it is assigned to a signal generator Channel at that time. Once the SVID leaves the Channel, SimGEN restores the default Channel conditions.

5.6.1 SWITCH_SAT

Notes

- 1):** All **SWITCH_SAT** commands apply only while the satellite is visible. If the satellite sets and subsequently rises during your scenario, the newly risen satellite is simulated in **State-Normal**.
- 2)** Certain **SWITCH_SAT** remote commands are complex. Spirent recommends you first copy the examples given in this section and run them to see their effect; then adjust the command parameters as required.
- 3):** Remote command channel numbering starts from 0 (channel numbering in the SimGEN **Channel Assignment** dialog starts from 1).
- 4)** SimREMOTE performs **SWITCH_SAT** commands in advance, on sampling interval boundaries, as part of the periodic satellite selection routine. Set the sampling interval using **[Constellation]-Satellite selection-Sampling interval**, see reference a). As the **Sampling interval** range is 1 to 600 seconds (default = 6 s) you must ensure you send **SWITCH_SAT** commands well in advance to ensure SimREMOTE actions the command at the time you want.
- 5)** You must take into account the significant fraction of a second it takes to start a channel. For example, if you set **Sampling interval** = 1 second, Spirent recommends you do not send **SWITCH_SAT** commands to turn ON/OFF a particular satellite at a rate of 1 second.

This command defines the state of signals for visible satellites and is the remote version of SimGEN's **Channel Assignment** toolbar button, see reference a).

Each command gives details of its specific parameters and an example of the command.

Common parameters

All SWITCH_SAT commands use the following parameters:

```
veh_ant_tn  ::= <vehicle_id>"_"<antenna_id>"_"<tx_net>
vehicle_id  ::= "v"<v_number>
v_number    ::= vehicle number starting from 1
antenna_id  ::= "a"<a_number>
a_number    ::= antenna number starting from 1
tx_net      ::= "VT"<tx_num>
tx_num      ::= "1" (GPS) | "2" (WAAS) | "3" (EGNOS) | "4" (MSAS)
              | "5" (GLONASS) | "10" (GALILEO) | "11" (Ground Tx)
              | "12" (QZ)
svid        ::= satellite svid
```

NORMAL

Description

Note: The **Normal** command has no effect at the start of a scenario

Returns the SVID to its "normal" state (counteracts other SWITCH_SAT commands). See Note 1 above.

Format

```
<timestamp>, SWITCH_SAT, <veh_ant_tn>, <svid>, <switch_type>,
```

Where

```
<switch_type>    ::= "0"
```

Return

status see returned response format, section 5.1.2

Example

At 1.600 seconds into the run, set vehicle 1, antenna 1 GPS SVID18 to Normal

```
00:00:01.600, SWITCH_SAT, V1_A1_VT1, 18, 0
```

FORCED

Description

Forces an SVID to use a Channel you specify.

Format

```
<timestamp>, SWITCH_SAT, <veh_ant_tn>, <svid>, <switch_type>,
<forced_channel>
```

Where

```
switch_type ::= "1"
```

```
forced_channel ::= channel on which to place <svid>,
                  numbering from 0
```

Return

status see returned response format, section 5.1.2

Example

At start of scenario, Force vehicle 1, antenna 1, GPS SVID 3 onto Channel 3

```
00:00:00, SWITCH_SAT, V1_A1_VT1, 3, 1, 3
```


BANNED

Description	Removes an SVID from the scenario.
Format	<timestamp>, SWITCH_SAT, <veh_ant_tn>, <svid>, <switch_type>, <switch_type> ::= "2"
Where	
Return	status see returned response format, section 5.1.2
Example	At start of scenario, for vehicle 1, antenna 1, Ban GPS SVID 6 00:00:00, SWITCH_SAT, V1_A1_VT1, 6, 2,

MULTI PATH	See Note 2) above
Description	<p>Defines multipath signals on a selected SVID and Channel.</p> <p>Use for ground reflection, fixed offset, Doppler offset, vertical plane, reflection pattern, Legendre, polynomial and sinusoidal multipath types.</p>
Format	<pre><timestamp>, SWITCH_SAT, <veh_ant_tn>, <svid>, <switch_type>, <mpth_type>, <remove_los>, <num_mpths>, <num_data_sets>, <data_sets></pre>
Where	<pre>switch_type ::= "3" mpth_type ::= "0" (Ground reflection / Line of sight, LOS) "1" (Fixed offset) "2" (Doppler offset) "3" (Vertical plane) "4" (Reflection pattern) "5" (Legendre) "6" (Polynomial) "7" (Sinusoidal) remove_los ::= remove LOS? "0" (no) "1" (yes) num_mpths ::= number of multipaths or echoes num_data_sets ::= number of <data_sets> = num_mpths+1 LOS always has a <data_set>, even if you remove it. data_sets ::= <data_set>{,, <data_set>,, ... ,, <data_set>} data_set ::= "<41>"<echo_no>,<channel>,<attenuation>, <range_offset>,<initial_delay_in_chips>, <doppler_offset>,<initial_phase>, <carrier_phase>,<carrier_doppler>, <reflection_loss>,<leg_A0>,<leg_A1>, <leg_A2>,<leg_A3>,<leg_A4>,<leg_A5>, <leg_D0>,<leg_D1>,<leg_D2>,<leg_D3>, <leg_D4>,<leg_D5>,<duration>,<poly_A1>, <poly_A2>,<poly_A3>,<poly_A4>,<poly_A5>, <poly_D1>,<poly_D2>,<poly_D3>,<poly_D4>, <poly_D5>,<sin_attn_peak>, <sin_attn_freq>,<sin_attn_phase>, <attenuation_bias>,<sin_delay_peak>, <sin_delay_freq>,<sin_delay_phase>, <delay_bias></pre> <p>For ease of use, every 10th <data_set> parameter shown above is inverted.</p> <p>Notes:</p> <ol style="list-style-type: none"> 1) There is no comma after <41> the first parameter in a <data_set>, all other parameters are separated by a single comma. 2) <data_sets> comprises one or more <data_set> fields. 3) Two commas separate each <data_set> field. 4) Terminate <data_sets> with two commas after the last <data_set>.

Where
(continued)**<data_set> parameters**

See Note 4) above

Each <data_set> consists of 41 parameters common to all types of multipath. You must set non-relevant parameters to zero.

Common parameters (used in each mpath_type)

<41> ::= number of parameters in data set.

echo_no ::= "0" (incident) | "1" (first mpath) | "2" (second mpath and so on, for all mpaths

channel ::= "-1" (not forced) | forced channel number

Parameters for specific Multipath types

Set non-relevant parameters to zero.

Ground reflection (mpath_type 0)

attenuation ::= attenuation in dB

initial_phase ::= "-1" (not applicable)

Fixed offset (mpath_type 1)

attenuation ::= attenuation in dB

range_offset ::= range offset in metres

initial_phase ::= "-1" (not applicable)

Doppler offset (mpath_type 2)

attenuation ::= attenuation in dB

initial_delay_in_chips ::= initial delay in C/A chips

doppler_offset ::= Doppler offset in Hz

initial_phase ::= "0" (not random)

| "1" (random)

| "-1" (not applicable)

carrier_phase ::= carrier phase in radians

(only used if initial_phase = 0,

otherwise carrier_phase = 1 (random))

Note: The Doppler offset (in Hz) is applied to L1.

The L2 Doppler offset is derived from the L2 pseudorange rate offset, which in SimGEN has been set to be equal to the L1 pseudorange rate offset (in $m.s^{-1}$).

Where
(continued)**Vertical plane (mpath_type 3)**

```

initial_phase      ::= "0" (not random)
                   | "1" (random)
                   | "-1" (not applicable)
carrier_phase      ::= carrier phase in radians
                   (used if initial_phase = 0)
carrier_doppler    ::= carrier Doppler in Hz
reflection_loss     ::= reflection loss in dB

```

Note: You must select **Scenario-Vehicle-Options-Vertical plane file**.

Reflection patterns (mpath_type 4)

```

initial_phase      ::= "-1" (not applicable)

```

Note: You must select **Scenario-Vehicle-Antenna-Options-Enable reflection pattern multipath**.

Legendre (mpath_type 5)

```

initial_phase      ::= "-1" (not applicable)
leg_A0-leg_A5      ::= relative amplitude coefficients
leg_D0-leg_D5      ::= delay coefficients, seconds
duration           ::= duration of a single period, seconds

```

Polynomial (mpath_type 6)

```

initial_phase      ::= "-1" (not applicable)
duration           ::= duration of a single period, seconds
poly_A1-poly_A5    ::= amplitude coefficients, dB
poly_D1-poly_D5    ::= delay coefficients, seconds
attenuation_bias   ::= coefficient A0, dB
delay_bias         ::= coefficient D0, seconds

```

Sinusoidal (mpath_type 7)

```

initial_phase      ::= "-1" (not applicable)
sin_attn_peak      ::= attenuation peak, dB
sin_attn_freq       ::= attenuation frequency, Hz
sin_attn_phase      ::= attenuation phase, radians
attenuation_bias    ::= attenuation bias, dB
sin_delay_peak      ::= delay peak, nano-seconds
sin_delay_freq      ::= delay frequency, Hz
sin_delay_phase     ::= delay phase, radians
delay_bias          ::= delay bias, nano-seconds

```

Returns

```

status see returned response format, section 5.1.2

```

Examples

Note: Use of double commas to separate each <data_set>.

For ease of use, every 10th <data_set> parameter in these examples is **inverted**

Ground reflection

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is ground reflection (LOS), LOS not removed, 2 multipaths, 3 data sets (LOS + 2 multipaths),

Details of applicable <data_set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, 5dB attenuation, random phase not applicable.

Second multipath: 41 parameters, echo 2, forced to channel 3, 6dB attenuation, random phase not applicable.

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 0, 0, 2, 3,
```

<41>0, -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,,

<41>1,-1,5,0,0,0,-1,0,0,**0**,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,
0,0,0,**0**,0,0,0,0,0,0,0,0,**0**,0,,

[illegible]

Note: There is no comma after <41> the first parameter in a <data_set>, all other parameters are separated by a single comma.

Fixed offset

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is fixed offset, LOS not removed, 1 multipath, 2 data sets (LOS + 1 multipath)

Details of applicable <data set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, 2.2 dB attenuation, 2.3 m range offset, random phase not applicable.

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 1, 0, 1, 2,
```

$$\langle 41 \rangle 0, -1, 0, 0, 0, 0, -1, 0, 0, \boxed{0}, 0, 0, 0, 0, 0, 0, 0, 0, \boxed{0}, 0, 0, 0, 0, 0, 0, \\ 0, 0, 0, \boxed{0}, 0, 0, 0, 0, 0, 0, 0, 0, 0, \boxed{0}, 0, ,$$

<41>1,-1,2.2,2.3,0,0,-1,0,0,**0**,0,0,0,0,0,0,0,0,**0**,0,0,0,0,
0,0,0,0,0,**0**,0,0,0,0,0,0,0,0,0,**0**,0,

Examples (continued)

Doppler offset

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is Doppler offset, LOS not removed, 2 multipaths, 3 data sets (LOS + 2 multipaths)

Details of applicable <data_set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, 3.3dB attenuation, 3.1 C/A chips initial delay, 3.2 Hz Doppler offset, random phase.

Second multipath: 41 parameters, echo 2, not forced, 3.6dB attenuation, 3.4 C/A chips initial delay, 3.5 Hz Doppler offset, 0.0645772 radians carrier phase.

```
00:00:00, SWITCH _SAT, V1_A1_VT1, 4, 3, 2, 0, 2, 3,
```

[illegible]

<41>1,-1,3.3,0,3.1,3.2,1,0,0,**0**,0,0,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,0,0,0,**0**,0,,

```
<41>2,-1,3.6,0,3.4,3.5,0,0.0645772,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,
```

Vertical plane

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is Vertical plane, LOS is removed, 2 multipaths, 3 data sets (LOS + 2 multipaths),

Details of applicable <data_set> parameters, all other parameters set to 0:

Note: Although LOS is removed you must enter data for the 41 LOS parameters (this data can be non-zero)

LOS: 41 parameters, echo 0, (not forced), (0 dB attenuation), (random phase not applicable).

First multipath: 41 parameters, echo 1, not forced, random phase, 4.2 Hz carrier Doppler, and 4.1dB reflection loss.

Second multipath: 41 parameters, echo 2, not forced, 0.0785398 radians carrier phase, 4.4 Hz carrier Doppler, 4.3dB reflection loss

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 3, 1, 2, 3,
```

 $\langle 41 \rangle = -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
 $0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$ [illegible][illegible]

Examples (continued)

Reflection pattern

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is Reflection pattern, LOS not removed, 1 multipaths, 2 data sets (LOS + 1 multipath)

Details of applicable <data set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, random phase not applicable

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 4, 0, 1, 2,
```

<41>0,-1,0,0,0,0,-1,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,
0,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,,

<41>1,-1,0,0,0,0,-1,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,
0,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,,

Legendre

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is Legendre, LOS not removed, 1 multipaths, 2 data sets (LOS + 1 multipath)

Details of applicable <data set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, random phase not applicable, relative amplitude coefficient A0 (6.1) to A5 (6.15), delay coefficient D0 (6.2 s) to D5 (6.25 s), duration 60 s

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 5, 0, 1, 2,
```

<41>0,-1,0,0,0,0,-1,0,0,**0**,0,0,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,
0,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,,

```
<41>1,-1,0,0,0,0,-1,0,0,0,6.1,6.11,6.12,6.13,6.14,6.15,6.2,
6.21,6.22,6.23,6.24,6.25,60,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,
```

Polynomial

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is Polynomial, LOS not removed, 1 multipath, 2 data sets (LOS + 1 multipath)

Details of applicable <data set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, random phase not applicable, duration 7.3 s, amplitude coefficient A1 (7.11 dB) to A5 (7.15 dB), delay coefficient D1 (7.21 s) to D5 (7.25 s), attenuation bias 7.1 dB, delay bias 7.2 ns.

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 6, 0, 1, 2,
```

$$\langle 41 \rangle = -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ,$$

<41>1,-1,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7.3,7.11,
7.12,7.13,7.14,7.15,7.21,7.22,7.23,7.24,7.25,0,0,0,7.1,0,0,0,7.2,,

Examples (continued)

Sinusoidal

At start of scenario, set Multipath for vehicle 1 antenna 1 on GPS SVID4, multipath type is Sinusoidal, LOS not removed, 1 multipath, 2 data sets (LOS + 1 multipath)

Details of applicable <data set> parameters, all other parameters set to 0:

LOS: 41 parameters, echo 0, not forced, 0 dB attenuation, random phase not applicable.

First multipath: 41 parameters, echo 1, not forced, random phase not applicable, attenuation peak 8 dB, attenuation frequency 8.1 Hz, attenuation phase 0.143117 radians, attenuation bias 8.3 dB, delay peak 8.4 ns, delay frequency 8.5 Hz, delay phase 0.150098 radians, delay bias 8.7 ns

```
00:00:00, SWITCH SAT, V1 A1 VT1, 4, 3, 7, 0, 1, 2,
```

<41>0,-1,0,0,0,0,-1,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,0,0,0,0,0,
0,0,0,**0**,0,0,0,0,0,0,0,0,0,0,**0**,0,,

```
<41>1,-1,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,8,8.1,0.143117,8.3,8.4,8.5,0.150098,8.7,,
```

INCLUDE

Description

Includes a specific SVID in the scenario.

Format

```
<timestamp>, SWITCH SAT, <veh ant tn>, <svid>, <switch type>,
```

Where

```
<switch type> ::= "4"
```

Return

status see returned response format, section 5.1.2

Example

At start of scenario, set vehicle 1, antenna 1, to Include GPS SVID 17

00:00:00, SWITCH SAT, V1 A1 VT1, 17, 4,

5.6.2 MP_SWITCH

Notes:

1) This command is actioned in the low rate thread which runs at a rate that you can select, of 1,2,3,6,30,... seconds. The timestamp must be 3 seconds in advance of the time at which the signal is required to appear. For example: you select a low rate thread of 6 s and use a command: 0 00:00:03.00,MP_SWITCH,v1_a1,GPS,14,1,1,9

Multipath signals appear 6 seconds after the start of the scenario.

In real-time mode, this command must be received at least 3 seconds BEFORE the multipath signal is to appear at the required update time for the simulated satellite.

2) Only use <other_chan> if you are ADDING multipaths.

3) REMOVING *n* multipaths removes the LAST *n* added.

4) A list of multipaths is maintained for each SVID, using a multipath index number 1,2,3.... When a multipath is deleted the last one is removed from the list.

5) If you do not explicitly specify <chan> SimGEN automatically determines the channel allocation.

Description	<p>Defines multipath signals affected by the MOD command.</p> <p>No offsets will be applied unless specified by a MOD command for that Channel. The MOD command MODifies the signal level and pseudorange for a specified signal, or one of its multipaths, at a particular frequency.</p> <p>To use this command the timestamp must be at least 3 seconds before the boundary on which you want to apply the command.</p>
Format	<pre><timestamp>,MP_SWITCH,<veh_ant>,<signal_type>,<svid>,<add_remove>,<number>,{<chan>,<other_chan>}</pre>
Where	<pre>timestamp ::= see section 5.1.1 veh_ant ::= <vehicle_id>"_"<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 signal_type ::= "EGNOS" "GALILEO" "GLONASS" "GROUND TX" "GPS" "MSAS" "WAAS" svid ::= svid number add_remove ::= add or remove multipath signals? "0" (remove) "1" (add) number ::= integer number of multipath signals to add_remove, see Notes 2,3 and 4 chan ::= 0-n other_chan ::= use when number>1 to define the additional channel numbers. See Note 2) above</pre>
Returns	status see returned response format, section 5.1.2
Example	<p>At start of scenario, for vehicle 1 antenna 1 GPS SVID 14, add 2 multipaths to channel 10 and 12</p> <pre>0 00:00:00.00,MP_SWITCH,v1_a1,GPS,14,1,2,9,11</pre>

5.6.3 MOD

Notes:

1) For correct time synchronisation of the pseudorange offset data the MOD commands should be supplied 30 ms in advance. (This must be 400 ms for STR4760 systems when using a simulation iteration rate of 100 ms).

SimGEN accepts remote commands for processing when the current run-time is within 400 ms of the command reference time (timestamp).

The MOD commands are buffered, so when read from file they will always be available at the time they need to be applied. In the case of commands being sent from a remote system (or separate process on the same PC) the commands can be sent with the correct timestamps, but they must be sent early by the required amount if it is necessary to get exact correspondence of the simulated and input profiles with time.

For rapidly varying data, the MOD commands should be supplied at a rate of between 10 and 100 Hz. Where data is available in advance, SimGEN will interpolate as required, and will apply changes in code and carrier pseudorange offset linearly over the simulation iteration rate period. If the definition of MOD commands is infrequent or irregular so that there is no future data, the offset values are held at the levels defined by the last command received.

2) Use NBLK2 for 100 Hz operation, section 5.10.4.

3) When using MOD data from the start of a scenario the first truth values generated for level and pseudorange are not influenced by the MOD data.

4) If you send an absolute POW_LEV command you have set the mode to absolute and subsequent MOD commands will have no effect.

5) If you send a relative POW_LEV command you have set the mode to relative. Relative POW_LEV and MOD commands will offset the power level from the originally calculated value

Description	<p>Defines signal offsets applied to direct or multipath signals.</p> <p>Use the NBLK2 option (see section 5.10.4) if applying offsets at 4, 5 or 10 ms intervals for GSS7700 signal generators.</p> <p>The power range you can apply is:</p> <p>Upper limit for all signal generators: +20 dBm Lower limit for 6560, 7xxx and GSS8000 generators: -49.8 dBm Lower limit for 4760 generators: -39.9 dBm</p> <p>Note: If you set power levels lower than these values, the signal generator LCD shows "Off".</p> <p>Contact Spirent for information on power accuracy below -20 dBm.</p>
Format	<pre><timestamp>,MOD,<veh_ant>,<signal_type>,<id>,<multi_index>,<mode>,<all_flag>,<freq>,<all_freq>,<sig_level>,<carr_offset>,<code_offset></pre>
Where	<p>These are used in Table 5-29: UDP commands for MOD</p> <pre>timestamp ::= see section 5.1.1 veh_ant ::= <vehicle_id>"_"<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 signal_type ::= "EGNOS" "GALILEO" "GLONASS" "GROUND TX" "GPS" "MSAS" "WAAS" id ::= 0-n (Channel number) 1-n (Satellite SVID) muti_index ::= "0" (incident signal) (number of the multipath starting at 1 - only applies in SVID mode) mode ::= "0" (svid mode, use id = 1-n) "1" (channel mode, use id = 0-n) all_flag ::= "0" (apply to specified channel / SVID) "1" (apply to all channels / SVIDs) freq ::= GPS "0" (L1) "1" (L2) "2" (L5) GALILEO "0" (E1) "1" (E6) "2" (E5) GLONASS "0" (L1) "1" (L2) EGNOS "0" (L1) "2" (L5) MSAS "0" (L1) "2" (L5) WAAS "0" (L1) "2" (L5) Note: If you want to use L5 for any SBAS satellite, you must use freq = "2". all_freq ::= "0" (single frequency) "1" (all frequencies) sig_level ::= signal level, dB, decimal value positive value gives increase in level carr_offset ::= carrier offset, m, decimal value positive value gives increase in prange code_offset ::= code offset, m, decimal value positive value gives increase in prange</pre>
Returns	<pre>status see returned response format, section 5.1.2</pre>

Examples**Example 1:**

```
0 00:00:00.00,MOD,v1_a1,gps,14,0,0,0,0,1,3.2,10.7,10.65
```

Apply 3.2 dB increase in level, 10.7 m increase in carrier prange and a 10.65 m increase in code prange for SVID 14, on all frequencies.

Example 2: MOD and MP_SWITCH command

Analysis of the logged data from the pre-run shows SVID 3 present on Channel 1 and you want to add a multipath copy of that signal at 6 seconds into the run, on Channel 10, for 6 seconds.

Simulated satellite set calculation rate is set to 6 s.

Simulation iteration rate is set to 10 ms.

Only L1 commands are shown. The command file would look like this:

```
00:00:03,MP_SWITCH,v1_a1,GPS,3,1,1,9
```

Note: *MP on, must be sent early*

```
00:00:06.00,MOD,v1_a1,GPS,3,9,1,1,0,0,3.0,10.6,10.7
```

```
00:00:06.01,MOD,.....
```

```
00:00:06.02,MOD,.....
```

...

```
00:00:08.99,MOD,....
```

```
00:00:09.00,MP_SWITCH,v1_a1,GPS,3,0,1
```

Note: *MP off, must be sent early*

```
00:00:09.00,MOD,.....
```

```
00:00:09.01,MOD,.....
```

...

```
00:00:11.99,MOD,.....
```

Insert additional multipath commands for this or other satellites in the same time-ordered sequence.

5.6.4 SWITCH_LMM

Description	<p>Select a new Land Mobile Multipath environment and / or category mask.</p> <p>Before you use this command, Spirent recommends you examine the environment and category mask you want to use by using the Land Mobile Multipath editor (Scenario-Vehicle-Options-Land mobile multipath) and ensure the details are those you want to use. The editor gives a number of pre-set masks you can edit.</p>
Format	<p>Land Mobile Multipath models apply to both static and land vehicles.</p> <p><timestamp>, SWITCH_LMM, <vehicle_id>, <environment>, <mask></p>
Where	<pre>timestamp ::= see section 5.1.1 vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 environment ::= "0" (Rural (Old)) "1" (Suburban (Old)) "2" (Urban (Old)) "3" (Highway (Old)) "4" (Rural) "5" (Suburban) "6" (Urban) "7" (Highway) "8" (Forest) "9" (Open park) mask ::= "0" (Default) "1" (Urban canyon (Old)) "2" (Highway flyover (Old)) "3" (Trees (Old)) "4" (Urban canyon) "5" (Trees) "6" (Suburban) "7" (Highway) "8" (Open sky) "9" (Light woodland)</pre>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, on vehicle 1, switch to using a land mobile multipath environment - Urban and category mask - Trees:</p> <pre>-, SWITCH_LMM, v1, 6, 5</pre>

5.6.5 FADER (embedded multipath)

Notes:

1) Supported by GSS7700 (with FPGA build 15 onward and GSBA 0082 signal generator cards), GSS7800 and GSS8000 signal generators only.

The System Messages window warns you if your hardware does not support this command.

2) Not real-time, applied when channel starts.

FADER is fixed while channel is on.

To update: send new FADER command, stop and re-start the channel.

3) SimGEN cannot start more than twelve multipaths (regardless of type) at the same time.

Description	Adds up to 4 multipath signals (sub-channels) to each Channel
Format	<pre><timestamp>,FADER,<veh_ant>,<signal>,<svid>,[<multi_path_index>,<freq>,<LOS_level_offset>,<sc0_lev_ofs>,<sc0_delay_ofs>,<sc0_phase_ofs>,<sc1_lev_ofs>,<sc1_delay_ofs>,<sc1_phase_ofs>,<sc2_lev_ofs>,<sc2_delay_ofs>,<sc2_phase_ofs>,<sc3_lev_ofs>,<sc3_delay_ofs>,<sc3_phase_ofs></pre>
Where	<pre>timestamp ::= see section 5.1.1 veh_ant ::= <vehicle_id>"_"<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 signal ::= "EGNOS" "GALILEO" "GLONASS" "GROUND TX" "GPS" "MSAS" "WAAS" svid ::= satellite svid multi_path_index ::= "ALL" "0" "1-16" Where: ALL = all multipaths, including the LOS 0 = for the incident signal only 1-16 = 1 for multipath echo 1, 2 for multipath echo 2 and so on, to 16 freq ::= "0" (L1) "1" (E6) "2" (E5) LOS_level_offset ::= Level offset of LOS in dB. Range: 0 to 99 dB. Resolution: 0.01 dB. Use 99 dB to turn off LOS sc0_lev_ofs ::= Level offset for Sub-channel 0 in dB. Range: 0 to 99 dB. Resolution: 0.01 dB. Use 99 dB to turn off Sub-channel sc0_delay_ofs ::= Code delay for Sub-channel 0 in ns. Range: 0 to 4158 ns. Resolution: 1 ns.</pre>

Where (continued)	sc0_phase_ofs	::= Carrier phase lag for Sub-channel 0, degrees. Range: 0 to 360 degrees. Resolution: 0.01 degree.
	sc1_lev_ofs	::= Level offset for Sub-channel 1. See sc0_lev_ofs for details
	sc1_delay_ofs	::= Delay offset for Sub-channel 1. See sc0_delay_ofs for details
	sc1_phase_ofs	::= Carrier phase lag for Sub-channel 1. See sc0_phase_ofs for details
	sc2_lev_ofs	::= Level offset for Sub-channel 2. See sc0_lev_ofs for details
	sc2_delay_ofs	::= Code delay for Sub-channel 2. See sc0_delay_ofs for details
	sc2_phase_ofs	::= Carrier phase lag for Sub-channel 2. See sc0_phase_ofs for details
	sc3_lev_ofs	::= Level offset for Sub-channel 3. See sc0_lev_ofs for details
	sc3_delay_ofs	::= Code delay for Sub-channel 3. See sc0_delay_ofs for details
	sc3_phase_ofs	::= Carrier phase lag for Sub-channel 3. See sc0_phase_ofs for details
Returns	status see returned response format, section 5.1.2	
Example	<p>At start of scenario, set Fader for vehicle 1, antenna 1, on Galileo SVID 1, for all multipaths, using frequency L1, LOS level offset = 10 dB, sc0 set to Off, sc 1 to 3 set as shown:</p> <pre>00:00:00,FADER,V1_A1,GALILEO,1,ALL,0,10.00,99,2,3.00,4,5,6. 00,7,8,9.00,10,11,12.00</pre>	

5.6.6 CODE_LEV_OFFSET

Note: Currently SimGEN can only apply this command to GPS satellites

Description	Applies signal level control to each component (C/A, P, Y) at each frequency
Format	<timestamp>,CODE_LEV_OFFSET,<network>,<svid>,<all_flag>,<band>,<code>,<offset>
Where	<p>timestamp ::= see section 5.1.1</p> <p>network ::= "GPS"</p> <p>svid ::= satellite svid</p> <p>all_flag ::= "0" (apply to specified channel / SVID) "1" (apply to all channels / SVIDs)</p> <p>band ::= "L1" "L2" "L5"</p> <p>code ::= "P" "C" "CA" "M"</p> <p>The allowed <band>,<code> pairs are given below:</p> <p style="padding-left: 40px;">"L1","P" "L1","CA" "L2","P" "L2","CA" "L2","C" "L1","M" "L2","M"</p> <p>offset ::= level offset from nominal in dB. This is not the global offset Range: -16 to +6 dB</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, apply code level offset of -10 dB to the L1,P signal transmitted on the specified GPS satellite SVID 1:</p> <pre>- ,CODE_LEV_OFFSET,GPS,1,0,L1,P,-10</pre>

5.7 Pseudorange commands

Command	Description
Pseudorange commands	
PR_RAMP	Pseudorange ramp
PR_ERRORS	Pseudorange errors

5.7.1 SVID rule

SimGEN applies SVID settings only while the SVID is visible. SimGEN does not retain non-default SVID settings after the SVID sets.

5.7.2 PR_RAMP

Description	Allows the user to ramp the pseudorange up or down. Note: <i>PR_RAMP is applied to the satellite and affects all antennas seeing that satellite.</i>
Format	<code><timestamp>,PR_RAMP,<tx_net>,<svid>,<all_flag>,<start_time>,<use_start>,<state>,<offset>,<up_time>,<hold_time>,<down_time></code>
Where	<p><code>timestamp</code> ::= see section 5.1.1</p> <p><code>tx_net</code> ::= "tn_GALILEO" "tn_GLONASS" "tn_GPS" "tn_EGNOS" "tn_MSAS" "tn_WAAS"</p> <p><code>svid</code> ::= satellite svid</p> <p><code>all_flag</code> ::= "0" (apply to specified channel / SVID) "1" (apply to all channels / SVIDs)</p> <p><code>start_time</code> ::= seconds into run</p> <p><code>use_start</code> ::= "0" (don't use start time) "1" (use start time)</p> <p><code>state</code> ::= "0" (retains all previous ramp conditions) "1" (resets all previous ramp conditions)</p> <p><code>offset</code> ::= pseudorange change (m)</p> <p><code>up_time</code> ::= duration of initial ramp section, seconds</p> <p><code>hold_time</code> ::= duration of hold section, seconds</p> <p><code>down_time</code> ::= duration of end ramp section, seconds. The ramp then returns to its previous value</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, ramp up GPS SVID 1. Start the new ramp at 5 s into the run, offset by 100 m. Complete the ramp to the new value after 10 s duration, hold the ramp at the value for a duration of 30 s and return to the previous value after 50 s duration.</p> <pre>-,PR_RAMP,tn_GPS,1,0,5,1,1,100,10,30,50</pre>

5.7.3 PR_ERRORS

Note: Turning off iono errors using *PR_ERRORS* will not affect the ionospheric parameters transmitted in the nav data message.

Description	Allows you to set pseudorange errors.
Format	<timestamp>, PR_ERRORS, <network>, <svid>, <all_flag>, <error_type>, <state>
Where	<p>timestamp ::= see section 5.1.1</p> <p>network ::= "EGNOS" "GALILEO" "GLONASS" "GROUND TX" "GPS" "MSAS" "WAAS"</p> <p>svid ::= satellite svid</p> <p>all_flag ::= "0" (apply to specified channel / SVID) "1" (apply to all channels / SVIDs)</p> <p>error_type ::= "relativity" "clock_errs" (delta clock errors - delta_Af0, delta_Af1; not declared clock errors Af0, Af1) "iscn" "track" "iono" "tropo" "all"</p> <p>state ::= "0" (do not apply error specified in scenario) "1" (apply error specified in scenario)</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, apply relativity and clock errors specified in the scenario to SVID 1 on the GPS network</p> <pre>-, PR_ERRORS, GPS, 1, 0, relativity, clock_errs, 1</pre>

5.8 Data Streaming commands

Command	Description
Data Streaming commands	
DS_ENABLE	Enable or disables Data Streaming
DS_IP	Specifies the Data Streaming I.P. address
DS_RATE	Specifies the Data Streaming update rate
DS_STATUS	Enables the transmission of status messages
DS_SYNC	Enables the transmission of time synchronisation messages
DS_VEH_MOT	Enables the transmission of vehicle motion messages
DS_VEH_CMS	Enable transmission of car motion sensor data messages.
DS_ANT_MOT	Enables the transmission of antenna motion messages
DS_ANT_SIG	Enables the transmission of signal data messages
DS_INFO	Returns data streaming capabilities of the running SimGEN

These commands allow remote control of Data Streaming. They override the settings in the Data Streaming definition file, see section 8.2.

All these commands except DS_ENABLE can be sent before or during a run. You can only send DS_ENABLE before a run has started. Similarly, all changes you have made are lost on rewinding a scenario, except for those made by DS_ENABLE.

5.8.1 DS_ENABLE

Description	Enables or disables Data Streaming.
Format	DS_ENABLE,<enable>
Where	enable ::= "0" (disable data streaming) "1" (enable data streaming)
Returns	status see returned response format, section 5.1.2
Example	Enable data streaming DS_ENABLE,1

5.8.2 DS_IP

Description	Specifies the Data Streaming I.P. address.
Format	DS_IP,<address>
Where	address ::= "BROADCAST" (sets IP address to 255.255.255.255) "LOOPBACK" (sets IP address to 127.0.0.1) specific numeric IP address
Returns	status see returned response format, section 5.1.2
Example(s)	Set the IP address to broadcast (255.255.255.255) DS_IP,BROADCAST Set the IP address to 12.24.1.255 DS_IP,12.24.1.255

5.8.3 DS_RATE

Description	Specifies the Data Streaming update rate.
Format	DS_RATE,<rate>
Where	<rate> ::= update rate in ms "10" "20" "40" "50" "100" "500" "1000"
Returns	status see returned response format, section 5.1.2
Example	Set the update rate to 100 ms DS_RATE,100

5.8.4 DS_STATUS

Description	Enable transmission of status messages.
Format	DS_STATUS,<enable>
Where	<enable> ::= "0" (disable status messages) "1" (enable status messages)
Returns	status see returned response format, section 5.1.2
Example	Enable transmission of status messages DS_STATUS,1

5.8.5 DS_SYNC

Description	Enables transmission of time synchronisation messages.
Format	DS_SYNC,<enable>
Where	<enable> ::= "0" (disable time sync messages) "1" (enable time sync messages)
Returns	status see returned response format, section 5.1.2
Example	Disable transmission of time synchronisation messages DS_SYNC,0

5.8.6 DS_VEH_MOT

Description	Enables transmission of vehicle motion messages.
Format	DS_VEH_MOT,<vehicle_id>,<enable>[,<egi>]
Where	vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 <enable> ::= "0" (disable vehicle motion messages) "1" (enable vehicle motion messages) <egi> ::= "0" ((default) normal vehicle motion) "1" (EGI motion)
Returns	status see returned response format, section 5.1.2
Example(s)	Enable motion messages for vehicle 1 DS_VEH_MOT,v1,1 Same as above, EGI parameter explicit DS_VEH_MOT,v1,1,0 Enable EGI motion messages for vehicle 1 DS_VEH_MOT,v1,1,1 Disable EGI motion messages for vehicle 1 DS_VEH_MOT,v1,0,1

5.8.7 DS_VEH_CMS

Description	Enables transmission of car motion sensor data messages.
Format	DS_VEH_CMS,<vehicle_id>,<enable>
Where	vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 <enable> ::= "0" (disable vehicle motion messages) "1" (enable vehicle motion messages)
Returns	status see returned response format, section 5.1.2
Example(s)	Enable motion sensor messages for vehicle 1 DS_VEH_CMS,v1,1

5.8.8 DS_ANT_MOT

Description	Enables transmission of antenna motion messages.
Format	DS_ANT_MOT,<veh_ant>,<enable>
Where	veh_ant ::= <vehicle_id>" "<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 enable ::= "0" (disable antenna motion messages) "1" (enable antenna motion messages)
Returns	status see returned response format, section 5.1.2
Example(s)	Enable antenna motion messages for the antenna 1 on vehicle 1: DS_ANT_MOT,v1_a1,1 Disable antenna motion messages for antenna 3 on vehicle 2: DS_ANT_MOT,v2_a3,0

5.8.9 DS_ANT_SIG

Description	Enable transmission of signal data messages.
Format	DS_ANT_SIG,<veh_ant>,<signal>,<enable>
Where	veh_ant ::= <vehicle_id>" "<antenna_id> vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 antenna_id ::= "a"<a_number> a_number ::= antenna number starting from 1 <signal> ::= "EGNOS" "GALILEO" "GLONASS" "GROUND TX" "GPS" "MSAS" "WAAS" <enable> ::= "0" (disable signal data messages) "1" (enable signal data messages)
Returns	status see returned response format, section 5.1.2
Example	Enable signal data for GPS, vehicle 1, antenna 2 DS_ANT_SIG,v1_a2,GPS,1

5.8.10 DS_INFO

Description	Returns the data streaming capabilities of the running SimGEN
Format	DS_INFO
Return syntax	<a>,,<c>,<d>
Returns	status see returned response format, section 5.1.2 together with: a ::= data streaming version b ::= data streaming port c ::= maximum size of datagram packet d ::= id of sync message type

Example | `<data> 12,15660,408,29 </data>`

5.9

5.10 Motion commands (trajectory delivery)

Command	Description
Motion commands	
MOT	Vehicle motion using ECEF axes with respect to WGS84
MOTB	Vehicle motion using Lat/Long/Height with respect to WGS84
AIDING_OFFSET	Sets the aiding offset for a given vehicle

5.10.1 MOT

Description	<p>This command defines the vehicle motion at the specified time.</p> <p>Vehicle motion is defined with respect to the WGS84 ECEF frame (X,Y,Z). Linear motion (such as position and velocity) is resolved in this frame.</p> <p>Rotational data is expressed in terms of the vehicle body axes (x,y,z), where x is along the vehicle longitudinal axis pointing towards the nose, y is along the lateral axis pointing right and z completes the right-handed set (nominally down).</p> <p>Attitude is represented as the set of Euler angle rotations obtained on moving from alignment with the local geographic frame (North, East, Down) in the sequence heading (about z), elevation (about y), and bank (about x).</p> <p>Angular motion is about the body axes, see section 5.9 for details.</p> <p>Units are metres, radians and seconds as appropriate.</p> <p>Note: The first MOT position message determines the motion of the vehicle at the start of the scenario, even if the timestamp of the message does not match the scenario start time. Subsequent messages will be used correspondingly.</p>
Format	<pre> <timestamp>,MOT,<veh_mot>, <x>,<y>,<z>, <vx>,<vy>,<vz>, <ax>,<ay>,<az>, <jx>,<jy>,<jz>, <h>,<e>,, <ωx>,<ωy>,<ωz>, <ω̇x>,<ω̇y>,<ω̇z>, <ω̈x>,<ω̈y>,<ω̈z> </pre> <p>Note: Parameters may be omitted from the end of the list, in which case they will be taken as a zero.</p>

Where	<p>timestamp ::= see section 5.1.1</p> <p>veh_mot ::= vehicle_id"_"motion_id</p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p>motion_id ::= "m"<m_number></p> <p>m_number ::= "1" (motion model number always 1)</p> <p>x,y,z ::= X,Y,Z position - metres</p> <p>vx,vy,vz ::= velocity in X,Y,Z axes - m/s</p> <p>ax,ay,az ::= acceleration in X,Y,Z axes - m/s²</p> <p>jx,jy,jz ::= jerk in X,Y,Z axes - m/s³</p> <p>h ::= heading - radians, range +/- π</p> <p>e ::= elevation - radians, range +/- $\pi/2$</p> <p>b ::= bank , radians - range +/- π</p> <p>$\omega_x, \omega_y, \omega_z$::= angular velocity about X,Y,Z body axes - rad/s</p> <p>$\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z$::= angular acceleration about X,Y,Z body axes - rad/s²</p> <p>$\ddot{\omega}_x, \ddot{\omega}_y, \ddot{\omega}_z$::= angular jerk about X,Y,Z body axes - rad/s³</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, define motion of vehicle 1, motion model 1, to be:</p> <p>(x,y,z) = (4058735.385,-251461.3003,4897339.144) metres;</p> <p>(vx, vy, vz) = (1.363,2.819,-0.035) m.s⁻¹;</p> <p>(ax, ay, az) = (0.512,-0.467,-0.624) m.s⁻²;</p> <p>(jx, jy, jz) = (0,0,0) m.s⁻³;</p> <p>-,MOT,v1_m1,4058735.385,-251461.3003,4897339.144,1.363,2.819,-0.035,0.512,-0.467,-0.624,0,0,0</p>

5.10.2 MOTB

Description	<p>This command defines the vehicle motion at the specified time; but with the vehicle position specified in Latitude, Longitude and Height. Linear motion is with respect to local geographic frame (North, East, Down)</p> <p>Note: <i>The first MOTB position message determines the motion of the vehicle at the start of the scenario, even if the timestamp of the message does not match the scenario start time. Subsequent messages will be used correspondingly.</i></p>
Format	<pre><timestamp>,MOTB,<veh_mot>, <lat>,<long>,<height>, <vel_n>,<vel_e>,<vel_d>, <acc_n>,<acc_e>,<acc_d>, <jerk_n>,<jerk_e>,<jerk_d>, <h>,<e>,, <ωx>,<ωy>,<ωz>, <ḡx>,<ḡy>,<ḡz>, <ḡ_{xyz <p>Note: <i>Parameters may be omitted from the end of the list, in which case they will be taken as a zero.</i></p>}</pre>
Where	<pre>timestamp ::= (see general notes on commands) veh_mot ::= vehicle_id"_"motion_id vehicle_id ::= "v"<v_number> v_number ::= vehicle number starting from 1 motion_id ::= "m"<m_number> m_number ::= "1" (motion model number always 1) lat ::= latitude - radians, range +/- π/2 long ::= longitude - radians, range +/- π height ::= height, m vel_n ::= velocity, Northward, m/s vel_e ::= velocity, Eastward, m/s vel_d ::= velocity, downward, m/s acc_n ::= acceleration, Northward, m/s² acc_e ::= acceleration, Eastward, m/s² acc_d ::= acceleration, downward, m/s² jerk_n ::= jerk, Northward, m/s³ jerk_e ::= jerk, Eastward, m/s³ jerk_d ::= jerk, downward, m/s³ h ::= heading - radians, range +/- (π/2 e ::= elevation - radians, range +/- (π/2 b ::= bank - radians, range +/- (π/2 (x,(y,(z ::= angular vel about X,Y,Z body axes - rad/s ḡx,ḡy,ḡz ::= angular acc about X,Y,Z body axes, - rad/s² ḡ_x, ḡ_y, ḡ_z ::= angular jerk about X,Y,Z body axes - rad/s³</pre>
Returns	status see returned response format, section 5.1.2

Example	<p>At 1 second into scenario use MOTB to set vehicle 1, motion model 1, to:</p> <p>(lat, long, height) = (1.4561541234,-2.6716575678,100) rad, rad, m; (vel_n, vel_e, vel_d) = (1.591,3.529,0.176) m/s; (acc_n, acc_e, acc_d) = (1.079,2.316,0.027) m/s²; (jer_n, jer_e, jer_d) = (0,0,0) m/s³;</p> <p>00:00:01,MOTB,v1_m1,1.456154123,-2.671657456,100, 1.591,3.529,0.176,1.079,2.316,0.027,0,0,0</p>
----------------	--

5.10.3 AIDING_OFFSET

Description	Sets the aiding offsets on the given vehicle
Format	<timestamp>,AIDING_OFFSET,<veh_aid>,<x>,<y>,<z>
Where	<p>timestamp ::= see section 5.1.1</p> <p><veh_aid> ::= "vehicle_id"_"aiding_id"</p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p><aiding_id> ::= "d"<d_number></p> <p>d_number ::= aiding offset number starting from 1</p> <p><x> ::= x offset, range -500 to +500 m</p> <p><y> ::= y offset, range -500 to +500 m</p> <p><z> ::= z offset, range -500 to +500 m</p>
Returns	status see returned response format, section 5.1.2
Example	<p>Now, set aiding offset for vehicle 1 aiding offset 1, to x = 100 m, y = 100 m, z = 300 m</p> <p>-,AIDING_OFFSET,v1_d1,100,100,300</p>

5.11 Vehicle body axes

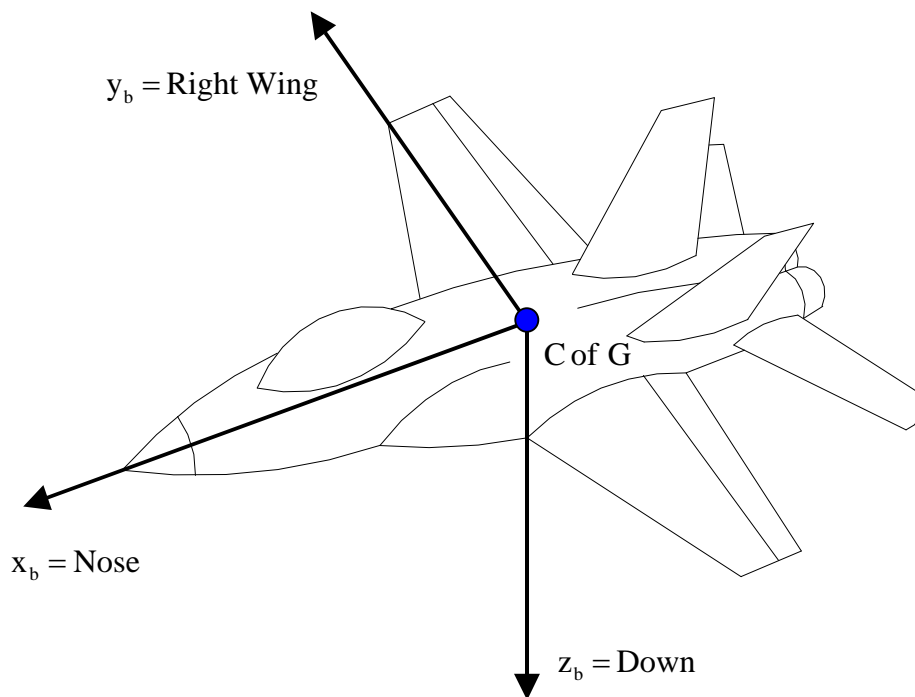
Vehicle body axes are defined at the vehicle Centre of Gravity (C of G). Table 5-22 describes the aircraft body axes (x_b , y_b , z_b).

Table 5-23 Definition of aircraft body axes

Axis	Description
x_b	Pointing towards the nose (the nominal direction of travel)
y_b	(Orthogonal to x_b) - pointing towards the starboard side (right wing)
z_b	Nominally down

Figure 5-2 shows the axes of an aircraft.

Figure 5-2 Vehicle Body Axes



Similar definitions apply to other vehicles.

5.11.1 Vehicle body attitude

Vehicle body attitude is defined with respect to the Local Geodetic frame.

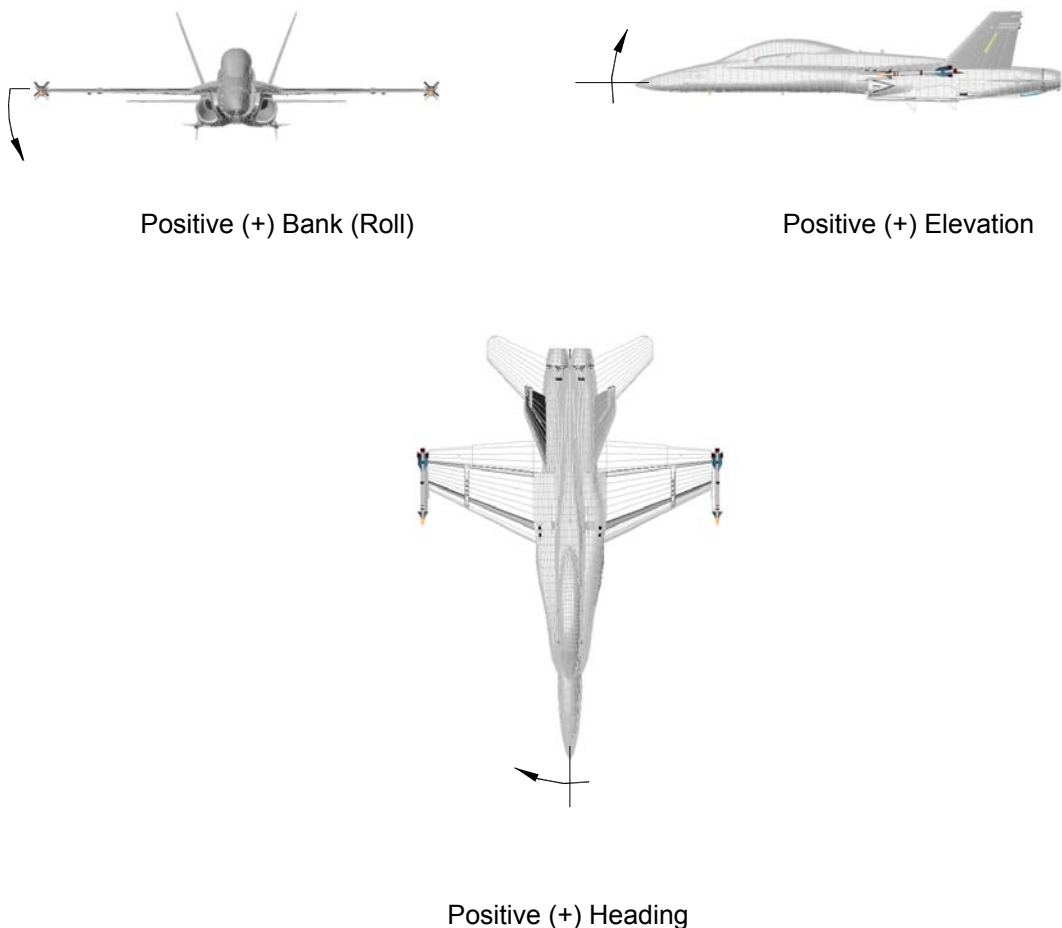
Vehicle body attitude is a series of angular rotations in moving the body axes from alignment with the Local Geodetic reference frame (North, East, Down) to the current vehicle orientation. Table 5-23 shows the sequence of generation of these angles (Euler angles).

Table 5-24 Definition of body axis angles

Sequence	Local Geodetic parameter	Description of rotation
First	Heading (ψ)	About the vehicle z_b axis
Second	Elevation (θ)	About the new direction of the y_b axis
Third	Bank (ϕ)	About the new direction of the x_b axis

Figure 5-3 shows the attitudes of an aircraft.

Figure 5-3: Attitude orientation in relation to body axes



5.12 Hardware and calibration commands

Note: Some Hardware and calibration commands are for Spirent use only.

Command	Description
Hardware and Calibration commands	
RFOF	Returns the serial number and rear connector offset setting.
CAL	Turns on calibration mode.
CAL_LEVEL	Adjusts the output level at the front panel.
NBLK2	Specifies an alternative hardware signal control block
PCAL	Performs self-calibration
FW_CMD	Use to pass commands to firmware
*IDN?	Gets ASCII equipment identification
LOCAL_LOCKOUT	Disables SimGEN user interface

5.12.1 RFOF

Description	<p>This command returns: 1) the signal generator serial number; and 2) the difference in signal level at the front panel and rear panel RF connectors.</p> <p>This command is for small boxes, GSS6700, GSS7xxx and GSS8000-series signal generators only.</p> <p>With GSS7xxx and GSS8000-series signal generators, this command gives a return of up to four values, one for each signal type in use, in the same order as they appear on the signal generator display..</p> <p>With GSS6700 signal generators only, you need to provide the constellation type.</p>
Format	RFOF, [<constellation_type>]
Where	<constellation_type> ::= "GPS" "GLONASS" "GALILEO" use with GSS6700 only
Returns	status see returned response format, section 5.1.2 data (a repeating list of serial numbers, followed by the signal level difference in dB typically around 60 dB) for example: <pre><msg> <status> 2 <\status> <data>1003,60.12<\data> <\msg></pre>
Example	RFOF

5.12.2 CAL

Description	Sets a big box signal generator into power-level calibration mode. Note: Only use in "Ready-to-run" mode
Format	CAL,<state>
Where	state ::= "ON" "OFF"
Returns	status see returned response format, section 5.1.2
Example(s)	CAL,ON CAL,OFF

5.12.3 CAL_LEVEL

Description	Set the level at the front panel connector to the value you specify - for big boxes only. Note: Only valid after CAL,ON
Format	CAL_LEVEL,<level>
Where	Level ::= level in dBx10 with respect to the nominal GPS power level For example: 11.5 dB is a Level of 115
Returns	status see returned response format, section 5.1.2
Example	Set the output level to -15 dB with respect to the nominal GPS level: CAL_LEVEL,-150

5.12.4 NBLK2

Note. This command only applies to GSS8000-series and GSS7xxx signal generators when you use a simulation iteration rate of 4 or 10 ms. This command applies antenna gain patterns in software instead of in the signal generator.
Do not use this command if you want to use antenna phase pattern offsets.

Description	Use this command to use an alternative hardware signal control block (NBLK) that gives improved resolution of code and carrier offsets, and level offset, when using a simulation iteration rate of 4 or 10 ms. In this mode, offsets you specify with the MOD command are applied every 4, 5, or 10 ms. In the default (NBLK) mode, MOD command signal offsets are applied every 100 ms. You must send NBLK2 before the run starts, for every run. For big boxes only.
Format	NBLK2
Returns	status see returned response format, section 5.1.2
Example	NBLK2

5.12.5 PCAL

Notes:

- 1) Only use PCAL when instructed by a Spirent Applications Support Engineer
- 2) This command only applies to GSS8000-series and GSS7xxx signal generators.

Description	<p>Performs self-calibration routines that take several minutes to complete. PCAL displays a progress bar which counts up to 30 seconds and resets. The progress bar disappears when PCAL is complete.</p> <p>Note: Do not use the signal generator while it is performing a PCAL</p>
Format	PCAL, <address>, <param>
Where	<p>address ::= "IEEE hardware address" "IP address" ::= "0" for all units defined in the hardware configuration</p> <p>param ::= provided by Spirent Customer Services</p>
Returns	<p>status see returned response format, section 5.1.2</p> <p>The return includes PCAL?<parameter></p> <p><parameter> is a 4 digit hexadecimal number. Each binary bit represents a channel (CHAN0 is LSB). A 1 bit represents a failed channel. A return of FFFF means something common to all channels failed. If CPUS=7 the combined (ORed) status is returned.</p> <p>PCAL?<parameter> other than PCAL?0000 means there is a PCAL problem.</p>
Example	<p>To perform the standard self check and channel alignment procedure using parameter \$0 (not a real parameter):</p> <pre>PCAL, 0, \$0</pre>

5.12.5.1 PCAL over IEEE

The PCAL command was written for TCP/IP, where a response always follows a request. IEEE is not a request / response based system. If you use the PCAL command over the IEEE bus you must either:

Send the PCAL command and wait for five minutes (without issuing any IEEE commands) which is long enough for the PCAL operation to complete. You will not read a response from the signal generator until the PCAL has completed.

When the PCAL command has completed, the PCAL? command is sent with a response as detailed in the table above.

Or:

Send the PCAL command and try reading from the signal generator every 30 seconds. When you are able to read from the signal generator, within five minutes of issuing the PCAL command, the PCAL is complete and you will see a response to the PCAL? Command, as detailed in the table above.

5.12.6 FW_CMD prefix

Notes:

- 1) Do not use this command while a scenario is running.
- 2) This command only applies to GSS8000-series and GSS7xxx signal generators.

Use the FW_CMD prefix with the *IDN? command in section 5.12.7 to get ASCII equipment identification from the processor (processor #2) in a big box signal generator.

Description	Use to pass the FW_CMD command to GSS7xxx or GSS8000-series signal generators
Format	FW_CMD<address>,<response>,<command>
Where	<p>address ::= IEEE hardware address IP address ::= "0" for all units defined in the hardware configuration</p> <p>response ::= "0" (no response from command expected) "1" (response from command expected)</p> <p>command ::= in section 5.12.7</p>
Returns	Status see returned response format, section 5.1.2 and see section 5.12.7
Example	<p>To send the *IDN? command to a signal generator with IP address 192.168.5.100, with expected response from the signal generator:</p> <pre>FW_CMD,192.168.5.100,1,*IDN?</pre>

5.12.7 *IDN?

Description	<p>Use this command on its own to get the SimGEN version number or with the FW_CMD prefix, section 5.12.6, to get ASCII equipment identification from the processor (processor #2). This combination only works with big boxes.</p> <p>Also see section 5.2.25 (these two *IDN? commands are NOT identical).</p>
Format	FW_CMD,0,1,*IDN?
Return syntax	<made_by>, <model> <op_mode> <flags>, <serial_num>,<fw_level>
Returns	<p>made_by ::= ASCII string "Spirent-GSS"</p> <p>model ::= ASCII string "GSS number"</p> <p>op_mode ::= ASCII string giving RF output mode (fixed):L1P1.L2P2:L1+L2 output units;</p> <p>flags ::= an eight-digit hexadecimal number. Each bit of this number may have a meaning. Unused bits return as zero</p> <p>serial_num ::= an ASCII string defining the signal generator serial number.</p> <p>fw_level ::= an ASCII string defining the firmware issue number, format nn.nn</p>
Example	Spirent-GSS, GSS7735 L1P1.L2P2 00000011, 8001, 01.22

5.12.8 LOCAL_LOCKOUT

Description	<p>Transfers command of a scenario from the SimGEN interface to using remote commands from a remote control interface.</p> <p>When enabled, LOCAL_LOCKOUT prevents you using SimGEN to access anything that affects a scenario by disabling: New, Start, Stop, Rewind, Power Adjustment (closes an open Power Adjustment dialog on enable), changing scenario source file, Most Recently Used list and exiting SimGEN.</p> <p>You can continue to use the SimGEN interface to read scenario data. However, SimGEN displays a warning if you try to edit files that will affect a running scenario.</p> <p>When enabled, white flashing text "LOCKED" on red background appears in the status bar.</p> <p>This command affects scenarios that are ready to run and running.</p>
Format	<code>LOCAL_LOCKOUT,<flag></code>
Where	<code>flag ::= "0" disabled</code> <code> "1" enabled</code>
Returns	<code>status see returned response format, section 5.1.2</code>

5.13 Data request commands

These commands return the requested simulation data item or items for the time requested. All data items that are available to quick-look and post-processing are available with these commands. The command syntax is identical to that logged in the .qll file created when a set of logged data items are specified. See the SimGEN Software User Manual, reference a). As with logging data from within SimGEN, data requests are on a Vehicle, Antenna, Signal or transmitter basis.

5.13.1 Vehicle Data Request

All the Vehicle data request commands share the same command format

Description	Generic Vehicle data request command
Format	<timestamp>,<command>,<vehicle_id>
Where	<p>timestamp ::= see section 5.1.1</p> <p>command ::= see Vehicle command list</p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p>
Returns	<p>status see returned response format, section 5.1.2</p> <p>data returned value(s)</p> <p>In the case of VEH_ALL the return data is:</p> <p>x,y,z position</p> <p>x,y,z velocity</p> <p>x,y,z acceleration</p> <p>x,y,z jerk</p> <p>h,e,b heading, elevation and bank</p> <p>x,y,z angular velocity</p> <p>x,y,z angular acceleration</p> <p>x,y,z angular jerk</p> <p>for example:</p> <pre><msg> <status> 2 </status> <data> 4063585.582,255362.452,4893118.726, 3.136,49.902,0.000, -0.001,0.000,0.000, -2.6934e-006,-9.59772e-008,-2.1684e018, 1.5708,0,0, 0,0,0, 0,0,0, -7.45526e-010,-5.89907e-016,-7.13709e-016 </data> </msg></pre>
Example	<p>Now, return the Vehicle latitude for vehicle 1</p> <pre>-,VEH_LAT,v1</pre>

Command	Description
Vehicle Data Request commands	
VEH_X_POS	Returns the X position of the vehicle
VEH_Y_POS	Returns the Y position of the vehicle
VEH_Z_POS	Returns the Z position of the vehicle
VEH_X_VEL	Returns the X velocity of the vehicle
VEH_Y_VEL	Returns the Y velocity of the vehicle
VEH_Z_VEL	Returns the Z velocity of the vehicle
VEH_N_VEL	Returns the north velocity of the vehicle
VEH_E_VEL	Returns the east velocity of the vehicle
VEH_D_VEL	Returns the down velocity of the vehicle
VEH_X_ANG_VEL	Returns the x angle velocity of the vehicle
VEH_Y_ANG_VEL	Returns the y angle velocity of the vehicle
VEH_Z_ANG_VEL	Returns the z angle velocity of the vehicle
VEH_X_ACC	Returns the x acceleration of the vehicle
VEH_Y_ACC	Returns the y acceleration of the vehicle
VEH_Z_ACC	Returns the z acceleration of the vehicle
VEH_X_JERK	Returns the x jerk of the vehicle
VEH_Y_JERK	Returns the y jerk of the vehicle
VEH_Z_JERK	Returns the z jerk of the vehicle
VEH_LAT	Returns the latitude of the vehicle
VEH_LONG	Returns the longitude of the vehicle
VEH_HEIGHT	Returns the height of the vehicle (GPS Ellipsoid)
VEH_HEADING	Returns the heading of the vehicle
VEH_ELEVATION	Returns the elevation of the vehicle
VEH_BANK	Returns the bank of the vehicle
VEH_MSL_HEIGHT	Returns the meters above sea level height of the vehicle
VEH_MSL_OFFSET	Returns the offset between MSL_HEIGHT and HEIGHT
VEH_SPEED	Returns the speed of the vehicle
VEH_ALL	Returns all of the above in a list
VEH_GROUND_SPEED	Return the ground speed of the vehicle

5.13.2 Antenna Data Requests

All the antenna data request commands share the same command format

Description	Generic antenna data request command
Format	<timestamp>,<command>,<veh_ant>
Where	<p>timestamp ::= see section 5.1.1</p> <p>command ::= see antenna command list</p> <p>veh_ant ::= <vehicle_id>"_"<antenna_id></p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p>antenna_id ::= "a"<a_number></p> <p>a_number ::= antenna number starting from 1</p>
Returns	<p>status see returned response format, section 5.1.2</p> <p>data returned value(s)</p>
Returns (continued)	<p>In the case of ANT_ALL the return data is:</p> <p>x,y,z position</p> <p>x,y,z velocity</p> <p>x,y,z acceleration x,y,z jerk</p> <p>h,e,b heading, elevation and bank</p> <p>x,y,z angular velocity</p> <p>x,y,z angular acceleration</p> <p>x,y,z angular jerk</p> <p>for example:</p> <pre><msg> <status> 2 </status> <data> 4063585.582,255362.452,4893118.726, 3.136,49.902,0.000, -0.001,0.000,0.000, -2.6934e-006,-9.59772e-008,-2.1684e018, 1.5708,0,0, 0,0,0, 0,0,0, -7.45526e-010,-5.89907e-016,-7.13709e-016 </data> </msg></pre>
Example	<p>Now, return all the antenna details for vehicle 1, antenna 1</p> <pre>-,ANT_ALL,v1_a1</pre>

Command	Description
Antenna Data Request commands	
ANT_X_POS	Returns the x position of the antenna
ANT_Y_POS	Returns the y position of the antenna
ANT_Z_POS	Returns the z position of the antenna
ANT_X_VEL	Returns the x velocity of the antenna
ANT_Y_VEL	Returns the y velocity of the antenna
ANT_Z_VEL	Returns the z velocity of the antenna
ANT_X_ACC	Returns the x acceleration of the antenna
ANT_Y_ACC	Returns the y acceleration of the antenna
ANT_Z_ACC	Returns the z acceleration of the antenna
ANT_DOP	Returns the DOP of the antenna
DOP_TYPE	Returns the DOP type of the antenna
ANT_LAT	Returns the latitude of the antenna
ANT_LONG	Returns the longitude of the antenna
ANT_HEIGHT	Returns the height of the antenna
ANT_ALL	Returns all position information
ANT_GROUND_SPEED	Returns the ground speed of the antenna
ANT_TERR_OBS_ANGLE	Returns the angle below which satellites will be obscured (rad)
ANT_TERR_OBS_DISTANCE	Returns the distance from the antenna to the obscuring obstacle
ANT_TERR_OBS_HEIGHT	Returns the height used to calculate the obscuration angle (either height above antenna if height is relative to vehicle's height is selected, otherwise it is just obstacle height)

5.13.3 Signal Data Requests

All the signal data request commands share the same command format:

Description	Generic signal data request command
Format	<timestamp>,<command>,<veh_ant>,<type>,<id>,<mode>,<multi_index>,<freq>
Where	<p>timestamp ::= see section 5.1.1</p> <p>command ::= see Signal command list</p> <p>veh_ant ::= <vehicle_id>"_ "<antenna_id></p> <p>vehicle_id ::= "v"<v_number></p> <p>v_number ::= vehicle number starting from 1</p> <p>antenna_id ::= "a"<a_number></p> <p>a_number ::= antenna number starting from 1</p> <p>type ::= "0" (GPS) "1" (SBAS) "2" (GLONASS)</p> <p>id ::= 0-n (Channel number) 1-n (Satellite SVID)</p> <p>mode ::= "0" (svid mode, use id = 1-n) "1" (channel mode, use id = 0-n)</p> <p>muti_index ::= "0" (incident signal) (number of the multipath starting at 1 - only applies in SVID mode)</p> <p>freq ::= see Table 5-24</p>

Table 5-25 Signal Data Request - <type> and <freq> parameters

freq	GPS	SBAS	GLONASS
0	L1	L1	L1
1	L2	not applicable	L2
2	L5	L5	not applicable

Returns

status see returned response format, section 5.1.2

data returned values

For SIG_ALL the data returned is:

channel id,
transmitter network type,
multipath index,
elevation, azimuth,
x,y,z,
tropo delay, iono delay,
pseudorange, pseudorange rate, range,
signal level

For example:

```
<msg>
  <status> 2 </status>
  <data>
    2,
    GPS,
    0,
    0.4200212061,2.6974735260,
    25461645.393,7561165.092,23899.991,
    5.962,2.992,
    23305565.201,664.249,23305556.247,
    10.88
  </data>
</msg>
```

For SIG_TXID the data returned is set to -1 if the channel or SVID is unallocated.

For SIG_MP_TYPE the data return format is:

```
<mpth_type>,<echo_no>,<channel>,<parameter_1>-
<parameter_n>
```

Section 5.6.2 gives details of the parameters

mpth_type, echo_no, data set, channel, parameter_1 to parameter_n

Examples

Now, return all signal data for vehicle 1, antenna 1, on GPS, channel 0, channel mode, for the incident signal on frequency L1:

```
-,SIG_ALL,v1_a1,0,0,1,0,0
```

Now, return multipath settings for SVID 23, echo number 3 on the GPS constellation, frequency L1:

```
-,SIG_MP_TYPE,v1_A1,0,23,0,3,0
```

Example data returns for SIG_MP_TYPE:

Note: The SIG_MP_TYPE data returns are given in the order displayed in the **Channel Assignment-State-Multipath-Manual multipath settings** dialog.

```
<data> 2,2,-1,0.5000,0.1000,-19.2,0.0000,1,0 </data>
```

Examples

(cont)

Where:

2	mpth_type	Doppler offset
2	echo_no	Second multipath or echo
-1	channel	Channel not forced
0.5000	parameter_1	Initial delay, C/A chips
0.1000	parameter_2	Doppler offset, Hz
-19.2	parameter_3	Attenuation, dB
0.0000	parameter_4	If parameter_5 = 1 then parameter_4 = 0.000 is defined to be randomized carrier phase. If parameter_5 = 0 then parameter_4 is carrier phase (degrees)
1	parameter_5	Initial phase - random (Initial phase-Random selected in the dialog)
0	parameter_6	Units for parameter_1 = Initial delay, C/A chips Units for parameter_2 = Doppler offset, Hz If parameter_6 = 1, then units for parameter_1 = initial delay, m and units for parameter_2 = pseudorange rate, m.s ⁻¹

<data> 1,3,5,14.6,900.1230 </data>

Where:

1	mpth_type	Fixed offset
3	echo_no	Third multipath
5	channel	Forced on channel 5, numbering from 0
14.6	parameter_1	Attenuation, dB
900.1230	parameter_2	Range offset, m

Command	Description
Signal Data Request commands	
SIG_X_POSN	Returns the x position of the satellite when signal sent
SIG_Y_POSN	Returns the y position of the satellite when signal sent
SIG_Z_POSN	Returns the z position of the satellite when signal sent
SIG_PRANGE	Returns the pseudo range of the signal (at time T the distance between the antenna and the satellite at time T-1)
SIG_TXID	Returns the SVID of the signals transmitter
SIG_ELEV	Returns the elevation of the signal
SIG_AZIM	Returns the azimuth of the signal
SIG_RANGE	Returns the range of the signal (at time T the distance between the satellite and the antenna)
SIG_LEVEL	Returns the signal level
SIG_TROPO_DELAY	Returns the troposphere delay of the signal
SIG_IONO_DELAY	Returns the ionosphere delay of the signal
SIG_PR_RATE	Returns the pseudorange rate of the signal
SIG_PRR_RATE	Returns the pseudorange and pseudorange rate

Command	Description
SIG_PRANGE_ERROR	Returns the pseudorange error
SIG_MPI	Returns the multipath index of a satellite (note this command should only be directed to a Channel)
SIG_ALL	Returns all
SIG_MP_TYPE	Returns multipath settings

5.13.4 Transmitter Data Requests

Description	Generic transmitter data request command
Format	<timestamp>,<command>,<network>,<svid>
Where	<p>timestamp ::= see section 5.1.1</p> <p>command ::= see Transmitter command list</p> <p>network ::= <gps> <sbas> <glonass></p> <p>gps ::= "tn_GPS"</p> <p>sbas ::= "tn_SBAS"</p> <p>glonass ::= "tn_GLONASS"</p> <p>svid ::= satellite svid</p>
Returns	<p>status see returned response format, section 5.1.2</p> <p>data returned values</p> <p>In the case of TX_POS the returned data is: x,y,z (in ECEF format)</p> <p>For example:</p> <pre><msg> <status> 2 </status> <data> 25461604.397,7561303.363,-23829.350 </data> </msg></pre>
Example	<p>Now, return the transmitter ECEF position for GPS SVID 1</p> <pre>-,TX_POS,tn_GPS,1</pre>

Command	Description
Transmitter Data Request commands	
TX_ECEF_X	Returns the x position of the transmitter
TX_ECEF_Y	Returns the y position of the transmitter
TX_ECEF_Z	Returns the z position of the transmitter
TX_ECEF_X_VEL	Returns the velocity of the transmitter in the x axes
TX_ECEF_Y_VEL	Returns the velocity of the transmitter in the y axes
TX_ECEF_Z_VEL	Returns the velocity of the transmitter in the z axes
TX_LAT	Returns the latitude of the transmitter
TX_LONG	Returns the longitude of the transmitter
TX_HEIGHT	Returns the height of the transmitter
TX_POS	Returns the ECEF for X,Y,Z
TX_SVID	Returns the SVID of the transmitter
TX_SIG_ID	Returns the signal id of the transmitter, such as GPS or SBAS

5.14 Navigation commands

Command	Description
Navigation commands	
GPS_NAV_DATA_ERR	Remote input and insertion into action queue of GPS nav data errors already scheduled in current scenario
GPS_NAV_DATA_MOD	Remote input and insertion into action queue of GPS nav data modifications already scheduled in current scenario
WAGE	Enables / disables WAGE for the whole constellation.

When you use the following commands:

GPS_NAV_DATA_ERR

GPS_NAV_DATA_MOD

you must ensure the timestamp (if any) you use is earlier than the value of `<start time ms>` you have used.

If you want to use `<start time ms>` equal to zero, first ARM the scenario and then send the command with a timestamp of “-“ (perform now) and `<start time ms>` set to zero. Then send the RUN command to run the scenario.

5.14.1.1 SVID rule

SimGEN applies SVID settings only while the SVID is visible. SimGEN does not retain non-default SVID settings after the SVID sets.

5.14.2 GPS_NAV_DATA_ERR

Description	Remote input and insertion into action queue of GPS nav data errors already scheduled in current scenario																								
Format	<code><timestamp>,GPS_NAV_DATA_ERR,<network>,<svid>, <start_time_ms>,<end_time_ms>,<word-1>,<subframe-1>, <all_svs>,<all_words>,<word3_10>,<all_subframes>, <error type></code>																								
Where	<table> <tr> <td>timestamp</td><td>::= see section 5.1.1</td></tr> <tr> <td>network</td><td>::= "tn_GPS"</td></tr> <tr> <td>svid</td><td>::= satellite svid</td></tr> <tr> <td>start_time_ms</td><td>::= start time, milliseconds</td></tr> <tr> <td>end_time_ms</td><td>::= end time, milliseconds</td></tr> <tr> <td>word-1</td><td>::= required word minus 1</td></tr> <tr> <td>subframe-1</td><td>::= required subframe minus 1</td></tr> <tr> <td>all_svs</td><td>::= "0" (false) "1" (true)</td></tr> <tr> <td>all_words</td><td>::= "0" (false) "1" (true)</td></tr> <tr> <td>word3_10</td><td>::= 3-10(word 3 to 10, as required)</td></tr> <tr> <td>all_subframes</td><td>::= "0" (false) "1" (true)</td></tr> <tr> <td>error type</td><td>::= "set one bit" "set two bits" "set three bits" "set all zero" "set word alternate 0 and 1" "diverge IODC and IODE"</td></tr> </table>	timestamp	::= see section 5.1.1	network	::= "tn_GPS"	svid	::= satellite svid	start_time_ms	::= start time, milliseconds	end_time_ms	::= end time, milliseconds	word-1	::= required word minus 1	subframe-1	::= required subframe minus 1	all_svs	::= "0" (false) "1" (true)	all_words	::= "0" (false) "1" (true)	word3_10	::= 3-10(word 3 to 10, as required)	all_subframes	::= "0" (false) "1" (true)	error type	::= "set one bit" "set two bits" "set three bits" "set all zero" "set word alternate 0 and 1" "diverge IODC and IODE"
timestamp	::= see section 5.1.1																								
network	::= "tn_GPS"																								
svid	::= satellite svid																								
start_time_ms	::= start time, milliseconds																								
end_time_ms	::= end time, milliseconds																								
word-1	::= required word minus 1																								
subframe-1	::= required subframe minus 1																								
all_svs	::= "0" (false) "1" (true)																								
all_words	::= "0" (false) "1" (true)																								
word3_10	::= 3-10(word 3 to 10, as required)																								
all_subframes	::= "0" (false) "1" (true)																								
error type	::= "set one bit" "set two bits" "set three bits" "set all zero" "set word alternate 0 and 1" "diverge IODC and IODE"																								
Returns	status see returned response format, section 5.1.2																								
Example	<p>At start of scenario, set GPS nav data errors on SVID 1. Errors start at 2,400 ms and end at 3,000 ms; word-1 is 3 (word 4 required), subframe-1 is 2 (subframe 3 required), all_svs is false, all_words is false, word3_10 is set to word 4, all_subframes is false, the error type is "set all zero":</p> <pre>00:00:00,GPS_NAV_DATA_ERR,tn_GPS,1,2400,3000,3,2,0,0,4,0,set all zero</pre>																								

5.14.3 GPS_NAV_DATA_MOD

Note: Regardless of your choice of <subframe-1>, you must also specify the <page-1> to which the navigation data modifications apply.

The ability to modify a single page offers greater control; although <subframe-1 = 0> to <subframe-1 = 2> contains the same data on all pages.

If you do not require this control, set <all_pages=1>.

Description	Remote input and insertion into action queue of GPS nav data modifications already scheduled in current scenario
Format	<pre><timestamp>, GPS_NAV_DATA_MOD,<network>,<svid>, <start time ms>,<end time ms>,<word-1>,<page-1>, <subframe-1>,<all_svs>,<all_words>, <all_pages>,<all_subframes> <data mods></pre>
Where	<pre>timestamp ::= see section 5.1.1 network ::= "tn_GPS" svid ::= satellite svid start time ms ::= start time, milliseconds end time ms ::= end time, milliseconds word-1 ::= required word minus 1 page-1 ::= required page minus 1 subframe-1 ::= required subframe minus 1 all_svs ::= "0" (false) "1" (true) all_words ::= "0" (false) "1" (true) all_pages ::= "0" (false) "1" (true) all_subframes ::= "0" (false) "1" (true) data mods ::= modify each bit using: "1" (set to 1) "0" (set to 0) "X" (invert current setting) "-" (do not change) These bits correspond to the first 24 bits of the 30-bit nav word. The remaining 6 bits are parity bits, calculated after the modifications to preserve parity. The 24 bits are in 8-bit sections separated by spaces.</pre>
Returns	status see returned response format, section 5.1.2
Example	<p>At start of scenario, set GPS nav data modifications on SVID25. Errors start at 6,000 ms and end at 12,000 ms; word-1 is 6 (required word is 7), page-1 is 4 (required page is 5), subframe-1 is 2 (required subframe is 3), all_svs is false, all_words is false, all_pages is false, all_subframes is false, data mods are: first group of 8 bits set to "0", second group of 8 bits set to "1", first 4 bits of third group set to "invert", second 4 bits of third group set to "do not change":</p> <pre>00:00:00,GPS_NAV_DATA_MOD,tn_GPS,25,6000,12000,6,4,2,0,0, 0,0,00000000 11111111 XXXX----</pre>

5.14.4 WAGE

Description	Enables / disables WAGE for the whole constellation. Actioned on next upload following the command.
Format	<timestamp>,WAGE,<network>,<state>
Where	timestamp ::= see section 5.1.1 network ::= "GPS" state ::= "0" (not encrypted) "1" (encrypted) "2" (unavailable) "3" (reserved)
Returns	status see returned response format, section 5.1.2
Example	At start of scenario, set WAGE for whole GPS constellation to be encrypted. 00:00:00,WAGE,GPS,1

5.15 Navigation Data related commands

Command	Description
Navigation Data related commands	
QZ_SYMB	Sends Quasi-Zenith navigation data
GPS_LEGACY_NAV	Sends GPS legacy navigation data

5.15.1.1 SVID rule

SimGEN applies SVID settings only while the SVID is visible. SimGEN does not retain non-default SVID settings after the SVID sets.

5.15.2 QZ_SYMB

[illegible]

5.15.3 GPS_LEGACY_NAV

Description	<p>Sends legacy GPS navigation data</p> <p>Notes:</p> <p>1) You must send this command at least 2 seconds before <time>, the time of applicability,</p> <p>2) <time>, the time of applicability, must be on a 6-second boundary</p>
Format	<timestamp>,<GPS_LEGACY_NAV>,<time>,<network>,<svid>,<nav_data>
Where	<p>timestamp ::= see section 5.1.1</p> <p>time ::= time of applicability. Time at which the nav data is valid. Integer seconds from start of scenario</p> <p>network> ::= "GPS"</p> <p>svid ::= satellite svid</p> <p>nav_data ::= 75 hex characters, containing the nav data for one subframe - 300 bits of data (6 seconds)</p>
Returns	status see returned response format, section 5.1.2
Example	<p>At the start of the scenario, send to SVID8 at a time of applicability 6 seconds from start of scenario these GPS legacy nav data symbols:</p> <pre>00:00:00,GPS_LEGACY_NAV,6,GPS,8,425e359ef8ffc403da416bbccf 7d1ddf03030303d7d8d70c37111dffffffffffffffffffffffff</pre>

Note: Frequency resolutions less than 0.01 Hz will exceed the Spirent Product Specification

5.16 Interferer commands

These commands let you remotely send interferer commands, instead of using the ISS GUI. Reference h) gives details of the ISS commands.

Command	Description
Interferer commands	
COHERENT_CW	Generates a coherent CW signal
NON_COHERENT_CW	Generates a non-coherent CW signal
SWEPT_CW	Generates a swept CW signal
AM	Generates an AM signal
FM	Generates an FM signal
NOISE	Generates white Gaussian noise
PULSE - basic	Overlays the current modulation with a pulsing signal defined using period and width
PULSE - using prf and duty cycle	Overlays the current modulation with a pulsing signal defined using prf and duty cycle
RF_ON_OFF	Specifies when the interference source is on
DELTA_LEVEL	Specifies an increment in signal level
LEVEL	Changes the absolute signal level
FIXED_MODE	Fixes current signal at a constant level
MODELLED_MODE	Fixes geographical position of interferer
TX_ANTENNA_PATTERN	Applies a transmit antenna pattern

5.16.1.1 Common interferer commands and returns

Common interferer commands are listed below:

Description	Common interferer commands
Common commands	timestamp ::= see section 5.1.1
	veh_ant ::= <vehicle_id>"_ "<antenna_id>
	vehicle_id ::= "v"<v_number>
	v_number ::= vehicle number starting from 1
	antenna_id ::= "a"<a_number>
	a_number ::= antenna number starting from 1
	interferer_ID ::= Equivalent to Channel in the ISS
	ref_level ::= RF reference level Range: 40 to -146 dBm in 0.02 dBm steps
	level ::= Relative to ref_level Range: 146 to -182 dB in 0.02 dB steps
	delta_level ::= increments the signal level Range: 142 to -142 dB in 0.02 dB steps
	RF_on ::= "on" (sets interfering signal on) "off" (sets interfering signal off)

Common commands (cont)	centre_freq ::= Centre frequency of interfering signal, Hz Range: 500 MHz to 2000 MHz Note: Frequency resolutions less than 0.01 Hz will exceed the Spirent Product Specification
	start_freq ::= Start frequency of swept interferer signal, Hz Range: 500 MHz to 4000 MHz Note: Frequency resolutions less than 0.01 Hz will exceed the Spirent Product Specification
	stop_freq ::= Stop frequency of swept interferer signal, Hz Range: 500 MHz to 4000 MHz Note: Frequency resolutions less than 0.01 Hz will exceed the Spirent Product Specification
	points ::= Number of discrete frequency points between start_freq and stop_freq Range: 2 to 65,535 discrete points
	dwell ::= Time, in ms, the interfering signal dwells at each <points> Range: 1 ms to 100,000 ms
	wave ::= wave form of interfering signal "sine" "square" "ramp" "triangle"
	rate ::= frequency of modulating signal, Hz Range: 0.1 to 50,000 Hz in 0.1 Hz steps

Common returns:

status - see returned response format, section 5.1.2

error - gives a description of the error.

Any error in an interferer signal generator parameter that will send the signal generator out of specification results in a 'clipped' error message, stating the parameter that is in error and that it has been clipped to ensure it is within specification.

5.16.2 COHERENT_CW

Description	Generates a coherent CW signal
Format	<timestamp>,<COHERENT_CW>,<veh_ant>,<interferer_ID>,<ref_level>,<level>,<RF_on>,<centre_freq>
Where	See common commands, section 5.14.1.1
Returns	
Example	Now, use a coherent CW interferer on vehicle 1 antenna 1 using interferer 1, with a reference level of -144 dBm and 144 dB relative to the reference level, with RF on, on this frequency (Hz): -,COHERENT_CW,v1_a1,1,-144,144,on,1234123123

5.16.3 NON_COHERENT_CW

Description	Generates a non-coherent CW signal
Format	<timestamp>,<NON_COHERENT_CW>,<veh_ant>,<interferer_ID>,<ref_level>,<level>,<RF_on>,<centre_freq>
Where	(See common commands, section 5.14.1.1)
Returns	
Example	Now, use a non-coherent CW interferer on vehicle 1 antenna 1 using interferer 1, with a reference level of -144 dBm and 154 dB relative to the reference level, with RF on, on this frequency (Hz): -,NON_COHERENT_CW,v1_a1,1,-144,154,on,1233523123

5.16.4 SWEPT_CW

Description	Generates a swept CW signal
Format	<timestamp>,<SWEPT_CW>,<veh_ant>,<interferer_ID>,<ref_level>,<level>,<RF_on>,<start_freq>,<stop_freq>,<points>,<dwell>
Where	(See common commands, section 5.14.1.1)
Returns	
Example	Now, use a swept CW interferer on vehicle 1 antenna 1 using interferer 1, with a reference level of -144 dBm and 124 dB relative to the reference level, with RF on, using these start and stop frequencies (Hz), with 400 points between the start and stop frequencies and dwelling for 10 ms at each point: -,SWEPT_CW,v1_a1,1,-144,124,on,1233000000,1235000000,400,10

5.16.5 AM

Description	Generates an AM signal
Format	<timestamp>,<AM>,<veh_ant>,<interferer_ID>,<ref_level>,<level>,<RF_on>,<centre_freq>,<wave>,<rate>,<depth>
Where	See common commands, section 5.14.1.1 depth ::= Modulation depth Range: 0 to 90% in 0.1% steps
Returns	
Example	Now, use an AM interferer on vehicle 1 antenna 1 using interferer 1, with a reference level of -144 dBm and 134 dB relative to the reference level, with RF on, using this centre frequency (Hz), with square wave modulation, frequency 100 Hz and 90% modulation depth: -,AM,v1_a1,1,-144,134,on,1234234234,square,100.0,90

5.16.6 FM

Description	Generates an FM signal
Format	<timestamp>,<FM>,<veh_ant>,<interferer_ID>,<ref_level>,<level>,<RF_on>,<centre_freq>,<wave>,<rate>,<deviation>
Where	See common commands, section 5.14.1.1 deviation ::= Deviation of modulating signal, Hz Upper and lower limits are dependant on <centre_freq> - see Table 5-25
Returns	
Example	Now, use an FM interferer on vehicle 1 antenna 1 using interferer 1, with a reference level of -144 dBm and 144 dB relative to the reference level, with RF on, using this centre frequency (Hz), with triangular modulation, frequency 100.0 Hz and 90,000 Hz deviation: <pre>- ,FM,v1_a1,1,-144,144,on,1234234234,triangle,100.0,90000</pre>

Table 5-26 Limits for <centre_freq> of FM interfering signal

<centre_freq>	Lower limit, Hz	Upper limit, Hz
Less than 500	5	5,000
500 to 1000	10	10,000
1,000 to 2,000	20	20,000

5.16.7 NOISE

Description	Generates white Gaussian noise
Format	<timestamp>,<NOISE>,<veh_ant>,<interferer_ID>,<ref_level>,<level>,<RF_on>,<centre_freq>,<bandwidth>
Where	See common commands, section 5.14.1.1 bandwidth ::= 1 Hz to 48 MHz in 0.01 Hz steps
Returns	
Example	Now, use a white Gaussian noise interferer on vehicle 1 antenna 1 using interferer 1, with a reference level of -144 dBm and 144 dB relative to the reference level, with RF on, using this centre frequency (Hz) and this bandwidth (Hz): <pre>- ,NOISE,v1_a1,1,-144,144,on,1234234234,48000000</pre>

5.16.8 PULSE - basic

Description	Overlays the current modulation with a pulsing signal described by its period and width
Format	<timestamp>, <PULSE>, <veh_ant>, <interferer_ID>, <pulsing_on>, <period>, <width>
Where	<p>See common commands, section 5.14.1.1</p> <p>pulsing _on ::= "on" (sets pulsing signal on) "off" (sets pulsing signal off)</p> <p>period ::= repetition period of pulsing signal Range: 2.1 microseconds to 42 seconds in 0.1 microsecond steps</p> <p>width ::= width of pulse in pulsing signal Range: at least 2 microseconds to no greater than <period> in 0.1 microsecond steps</p>
Returns	
Example	<p>Now, overlay a pulse on the current modulation on vehicle 1 antenna 1 using interferer 1, with RF on, using a period of 1000 μs with pulse width 100 μs:</p> <pre>- , PULSE, v1_a1, 1, on, 1000, 100</pre>

5.16.9 PULSE - using prf and duty cycle

Description	Overlays the current modulation with a pulsing signal described by its period, width, prf and duty cycle
Format	<timestamp>,<PULSE>,<veh_ant>,<interferer_ID>,<pulsing_on>,<prf_and_duty>,<period>,<width>,<prf>,<duty_cycle>
Where	See common commands, section 5.14.1.1
	<p>pulsing_on ::= "on" (sets pulsing signal on) "off" (sets pulsing signal off)</p> <p>prf_and_duty ::= "on" (use prf and duty cycle) "off" (do not use prf and duty cycle)</p> <p>period ::= repetition period of pulse in pulsing signal Range: 2.1 microseconds to 42 seconds in 0.1 microsecond steps</p> <p>width ::= width of pulse in pulsing signal Range: at least 2 microseconds to no greater than <period> in 0.1 microsecond steps</p> <p>prf ::= pulse repetition frequency of pulsing signal Range: ~0 to ~476 kHz (set by Agilent N5182A ISG)</p> <p>duty_cycle ::= duty cycle of pulsing signal Range: ~0 to ~100% (set by Agilent N5182A ISG)</p>
Returns	
Example	<p>Now, overlay a pulse on the current modulation on vehicle 1 antenna 1 using interferer 1, with pulsing on and using the prf and duty cycle display option in the interferer command editor using a period of 1000 μs with pulse width 450 μs and a prf of 1 kHz with a duty cycle of 45%:</p> <pre>-,PULSE,v1_a1,1,on,on,1000,450,1,45</pre>

5.16.10 RF_ON_OFF

Description	Specifies when the interference source is on
Format	<timestamp>,<RF_ON_OFF>,<veh_ant>,<interferer_ID>,<RF_on>
Where	See common commands, section 5.14.1.1
Returns	
Example	Now, turn off the interfering source on vehicle 1 antenna 1 using interferer 1: -,RF_ON_OFF,v1_a1,1,off

5.16.11 DELTA_LEVEL

Description	Specifies an increment in signal level
Format	<timestamp>,<DELTA_LEVEL>,<veh_ant>,<interferer_ID>,<delta_level>
Where	See common commands, section 5.14.1.1
Returns	
Example	Now, increment the interfering signal level on vehicle 1 antenna 1 using interferer 1 by +25.0 dB: -, DELTA_LEVEL, v1_a1, 1, 25.0

5.16.12 LEVEL

Description	Changes the absolute signal level
Format	<timestamp>,<LEVEL>,<veh_ant>,<interferer_ID>,<ref_level>,<delta_level>
Where	See common commands, section 5.14.1.1
Returns	
Example	Now, change the absolute signal level on vehicle 1 antenna 1 using interferer 1 to a reference level of -125 dBm, incremented by +25 dB: -, LEVEL, v1_a1, 1, -125.0, 25.0

5.16.13 FIXED_MODE

Description	Fixes current signal at a constant level
Format	<timestamp>,<FIXED_MODE>,<veh_ant>,<interferer_ID>,<level>
Where	See common commands, section 5.14.1.1
Returns	
Example	Now, fix the signal level currently applied to vehicle 1 antenna 1 using interferer 1 at a constant level of -100.0 dB relative to the reference level: -, FIXED_MODE, v1_a1, 1, -100.0

5.16.14 MODELLED_MODE

Description	Fixes geographical position of interferer <i>Note: When you use MODELLED_MODE the power level is taken from the previous modulation command and the base level is set from the instant the command is entered, even if the vehicle has moved away from the interferer</i>
Format	<timestamp>,<MODELLED_MODE>,<veh_ant>,<interferer_ID>,<latitude>,<longitude>,<height>
Where	See common commands, section 5.14.1.1 latitude ::= interferer latitude, radians longitude ::= interferer longitude, radians height ::= interferer height, metres
Returns	
Example	Now, fix the geographical position of the interfering signal applied to vehicle 1 antenna using interferer 1 to this latitude (radians), longitude (radians) and height (m) -,MODELLED_MODE,v1_a1,1,-0.00003,-0.00003,-1.23456

5.16.15 TX_ANTENNA_PATTERN

Description	Applies an antenna attenuation pattern file to the interferer in modelled mode
Format	<timestamp>,<TX_ANTENNA_PATTERN>,<veh_ant>,<interferer_ID>,<apply_pattern>,<boresight_az>,<boresight_el>,<file>
Where	See common commands, section 5.14.1.1 apply_pattern ::= "on" (applies attenuation) "off" (disables attenuation) boresight_az ::= boresight azimuth of interferer antenna boresight_el ::= boresight elevation of interferer antenna file ::= attenuation file name (*.ant_pat)
Returns	
Example	Now, apply the antenna attenuation pattern file (<i>atten.ant_pat</i>) to the interferer modelled mode signal level on vehicle 1 antenna 1 interferer 1 and assume the interferer antenna boresight is at an azimuth of 4.5° and an elevation of 3.5°: -,TX_ANTENNA_PATTERN,v1_a1,1,on,4.5,3.5, atten.ant_pat

5.17 UDP commands

You can send UDP/IP commands from SimREMOTE to SimGEN using port 15650. In SimGEN select **Options - Remote Command Setting - Ethernet Remote Commands over TCP/UDP** and select **Enable port**.

UDP only supports the following binary commands:

Command	Description
MOT	UDP command to send MOT data
MOTB	UDP command to send MOTB data
MOD	UDP command to send MOTD data

The UDP command structure is given in the file *UDP_rx_commands.h*

5.17.1 Command structure

Notes:

- 1) If you use a remote source, a latency message appears in the SimGEN **System Messages** window.
- 2) All UDP data must be little-endian, packed in an 8-byte format.

The UDP command structure consists of a generic part and a union defining the data packets.

```
struct Command
{
    enum Type
    {
        mot = 0,
        motb,
        mod
    };
    enum Time_action
    {
        action_immediately = 0,
        action_at_timestamp
    };
    Type                type_;
    Time_action         time_action_;
    unsigned int        time_of_validity_ms_;

    // Starts from 1...n
    unsigned int        vehicle_id_;

    union
    {
        Mot_data        mot_;
        Motb_data       motb_;
        Mod_data        mod_;
    } data;

    int latency_wrt_tov_and_current_tir_ms_;
};
```

Table 5-26 lists UDP structure commands and their equivalent commands.

Table 5-27: UDP Command structure

UDP command	Description	Equivalent command
Type	mot = 0, motb = 1 mod = 2	MOT MOTB MOD
time_action_	action_immediately = 0 action_at_timestamp = 1	<timestamp> = “-“ (now); or “d hh.mm.ss.ms”
time_of_validity_ms_	sets time in ms for when action_at_timestamp is set	
vehicle_id_	from 1 to n	<vehicle_id>
latency_wrt_tov_and_ current_tir_ms_	This data is only included in the binary logging file. Positive number - data arriving early (before SimGEN epoch). Negative number - data arriving late (after SimGEN epoch)	In SimGEN select Options - Remote Command Setting - Ethernet Remote Commands over TCP/UDP and select Enable port and Enable binary logging . Click Browse and navigate to the folder where you want to store the binary logging file (.bin).

5.17.2 MOT

```

struct Mot_data
{
    double    position_ecef_xyz_           [ 3 ];
    double    velocity_mps_xyz_           [ 3 ];
    double    acceleration_mps2_xyz_;      [ 3 ];
    double    jerk_mps3_xyz_              [ 3 ];
    double    heb_                        [ 3 ];

    // About body axis
    double    angular_velocity_radps_xyz_ [ 3 ];
    double    angular_acceleration_radps_xyz_ [ 3 ];
    double    angular_jerk_radps_xyz_      [ 3 ];
};

```

Table 5-27 gives UDP commands and their equivalent MOT commands.

Table 5-28: UDP commands for MOT

UDP command	MOT command from section 5.8.1	Data
position_ecef_xyz_	x, y, z	double precision
velocity_mps_xyz_	vx, vy, vz	double precision
acceleration_ned_mps2_	ax, ay, az	double precision
jerk_ned_mps3_	jx, jy, jz	double precision
heb_	h, e, b	double precision
angular_velocity_radps_xyz_	$\omega_x, \omega_y, \omega_z$	double precision
angular_acceleration_radps_xyz_	$\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z$	double precision
angular_jerk_radps_xyz_	$\ddot{\omega}_x, \ddot{\omega}_y, \ddot{\omega}_z$	double precision

Table 5-29 shows the layout of the C++ structure SimGEN uses to receive UDP commands.

All data must be formatted as little endian and aligned on an eight-byte boundary

Table 5-29 C++ structure for MOT command

Offset	Field	Size (bytes)
0x00	type_	4
0x04	time_action_	4
0x08	time_of_validity_ms_	4
0x0c	vehicle_id_	4
0x10	data.mot_.position_ecef_xyz_[0]	8
0x18	data.mot_.position_ecef_xyz_[1]	8
0x20	data.mot_.position_ecef_xyz_[2]	8
0x28	data.mot_.velocity_mps_xyz_[0]	8
0x30	data.mot_.velocity_mps_xyz_[1]	8
0x38	data.mot_.velocity_mps_xyz_[2]	8
0x40	data.mot_.acceleration_mps2_xyz_[0]	8
0x48	data.mot_.acceleration_mps2_xyz_[1]	8
0x50	data.mot_.acceleration_mps2_xyz_[2]	8
0x58	data.mot_.jerk_mps3_xyz_[0]	8
0x60	data.mot_.jerk_mps3_xyz_[1]	8
0x68	data.mot_.jerk_mps3_xyz_[2]	8
0x70	data.mot_.heb_[0]	8
0x78	data.mot_.heb_[1]	8
0x80	data.mot_.heb_[2]	8
0x88	data.mot_.angular_velocity_radps_xyz_[0]	8
0x90	data.mot_.angular_velocity_radps_xyz_[1]	8
0x98	data.mot_.angular_velocity_radps_xyz_[2]	8
0xa0	data.mot_.angular_acceleration_radps_xyz_[0]	8
0xa8	data.mot_.angular_acceleration_radps_xyz_[1]	8
0xb0	data.mot_.angular_acceleration_radps_xyz_[2]	8
0xb8	data.mot_.angular_jerk_radps_xyz_[0]	8
0xc0	data.mot_.angular_jerk_radps_xyz_[1]	8
0xc8	data.mot_.angular_jerk_radps_xyz_[2]	8
0xd0	latency_wrt_tov_and_current_tir_ms_	4
0xd4	Packing bytes	4

5.17.3 MOTB

```

struct Motb_data
{
    double    latitude_;
    double    longitude_;
    double    height_;

    double    velocity_ned_mps_           [ 3 ];
    double    acceleration_ned_mps2_     [ 3 ];
    double    jerk_ned_mps3_             [ 3 ];
    double    heb_                        [ 3 ];

    // About body axis
    double    angular_velocity_radps_xyz_ [ 3 ];
    double    angular_acceleration_radps_xyz_ [ 3 ];
    double    angular_jerk_radpsps_xyz_ [ 3 ];
};

```

Table 5-28 gives UDP commands and their equivalent MOTB commands.

Table 5-30: UDP commands for MOTB

UDP command	MOTB command from section 4.4.9.2	Data
latitude_	lat	double precision
longitude_	long	double precision
height_	height	double precision
velocity_ned_mps_	vel_north, vel_east, vel_down	double precision
acceleration_ned_mps2_	acc_north, acc_east, acc_down	double precision
jerk_ned_mps3_	jerk_north, jerk_east, jerk_down	double precision
heb_	h, e, b	double precision
angular_velocity_radps_xyz_	$\omega_x, \omega_y, \omega_z$	double precision
angular_acceleration_radps_xyz_	$\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z$	double precision
angular_jerk_radpsps_xyz_	$\ddot{\omega}_x, \ddot{\omega}_y, \ddot{\omega}_z$	double precision

Table 5-31 shows the layout of the C++ structure SimGEN uses to receive UDP commands.

All data must be formatted as little endian and aligned on an eight-byte boundary

Table 5-31 C++ structure for MOTB command

Offset	Field	Size (bytes)
0x00	type_	4
0x04	time_action_	4
0x08	time_of_validity_ms_	4
0x0c	vehicle_id_	4
0x10	data.motb_.latitude_	8

Offset	Field	Size (bytes)
0x18	data.motb_.longitude_8	8
0x20	data.motb_.height_	8
0x28	data.motb_.velocity_ned_mps_[0]	8
0x30	data.motb_.velocity_ned_mps_[8
0x38	data.motb_.velocity_ned_mps_[8
0x40	data.motb_.acceleration_ned_mps2_[0]	8
0x48	data.motb_.acceleration_ned_mps2_[1]	8
0x50	data.motb_.acceleration_ned_mps2_[2]	8
0x58	data.motb_.jerk_ned_mps3_[0]	8
0x60	data.motb_.jerk_ned_mps3_[1]	8
0x68	data.motb_.jerk_ned_mps3_[2]	8
0x70	data.motb_.heb_[0]	8
0x78	data.motb_.heb_[1]	8
0x80	data.motb_.heb_[2]	8
0x88	data.motb_.angular_velocity_radps_xyz_[0]	8
0x90	data.motb_.angular_velocity_radps_xyz_[1]	8
0x98	data.motb_.angular_velocity_radps_xyz_[2]	8
0xa0	data.motb_.angular_acceleration_radps_xyz_[0]	8
0xa8	data.motb_.angular_acceleration_radps_xyz_[1]	8
0xb0	data.motb_.angular_acceleration_radps_xyz_[2]	8
0xb8	data.motb_.angular_jerk_radps_xyz_[0]	8
0xc0	data.motb_.angular_jerk_radps_xyz_[1]	8
0xc8	data.motb_.angular_jerk_radps_xyz_[2]	8
0xd0	latency_wrt_tov_and_current_tir_ms_	4
0xd4	Packing bytes	4

5.17.4 MOD

```

struct Mod_data
{
    enum SigType
    {
        gps = 0,
        waas,
        egnos,
        msas,
        glonass,
        galileo,
        gps_2
        gps_2
        gps_3
        gps_4
        gps_5
        gps_6
        gps_7
        gps_8
        gps_9
        gps_10
        gps_11
        gps_12
    };

    enum Mode
    {
        svid = 0,
        channel
    };

    enum Applicability
    {
        single = 0,
        all
    };

    enum Frequency
    {
        L1 = 0,
        L2,
        L5
    };

    // Starts from 1
    unsigned int    antenna_id_;

    SigType         signal_type_;
    unsigned int    svid_or_channel_no_;
    unsigned int    multipath_index_;
    Mode            mode_;
    Applicability   all_channels_;

    Frequency       frequency_;
    Applicability   all_frequencies_;

    double          signal_level_;
    double          carrier_offset_;
    double          code_offset_;
};

```

Table 5-29 gives UDP commands and their equivalent MOD commands.

Table 5-32: UDP commands for MOD

UDP command	MOD command from section 5.6.4	Data
signal_type_	signal_type	gps = 0 waas = 1 egnos = 2 msas = 3 glonass = 4 galileo = 5
mode_	id	SVID = 0 Channel = 1
Applicability	single all (applies to mode, all_flag, all_freq)	single = 0 all = 1
Frequency	freq	L1 = 0 L2 = 1 L5 = 2
antenna_id_	ant_id	starting from 1
multipath_index_;	multi_index	0 = no multipath 1-n = number of multiple paths used
signal_level_;	sig_level	dB in double precision
carrier_offset_;	carr_offset	dB in double precision
code_offset_;	code_offset	dB in double precision

Table 5-33 shows the layout of the C++ structure SimGEN uses to receive UDP commands.

All data must be formatted as little endian and aligned on an eight-byte boundary

Table 5-33 C++ structure for MOTB command

Offset	Field	Size (bytes)
0x00	type_	4
0x04	time_action_	4
0x08	time_of_validity_ms_	4
0x0c	vehicle_id_	4
0x10	data.mod_.antenna_id_	4
0x14	data.mod_.signal_type_	4
0x18	data.mod_.svid_or_channel_no_	4
0x1c	data.mod_.multipath_index_	4
0x20	data.mod_.mode_	4
0x24	data.mod_.all_channels_	4
0x28	data.mod_.frequency_	4
0x2c	data.mod_.all_frequencies_	4
0x30	data.mod_.signal_level_	8
0x38	data.mod_.carrier_offset_	8
0x40	data.mod_.code_offset_	8

Offset	Field	Size (bytes)
0x48	Packing bytes	136
0xd0	latency_wrt_tov_and_current_tir_ms_	4
0xd4	Packing bytes	4

Chapter 6: Propagation of remote motion data

Motion data from a remote source may be applied to SimGEN using a suitable interface (TCP/IP, IEEE-488 or SCRAMNet) or may be read from a file of data. This data may be formatted as ECEF (MOT) or geodetic (MOTB) messages. The formats for these messages are given in section 5.9.

6.1 Motion message formats

For MOT messages the following parameters are defined:

- a) Time of Application (Toa), (in units of ms)
- b) ECEF Position Vector (P_x, P_y, P_z)
- c) ECEF Velocity Vector (V_x, V_y, V_z)
- d) ECEF Acceleration (A_x, A_y, A_z)
- e) ECEF Jerk (J_x, J_y, J_z)
- f) Vehicle Attitude (Heading, Elevation, Bank)
- g) Body Rotation Rates ($\omega_x, \omega_y, \omega_z$)
- h) Body Angular Acceleration ($\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z$)
- i) Body Angular Jerk ($\ddot{\omega}_x, \ddot{\omega}_y, \ddot{\omega}_z$)

Parameters are defined in units of metres, radians and hh:mm:ss.sss, as appropriate.

All motion is with respect to the ECEF frame (WGS-84). This means that linear motion does not include inertial acceleration terms (that is Coriolis and centripetal effects), and angular rotation rate does not include Earth rate. Linear motion is also resolved in the ECEF frame.

Attitude is defined as an Euler angle rotation sequence (in order heading, elevation, bank) from alignment of the body axes (x,y,z) with the local geodetic frame (North, East, Down) to the current orientation. Rotation rate, acceleration and jerk are all resolved with respect to the vehicle body axes.

Data is valid at the Time of Application (Toa), and describes the vehicle motion until the time of the next message, (normally one time-step).

6.2 Application of remote motion data

SimGEN supports use of motion sample rates from 1 Hz (or even lower if appropriate) up to 500 Hz (2 ms interval).

The method used to calculate and update signal pseudorange rate within the hardware varies depending on the hardware type and/or operating mode.

The method also varies slightly depending on the source of the data, such as whether the data is being read from a file (where all samples are available) or being delivered from an external source in real-time (where only previous samples are available).

In all cases, the method employed seeks to deliver an accurate representation of the correct simulated position at the correct time, and hence yields a situation where there is **Zero Effective Latency** between the intended and actual simulation for all practical vehicle trajectories.

SimGEN successfully achieves this performance goal using a proprietary prediction method.

6.3 High-end signal generators

These include the GSS8000, GSS7900 and GSS 7790 signal generators and superseded signal generators such as the STR4760, STR4790 and the STR4780 GLONASS

Apart from the STR4780 (which only supports the 100 ms iteration rate) these signal generators support two iteration rates, 10 ms and 100 ms, which represent the interval over which the SimGEN software internally calculates pseudorange data. These periods are aligned to the integer 10 ms or 100 ms epochs relative to the simulation start time.

The software embedded in the signal generator performs a significant role in interpolating pseudorange data between epochs to a higher rate - 500 Hz for 10 ms iteration rate and 250 Hz for 100 ms iteration rate.

The latest GSS8000 and GSS7xxx platforms (which have Ethernet connectors on the rear panel and firmware version 1-24 or later) also support 4 ms iteration rate and update the hardware control registers at 1000 Hz for 4 ms and 10 ms iteration rates.

6.4 Low-end signal generators

Currently, the only product to which this category applies is the GSS6560.

In the GSS6560, SimGEN performs all pseudorange calculations and delivers this directly to the hardware every 10 ms.

The method by which SimGEN calculates the pseudorange at the epochs is the same as for high-end platforms operating at 10 ms iteration rate.

6.5 Descriptions of iteration rates

The following sections describe the available iteration rates.

6.5.1 Iteration rate of 4 milliseconds

Note: The 4 ms iteration rate is similar to the 10 ms iteration rate and is not described separately.

6.5.2 Iteration rate of 10 milliseconds

Note: The following description of the propagation method employed uses the case where trajectory data is delivered in real time and where the samples are received very close to their intended timestamp time. The method for data read from a file is very similar but as data for the current and next epoch is always available in advance estimates of positions are not required.

Every 10 ms of simulation time, SimGEN provides the signal generator with calculated pseudorange data valid for the next 10 ms epoch.

Every 10 ms, SimGEN first calculates an estimate of the vehicle position for the end of the next period based on the most recent motion data sample that is available. It does this using the entire content of the sample (position, velocity, acceleration and jerk) to perform a straightforward mathematical extrapolation from the time of validity of the sample out to the end of the next 10 ms period.

SimGEN then calculates the pseudorange and pseudorange-rate for each simulated satellite at the end-of-period epoch using this projected position.

In the case of the low-end platforms, the pseudorange-rate is calculated in a format that can be directly used by the hardware and sent by SimGEN every 10 ms.

For the high-end platforms, in order to minimise data volume across the interface bus, SimGEN only sends pseudorange and pseudorange-rate at 100 ms epoch boundaries to the signal generator.

Additionally, every 10 ms it calculates pseudorange-acceleration for each 10 ms period, using the pseudorange data obtained for the end of the current 10 ms period plus the data for the end of the next period. This acceleration term is sent every 10 ms to the signal generator hardware. It is used within the signal generator to calculate locally the subsequent 10 ms epoch data and to interpolate between the 10 ms data to an intermediate 1 or 2 ms hardware pseudorange-rate update, giving a 1000 or 500 Hz hardware control register update rate. Care is taken to ensure that there are no cumulative error effects in the extrapolation/interpolation process. Pseudorange then changes continuously in a linear fashion over the 1 or 2 ms period at the calculated pseudorange rate.

6.5.3 Iteration rate of 100 milliseconds (high-end platforms only)

Note: *The following description of the propagation method employed uses the case where trajectory data is delivered in real time and where the samples are received very close to their intended timestamp time. The method for data read from a file is very similar but as data for the current and next epoch is always available in advance estimates of positions are not required.*

Every 100 ms of simulation time, SimGEN provides the signal generator with pseudorange data for the next entire 100 ms period.

Every 100 ms SimGEN calculates an estimate of the vehicle position for the end of the next period based on the most recent motion data sample that is available. It does this using the entire content of the sample (velocity, acceleration and jerk) to perform a straightforward mathematical extrapolation from the time of validity of the sample out to the end of the next 100 ms period.

Using the starting and ending pseudorange rates, SimGEN then calculates the coefficients of a clamped cubic spline curve-fit between the projected pseudorange for the end of the next period and that used for the end of the current period. These coefficients (comprising change of pseudorange and deltarange over the period plus pseudorange-rate/acceleration/jerk) are passed every 100 ms to the signal generator.

The signal generator uses the pseudorange coefficients to fit a cubic expansion between the two 100 ms end-points with a resolution of 4ms, giving a 250 Hz simulation rate for pseudorange-rate. The change in pseudorange over the period is used to clamp the process and prevent cumulative error. Pseudorange changes continuously, in a linear fashion, over a 4ms period at the calculated pseudorange rate.

6.6 Synchronisation of external motion data

SimREMOTE is extremely tolerant of the timing associated with delivery of the remote motion data.

All samples are normally time tagged with their time of validity.

Because SimGEN always uses the most recent sample in its buffer, small variations in the time of reception have little impact. They only affect the absolute accuracy of the motion prediction. Take the case of a missed 10 ms (100 Hz) sample. SimGEN would predict based on the previous sample, which is only 10 ms 'old'. Very few trajectories have significant change in dynamics (large Jerk) over 10 ms, so the accuracy of the predicted position is still very high. Of course, this accuracy degrades when many consecutive samples are not transmitted. However, whenever a sample is received close to its time-tagged time, the system accuracy is restored automatically.

This means that the remote system does not even have to comply with sending the data accurately on the time-tag boundaries. Individual samples can be sent slightly early or late with little or no impact.

In fact, the incoming data rate does not need to match SimGEN's simulation iteration rate at all, though the resolution of the time tags must be in integer milliseconds. It is perfectly suitable, for example, to send data at 4 Hz using either 10 ms or 100 ms iteration rates, or at 250 Hz or 500 Hz.

6.7 Latency

Note: *The issue of latency is complex and is prone to misinterpretation unless carefully defined.*

Optimum accuracy is obtained when data samples are sent slightly prior (approximately 6 ms but dictated by processing time. In the 4 ms iteration rate, data is sampled one time step early to the

epoch to which they relate. This is generally only possible in the open-loop case, where data is being read from a file, either located on the SimGEN PC or being delivered using the SimREMOTE interface.

The latency could be defined as the delay between the time-stamp of the sample and the simulation time when that position is actually represented accurately at RF. In this case, the latency would be zero, as the data is known in advance and delivered early, so SimGEN can ensure it is applied at the correct time.

However, latency could be defined as the delay between time of transmission of the sample and when the sample is accurately represented in the output. In the above optimal open-loop case, the latency would then be 6 ms, even though the correct position is simulated at exactly the correct time!

The issue is further complicated by SimGEN's treatment of samples that have a higher rate than its processing rate (10 ms or 100 ms iteration rates). By definition, not all the samples are used, just the most recent. Latency based on the definition tied to transmission time can only be defined for the used samples. For the 10 ms iteration rate, the following latencies occur:

The maximum latency to a sample is 16 ms: 6 ms processing time plus a complete 10 ms cycle (for the case when a sample arrives immediately after the SimGEN processing epoch).

The minimum latency is 6 ms for a sample arriving immediately before the processing epoch.

For 100 ms iteration rate, the range (using the above definition) is 6 to 106 ms.

In Spirent's predictive system, however, the system only applies the sample itself at RF if read from a file in advance. Instead it applies an accurately predicted position based on the most recent sample available at the processing epoch.

Because the prediction is always very accurate (assuming the data samples are of the required quality) there is **ZERO EFFECTIVE LATENCY** when comparing the generated position at RF and the position intended by the sample, even against those samples that are not used. For extremely high dynamic situations, there is potentially (for a very limited period) a finite, but very small, difference between the position simulated and that intended.

The scale of the error will be dependent on several factors. For example, with a 100 ms iteration rate, the prediction will be inherently less accurate for 500 Hz samples as the prediction interval is greater.

With a 10 ms iteration rate, there has to be extraordinary dynamic content within the samples, in particular large changes in Jerk, to introduce significant transient error. Take the example of a change in Jerk of 6000 m/s³ (a very large value) that occurs in a sample received just at the start of the current 10 ms period. This would mean that its impact would not be implemented for 10 ms. The positional change (and hence error) due to this jerk being missed over a 10 ms period is given by:

$$\text{Error} = J(T)^3/6 = 6000(0.01)^3/6 = 0.001 \text{ or } 1 \text{ mm}$$

Where:

J is the Jerk, m.s⁻³

T is the interval, seconds; and

Error is in metres

The mean error at the following epoch due to a step jerk of this magnitude in an unused sample occurring some time during the 10 ms period is just 0.125 mm.

Subsequently, the error will disappear, until the dynamic manoeuvre is completed by a nominally equal and opposite change of jerk, generating an opposite transient error of the same magnitude.

Jerk of this magnitude is extremely rare in practical trajectories, except for the special cases of, for example, vehicles spinning at high rotation rates (see next paragraph) or during impact or gun-launch (where the severity means the instantaneous error is largely irrelevant as the GPS receiver would be unable to use the received signal).

For **spinning vehicles**, the signal generator's signal dynamic capability and simulation iteration rate for pseudorange-rate are the limiting factors, rather than the SimREMOTE processing rate.

Rates of spin up to approximately 30 revolutions per second can be successfully handled directly by simply sampling the antenna position locus. Above this rate, Spirent recommends an alternative approach for which additional specialised documentation is available. In this case, however, the linear motion of the vehicle is still represented by the standard SimREMOTE method.

6.8 Calculation of Prange control coefficients (100 ms iteration rate)

Note: An information note on this subject is available from Spirent.

As mentioned above, SimGEN uses a clamped cubic spline method in defining the pseudorange profile over a 100 ms period. If we have consecutive sets of position and velocity data then the appropriate acceleration and jerk coefficients may be determined by using a clamped cubic spline fit between adjacent sets. This is determined as follows:

P_{n-1} = Pseudorange at time t_{n-1}

V_{n-1} = Velocity at time t_{n-1}

P_n = Pseudorange at time $t_n = t_{n-1} + T$

V_n = Velocity at time $t_n = t_{n-1} + T$

T = Time interval between adjacent sets of data = $t_n - t_{n-1}$ Let change in pseudorange over the period T be represented by a cubic equation, so that:

$$P = a + bt + ct^2 + dt^3 \quad (t = 0 \text{ to } T)$$

hence $V = b + 2ct + 3dt^2$ At time $t = 0$, (t_{n-1}) we get:

$a = P_{n-1}$, $b = V_{n-1}$ At $t = T$ (that is, at t_n). Substituting for a and b gives the following expressions for coefficients c and d :

$$c = \frac{3(P_n - P_{n-1})}{T^2} - \frac{(V_n + 2V_{n-1})}{T}$$

$$d = \frac{2(P_{n-1} - P_n)}{T^3} + \frac{(V_n + V_{n-1})}{T^2}$$

In terms of pseudorange acceleration (A) and jerk (J) this means that at t_{n-1} :

$$A_{n-1} = 2c$$

$$J_{n-1} = 6d$$

These calculations are performed separately for each pseudorange component in the ECEF frame.

This page is intentionally blank

Chapter 7: Synchronisation in closed-loop mode

Note: For simplicity, this chapter refers to a 10 MHz reference frequency, but you can use any valid reference frequency. However, for 65xx signal generators and the superseded 45xx signal generators, the timing can be more tightly controlled and there is a benefit in using a reference frequency of 10 MHz.

There is no such benefit for GSS8000-series and GSS7790 signal generators or the superseded 7xxx or 47xx signal generators.

This chapter covers synchronisation by considering remote operation using a 10 Hz motion data update rate and a GSS8000 or GSS77xx signal generator. Similar principles can be inferred for 100 Hz operation, and with the superseded STR47xx platforms.

Spirent signal generators use the same three modes for triggering the start of a simulation.

If you use a GSS6560 based system, read through this section to gain an understanding of the basic principles and see section 7.10 for specific details on the GSS6560.

7.1 General principles

The remote system and the signal generator(s) must use the same timing references. Spirent signal generators use the rising edge of a 1PPS signal as their time reference, see Table 7-1.

Table 7-1 1PPS signal reference frequencies

Signal generator family	1PPS reference source
8000-series	10.23 MHz
7xxx / 47xx /	10.23 MHz
65xx / 45xx	10 MHz

The signal generators have the capability to generate a reference clock signal and are able to use, and lock to, reference signals generated by your equipment. Table 7-2 describes the modes for 1PPS.

Table 7-2 1PPS modes

1PPS mode	Description
Continuous and free running	The remote system must synchronise to 1PPS in order to control the start of the scenario run
Continuous	Synchronised to the remote system
Triggered	1PPS is held off and commences running on receipt of a hardware trigger.

The 1PPS signal is the time reference for a scenario. The scenario starts with the 1PPS, the rising edge of the first 1PPS pulse marking time $t = 0$ of the scenario.

To achieve synchronism, it is crucial all components of the simulation start at the appropriate instant. Table 7-3 shows the triggering modes / sequences supported by SimREMOTE.

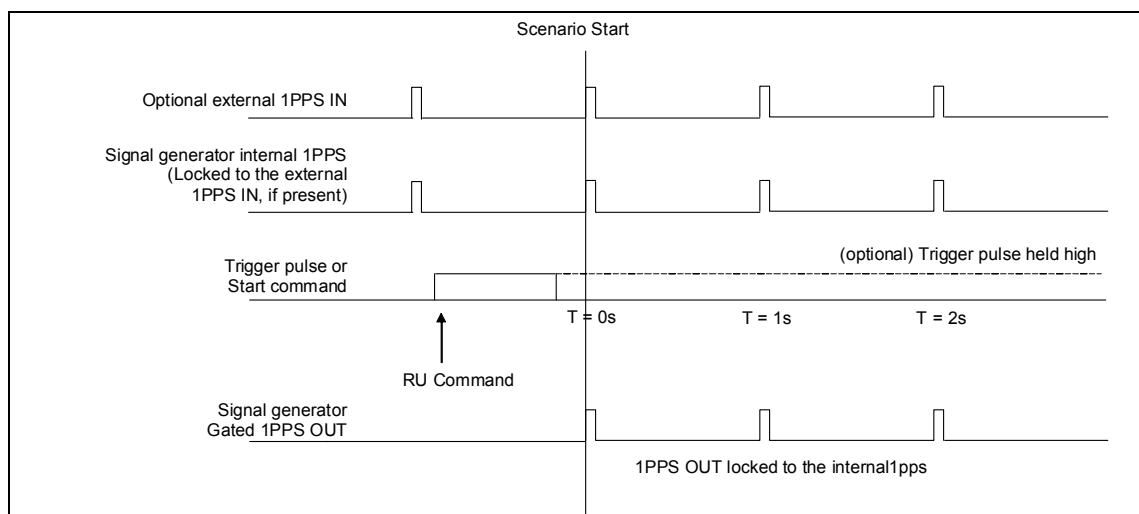
Table 7-3 Supported trigger modes

Trigger mode	Description	See
Disabled	SimGEN induces start on a 1 PPS event	Figure 7-1
Immediate	Rising edge on TRIG IN causes immediate start	Figure 7-2
Delayed	Rising edge on TRIG IN causes on next 1 PPS event	Figure 7-3

Figure 7-1 shows the relationship between the 1PPS signals and the trigger pulse / scenario start command for trigger Disabled and Delayed modes. Figure 7-2 shows the relationship for Immediate trigger mode.

In the absence of an External 1PPS signal applied to SYNC IN the signal generator maintains an internal 1PPS at a constant phase set by the signal generator.

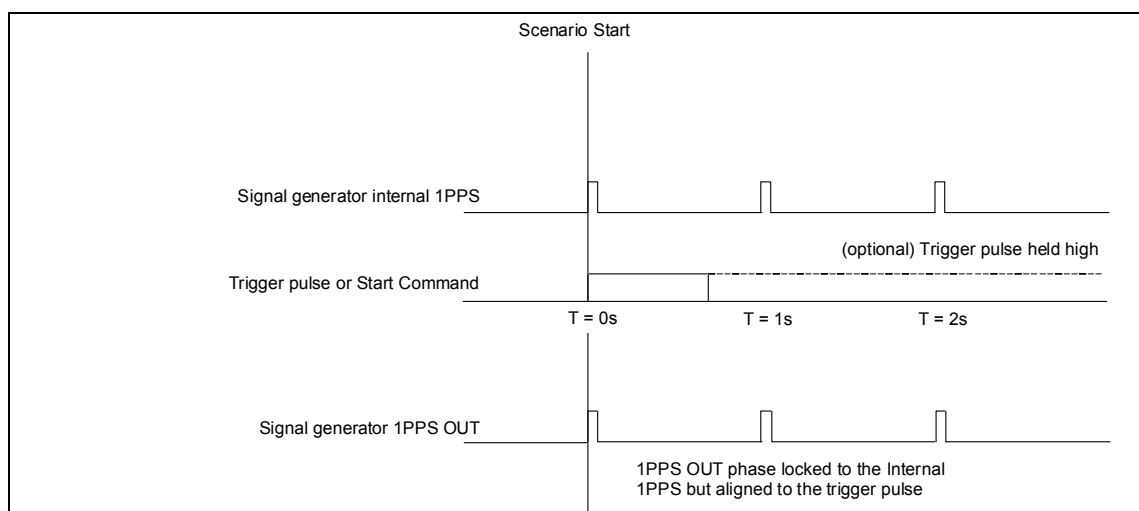
Figure 7-1: Alignment of the 1PPS signal, trigger disabled and delayed mode



The alignments are essentially the same for trigger modes Disabled and Delayed.

Note: When triggering is Disabled, the scenario starts on a software command; however in Delayed mode the scenario starts on a trigger pulse.

Figure 7-2: Alignment of the 1PPS signal, immediate trigger mode



With Immediate trigger mode the remote system chooses the start instant by the timing of the trigger pulse.

Sections 7.3 to 7.5 describe the three trigger modes (simulation start-up modes). Rules for synchronisation are given in each case. Figure 7-3 to Figure 7-8 show the interconnections for the three trigger modes.

The signal generator derives its 1PPS output signal from an internal frequency reference. Use the EXT REF IN connector on the rear panel of the GSS6560 signal generator (or the superseded STRn7n0 signal generator) to lock the internal frequency reference to an external 10 MHz reference. The external reference frequency must be within ± 0.01 ppm of nominal or the internal reference will fail to lock.

If the remote system cannot accept, and synchronise to, the 1PPS output from the signal generator; an alternative method is for the remote system to generate a 10 MHz reference signal and derive a 1PPS signal from it. Connect these signals to the appropriate connectors on the rear panel of the signal generator (EXT REF IN for 10 MHz, SYNC IN (GSS6560 1PPS IN) for 1PPS). The superseded STRn7n0 synchronises to both these inputs. You can start a remote run using any of the three trigger modes; however, timing is referenced to the remote system.

The remaining connections between the remote system and the signal generator vary depending on the method of starting the simulation run, the trigger mode, and the synchronisation method. For clarity, only the connections relevant to synchronisation and triggering are shown. Refer to the interconnection diagrams supplied with the signal generator for details of the remaining connections, such as links and terminations.

7.2 Motion data alignment

The connection from the remote system to the host PC is a fundamental requirement for the closed-loop mode. Command and Motion Data delivery requires it is always present. You can use either IEEE-488 or TCP/IP (Ethernet) communication and SimGEN supports the SCRAMNet shared memory interface system.

TCP/IP has the advantage of simplicity. Ethernet ports are available on PCs and no special software is required. However, the main disadvantage of the TCP/IP protocol is that it is fundamentally non-deterministic. You can avoid the timing uncertainties arising from conflicts by using a dedicated network, but there remains a problem with small messages. The Nagle Algorithm, RFC 896, of the TCP/IP protocol avoids network congestion by buffering small messages. The transmitting PC may hold a small message for up to 200 ms on the assumption that a further small message may follow. Congestion can be reduced by sending the small messages together in a larger group message. This effect can become apparent during scenario preparation, which requires transmission of a sequence of short commands.

Motion data messages are termed large messages and transmitted without delay. You can successfully use the TCP/IP protocol at the higher 100 Hz data rate.

The IEEE-488 protocol is free of the delays that may arise with TCP/IP. Although the data transmission speed is much lower, it remains sufficient for SimREMOTE based systems.

Note: *Spirent recommends the IEEE-488 communication method for most applications.*

For the remote closed loop motion option to run correctly, it is essential the SimGEN PC receives motion data updates sufficiently in advance of the Time of Applicability so the signal generator can be effectively controlled.

The motion data defining the motion for a time step needs to be available to SimGEN when the control data is calculated for that time step. There is some slight variability in this timing due to the tasks being performed and the performance of the PC. Typically, the calculation commences during the first millisecond of the prior time step. For example, using 10 ms iteration rate, the computation of the control data for 1210 to 1220 ms is performed at 1201 ms into the run. This means the motion data for 1210 to 1220 ms must be delivered to the SimGEN PC no later than the 1199 ms of the run.

Note: *The remote command OFFSET_PROC_TIME (section 5.3.13) delays processing and allows lower latency.*

SimGEN needs to maintain synchronism with the signal generator to deliver control and motion data at the correct time. To do this, SimGEN polls a 'simulation time into run' value from the signal generator using the IEEE-488 or USB bus. No other signal is required for SimGEN to maintain synchronism with the signal generator. SimGEN automatically compensates for variations or drift in the PC's internal clock relative to the signal generator's high precision clock.

Note: SimGEN requires better timing control for the 4 ms iteration rate, so the signal generator hardware generates a 1 ms interrupt.

SimREMOTE provides synchronisation signals from the PCI-215 timer card that allows an application running on the remote PC to keep accurately in step with the signal generator and SimGEN. SimREMOTE supplies motion data at the necessary rate, within the required timing window and sends messages to control the progress of the simulation and induce simulated events, such as satellite signal errors.

7.3 Software start-up - trigger Disabled

Notes:

1) If an external 1PPS is used to synchronise the system, the SYNC input is enabled only when the signal generator is not running.

During the run, supplying the signal generator with the same frequency reference used to derive the 1PPS SYNC signal ensures it remains in synchronisation with the remote system and eliminates any possibility of re-synchronisation occurring.

2) The timer outputs of the GSS8000-series signal generator (and the superseded GSS77xx/STR47xx signal generators) are programmable.

This document uses the factory default settings for the GSS77xx signal generator.

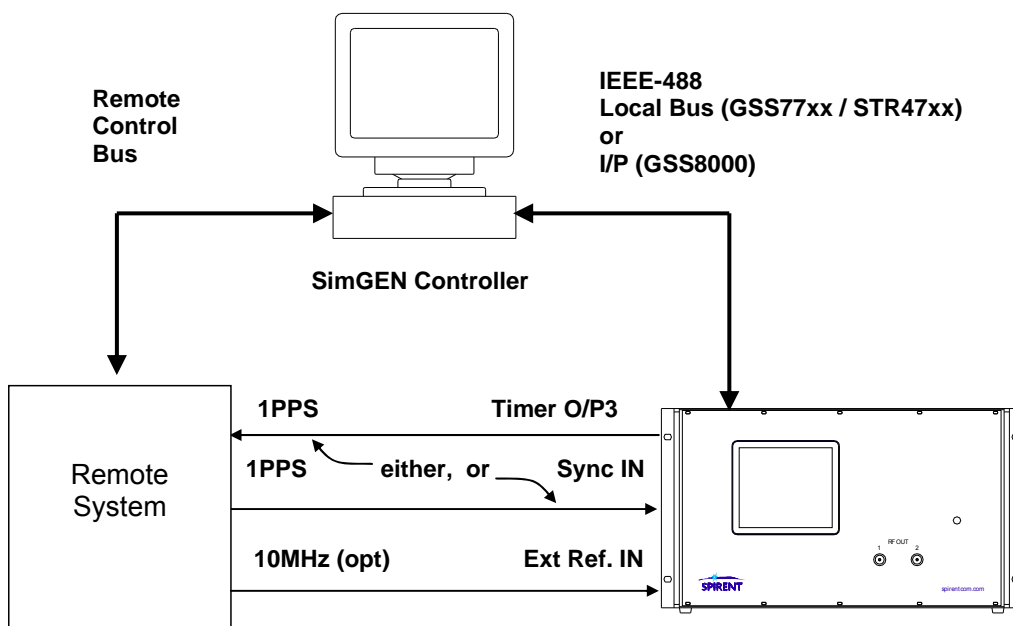
Timer Output 1 is an additional un-gated 1PPS.

Refer to section 7.8 for details on changing the settings of the timer outputs.

The default trigger mode is Disabled; however, Spirent advises you set this explicitly with the command **TR,0**, (see section 5.2.7).

To use trigger mode Disabled, connect your system as in Figure 7-3

Figure 7-3: System interconnection - trigger Disabled

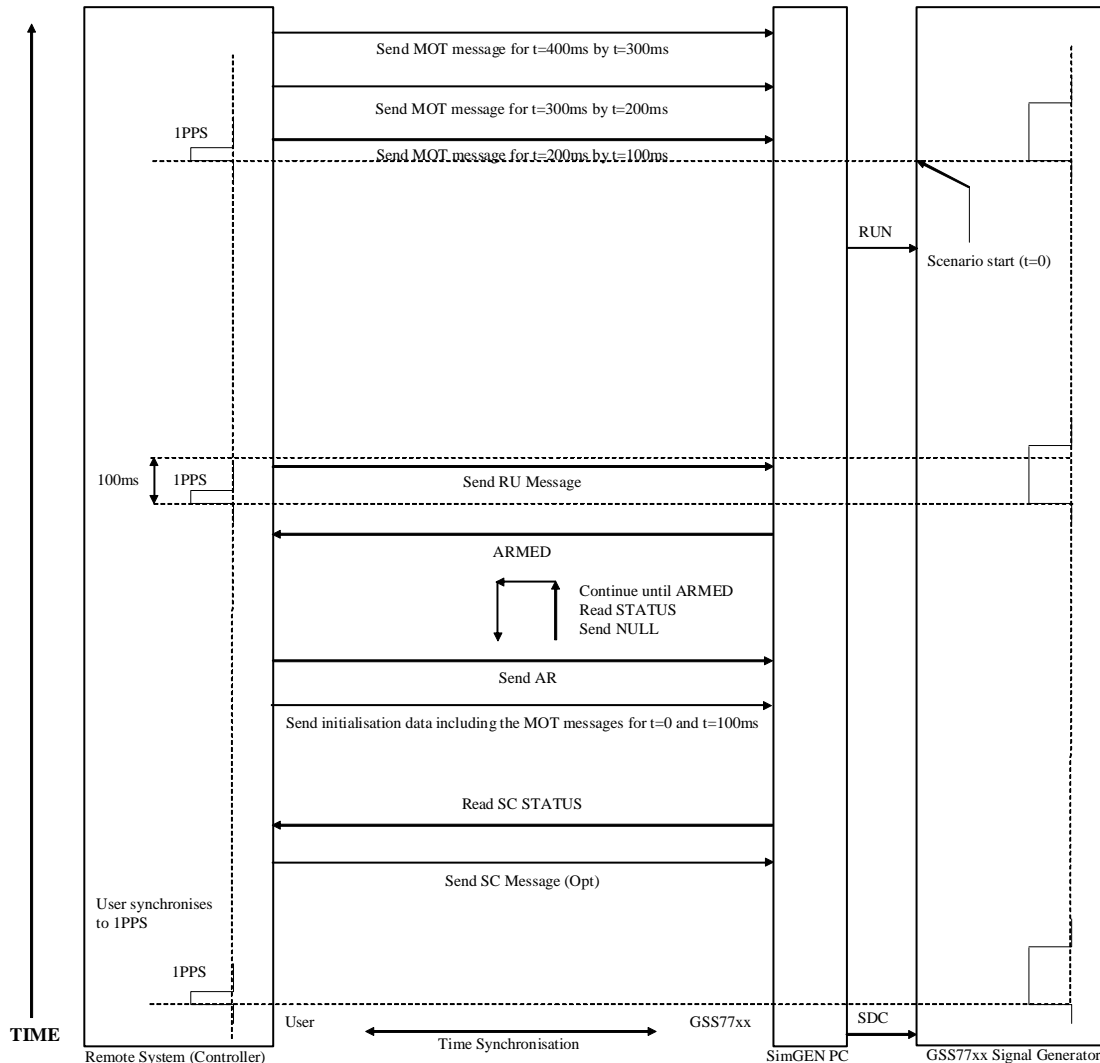


In the simplest implementation of closed-loop operation, the clock in the remote system uses the 1PPS signal from the signal generator (Timer Output 3 on a GSS8000-series GSS77xx or the

superseded STR47xx signal generator, or 1PPS Out on a GSS6560) to re-synchronise every second. Figure 7-4 shows the timing sequence.

The local clock in the remote system, or the PCI-215 Timer Card millisecond counter, determines the exact time or the 100 ms epochs within each one-second period.

Figure 7-4: Software start-up sequence



After transmission of initialisation data, the command **AR** is issued and the simulation status is continually polled (by sending a **NULL** command) until the status changes from ARMING to ARMED. Then, the simulation run is started by sending the **Run** message (see Section 5.2.8) to the SimGEN controller.

The start-up procedure is:

- Select and run Remote simulation on the SimGEN controller.
- The remote system sends initialisation data and motion data messages for $t = 0$ and $t = 100$ ms to the SimGEN controller.
- Send the **AR** command to arm the signal generator.
- The remote system reads the PC status message until the signal generator is ready to start (ARMED).
- The remote system sends the **Run** message immediately after a 1PPS pulse (within 100 ms).

- f) The SimGEN controller ends a RUN message to the signal generator using its local bus, so simulation will start on the next 1PPS rising edge.
- g) In response to the 1PPS pulse generated by the signal generator, the remote system sends the motion data message for $t = 200$ ms, followed by consecutive motion data messages at 100 ms intervals (timed with respect to 1PPS) until the end of the simulation.

Figure 7-4 shows this process.

7.4 External start pulse - Immediate trigger mode

Notes:

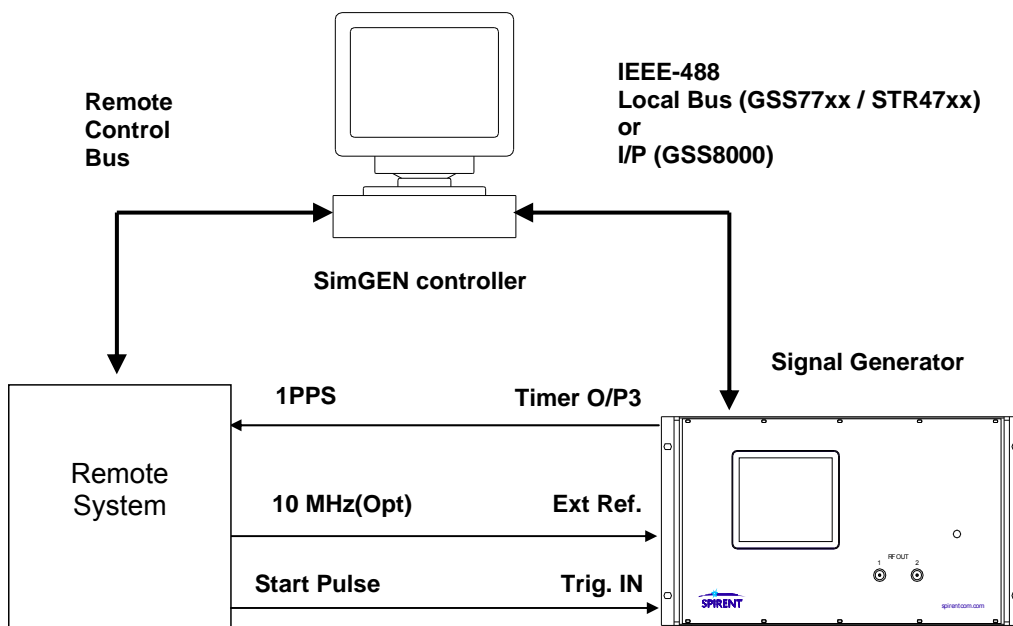
1) In this mode, applying an external 1PPS signal using SYNCIN is of no value, because the initiation of the Internal 1PPS is referenced to the start pulse.

2) The signal generator does not sense the 1PPS input during a simulation and when you use Immediate trigger mode you can leave an external 1PPS signal applied to the signal generator. The signal generator generates a 1PPS signal, referenced to the trigger pulse and not to your external 1PPS. At the end of the run, the signal generator re-synchronises to your 1PPS.

In Immediate trigger mode, the signal generator's internal 1PPS signal remains halted until a rising edge from the remote system appears at the TRIG IN socket. The 1PPS signal starts after a delay of 400 to 600 ns and simulation begins on its rising edge.

Figure 7-5 shows the system interconnections.

Figure 7-5: System interconnection – Immediate trigger mode



The start-up procedure is:

- a) Select and run Remote simulation on the SimGEN controller.
- b) The remote system sends a **TR,1** command (set Immediate trigger mode) to the SimGEN controller
- c) The SimGEN controller relays this command to the signal generator, halting the signal generator's 1PPS.
- d) The remote system sends initialisation data and motion data messages for $t = 0$ and $t = 100$ ms to the SimGEN controller.

- e) The remote system sends the **AR** (arm) command.
- f) The SimGEN controller initialises the signal generator.
- g) The remote system reads the SimGEN controller status message to determine when the signal generator is ready to start (ARMED).
- h) The remote system sends a **RUn** command to the SimGEN controller and SimGEN moves into a Trigger Wait state,
- i) The remote system sends a rising edge trigger directly to the signal generator.
- j) The first 1PPS rising edge that marks the start of the simulation occurs 400 to 600 ns later

Simultaneously with the start pulse, the remote system sends the motion data message for $t = 200$ ms, followed by consecutive motion data messages at 100 ms intervals (timed with respect to 1PPS) until the end of the simulation.

Section 7.11 gives details of the optional Timer Cable Assembly used to connect the remote system to the signal generator.

7.5 Hardware start pulse - Delayed trigger mode

Notes:

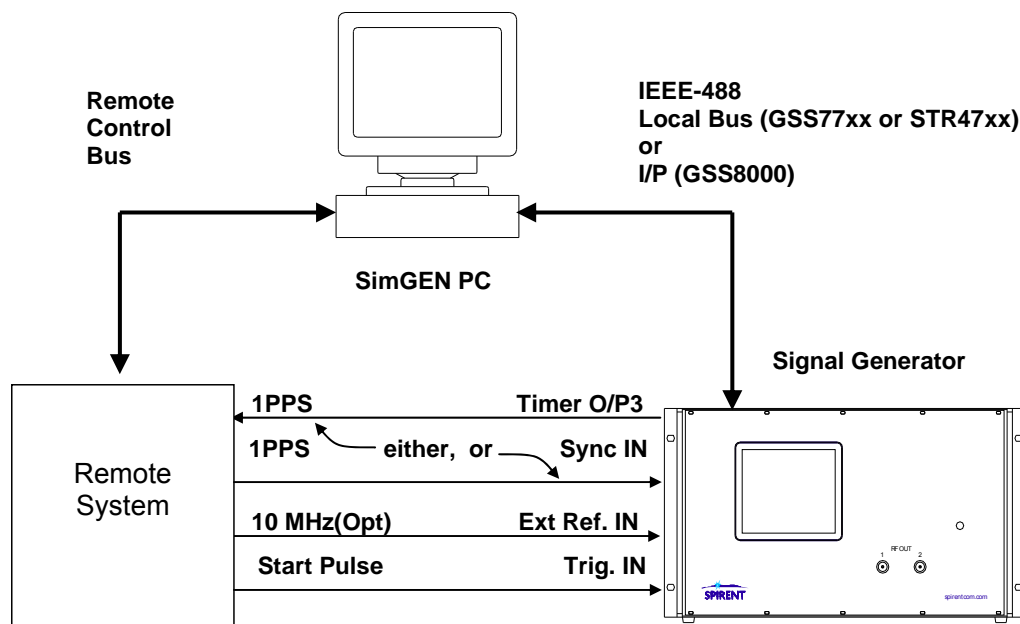
1) You can use the timer in the signal generator as the reference, or lock it to the external reference (10 MHz) and 1PPS, as described in section 7.3; and see item g) in the previous list.

2) For dual GSS8000, GSS7700 and STR4760 systems, and systems utilising a GSS4767 Distribution Unit, refer to section 7.7.

This is similar to the Disabled trigger mode, section 7.3. The start-up sequence is the same as in Immediate trigger mode, see section 7.4; except a **TR,2** command is sent during initialisation. You must apply a pulse to TRIG IN immediately following a 1PPS rising edge. The first motion data message (for $t = 200$ ms) is sent at the start time of the scenario.

Figure 7-6 shows the system interconnections.

Figure 7-6: System interconnections - Delayed trigger mode



The start-up sequence is:

- a) Select and run Remote simulation on the SimGEN controller.
- b) The remote system sends initialisation data and a **TR,2** command during initialisation.
- c) The remote system sends motion data messages for $t = 0$ and $t = 100$ ms to the SimGEN controller.
- d) The AR command arms the signal generator.
- e) The remote system reads the SimGEN controller status message until the signal generator is ready to start (ARMED).
- f) The remote system sends the RU message.
Note: You can send this message at any time.
- g) SimGEN enters the Trigger Wait state.
- h) Start the simulation by applying a rising edge to TRIG IN.
Note: The rising edge you apply to TRIG IN must be at least 100 ms before the rising edge of the 1PPS on which you want to start the simulation and after the previous 1PPS.
- i) The scenario starts on the following 1PPS.
- j) The motion data message for $t = 200$ ms is sent at the start time of the scenario, followed by consecutive motion data messages at 100 ms intervals (timed with respect to 1PPS) until the end of the simulation.

Section 7.11 gives details of the optional Timer Cable Assembly used to connect the remote system to the signal generator.

7.6 Using reduced data rates

Note: Reduced data rates compromise accuracy.

While all the trigger schemes specify motion data at a 100 ms rate, the software will handle data at lower rates that are multiples of 100 ms, or handle occasional missing blocks, by extrapolating the data.

7.7 Multiple unit configurations

Notes:

- 1) The GSS8000, GSS77xx series and STR47xx series timer outputs are programmable. You must correctly connect and configure the time outputs, see section 7.8.1.
- 2) The timer output settings can be different for each CONF setting, see section 7.8.1.
- 3) Switch on the Master unit (and the GSS4767 Distribution Unit if present) before the Auxiliary unit(s).

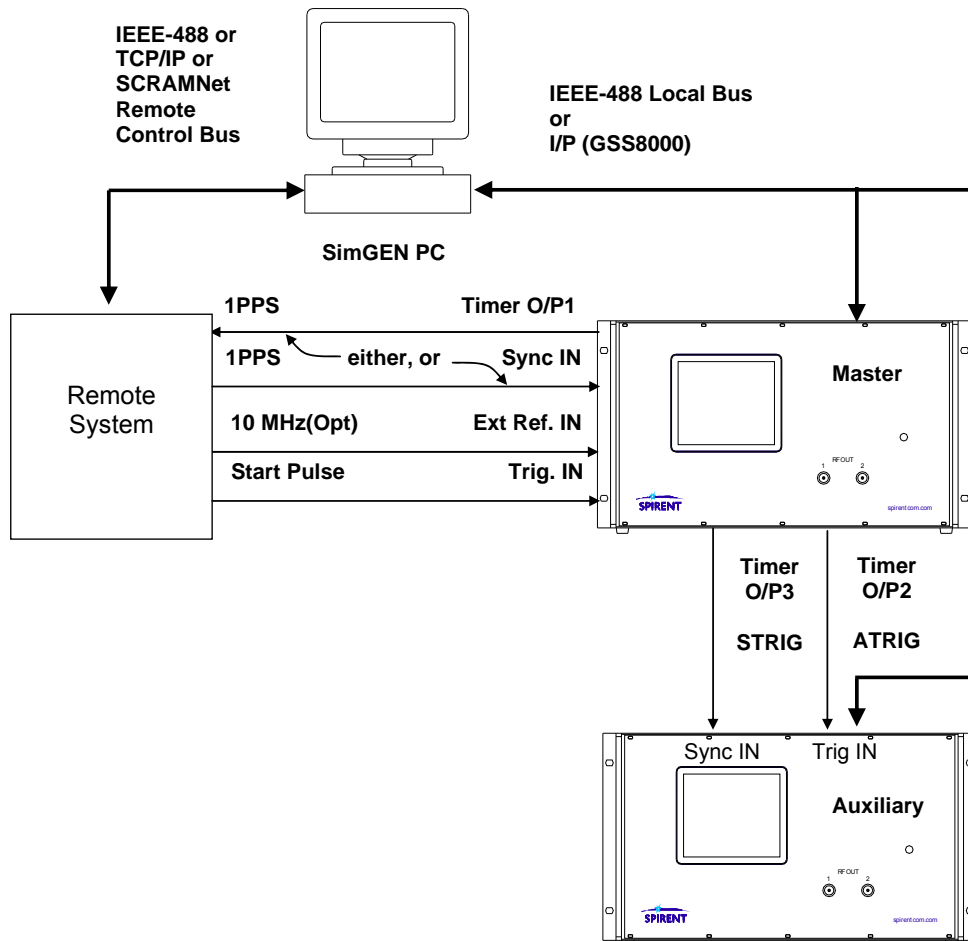
The interconnections required for multiple unit configurations are given below.

For clarity, only the connections relevant to synchronisation and triggering are shown in the following figures. Refer to the interconnection diagrams in reference c) for details of the remaining connections.

7.7.1 Two GSS8000 or a STR4780 and a GSS77xx / STR4760

You can use two GSS8000-series, GSS77xx, STR47xx signal generators, or an STR4780 and a GSS7700/STR4760 in trigger modes Immediate or Delayed if you connect and configure them as shown in Figure 7-7.

Figure 7-7: Interconnections for two signal generators



The STRIG setting of Timer Output 3 propagates the trigger to the Auxiliary signal generator in Immediate trigger mode, and synchronises the timers of the two signal generators when in Delayed trigger mode or the trigger is Disabled. The ATRIG setting of Timer Output 2 propagates the trigger to the second signal generator in Delayed trigger mode. You can omit this connection if you do not require Delayed trigger mode.

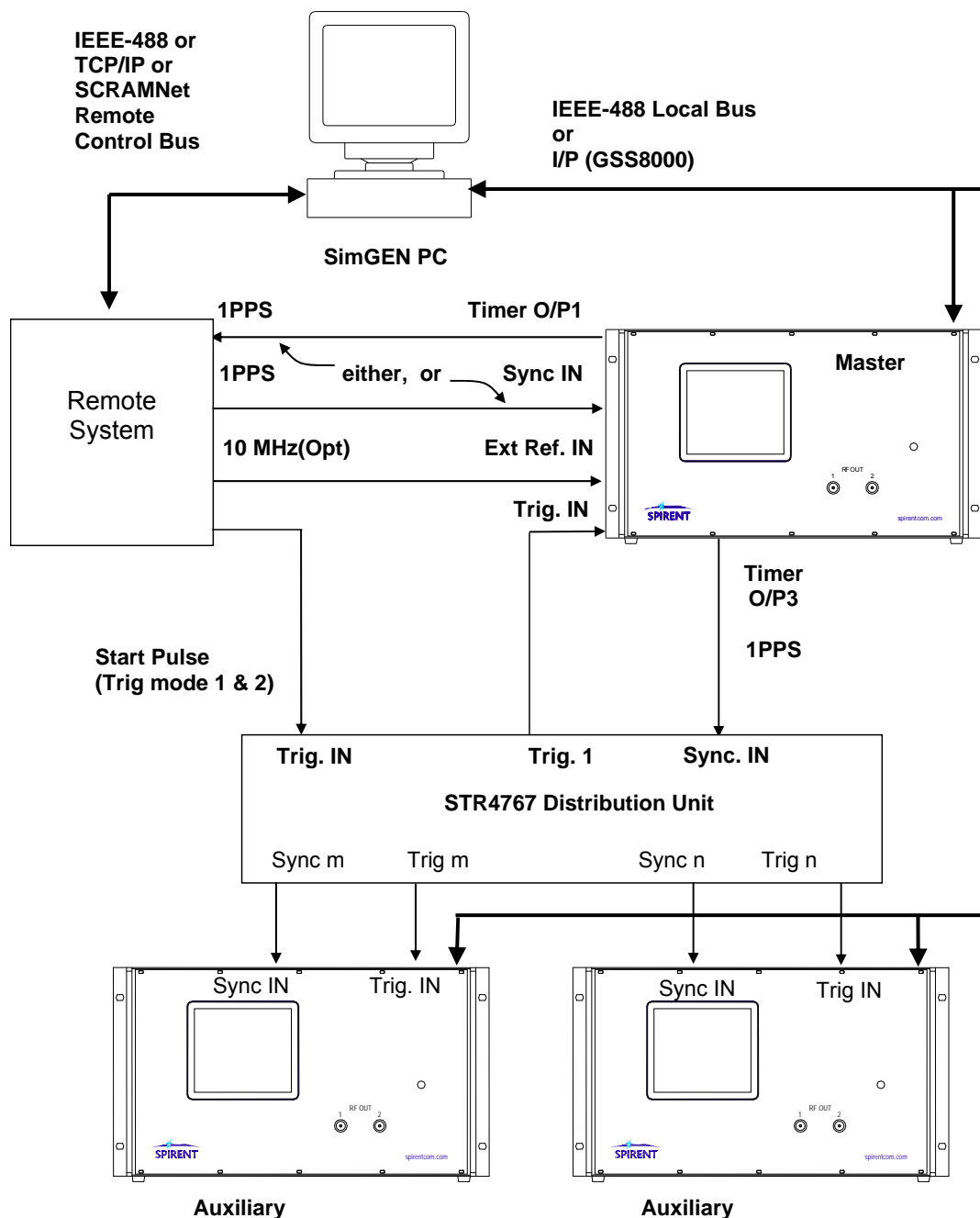
Section 7.11 gives details of the optional Timer Cable Assembly used to connect the remote system to the signal generator.

Note: Refer to section 7.5 before using Delayed trigger mode with a simulation iteration rate less than 10 ms.

7.7.2 Using the GSS4767 distribution unit

When you use a Distribution Unit to connect two or more GSS77xx or STR47xx signal generators, the Distribution Unit incorporates a facility for distributing the trigger and synchronising it to the system clock. This ensures all the signal generators start in synchronism. Figure 7-8 shows these interconnections (previously, the GSS4767 was marketed as the STR476).

Figure 7-8: Interconnections using a GSS4767 distribution unit



Section 7.11 gives details of the optional Timer Cable Assembly used to connect the remote system to the signal generator.

7.8 GSS77XX / STR47XX timer inputs and outputs

Note: The remote control connection from your system to the SimGEN controller is a fundamental requirement. Other connections to the system vary according to the synchronisation method and trigger mode you are using.

Figure 7-7 and Figure 7-8 show typical interconnections.

Table 7-4 shows the characteristics for I/O ports on the signal generator.

Table 7-4: Signal generator I/O port characteristics

Description	Comment
External 10 MHz signal	Connected to EXT REF IN on GSS7700/STR47n0
Input:	BNC socket, 50 Ohm impedance.
Signal level:	-10 dBm to +10 dBm
Waveform:	Sine or square. Frequency accuracy +0.01 ppm
Note: Configure using SimGEN	
External 1 PPS signal	Connected to SYNC IN on GSS7700/STR47n0
Input:	BNC socket
Signal level:	TTL into 50 Ohms -0.5 V < Low < +0.8 V, +2.4 V < High < +5.5 V
Pulse Width:	0.5 ms, minimum
Active edge:	Low to high transition indicates 1 s epoch in GPS time.
Start Pulse signal	Connected to TRIG IN on GSS7700/STR47n0
Input:	BNC socket
Signal level:	TTL into 50 Ohms -0.5 V < Low < +0.8 V, +2.4 V < High < +5.5 V
Pulse Width:	> 500 ns
Active edge:	Low to High transition
Description	Comment
Signal generator 1 PPS output	Timer Output 3 on GSS7700/STR47n0
Output:	BNC socket
Signal level:	50 Ohms TTL output -0.5 V < Low < +0.8 V, +2.4 V < High < +5.5 V
Pulse Width:	Positive going, 100 ms (GSS7700/STR47n0)
Active edge:	Low to high transition indicates 1 s epoch in GPS time

Section 7.11 gives details of the optional Timer Cable Assembly used to connect the remote system to the signal generator.

7.8.1 Timer output settings for remote operation

The functions of the GSS8000-series / GSS77xx / STR47xx rear panel timer outputs are user programmable. In addition to the basic timer outputs (such as 1PPS) a number of settings are available for cascading trigger signals between multiple signal generators when using remote trigger modes. Once configured, these settings are held in non-volatile memory and will remain until changed. See reference a) for details on how to change these settings.

The remote trigger settings STRIG/ATRIG/XTRIG and SYNC IN operation, as described in Table 7-5, are available from firmware version 4.20 onwards (MVME143 processor) and 4.20 onwards (MVME172 processor).

Table 7-5 shows the available timer settings.

Table 7-5: Available timer settings

Description	Comment
1PPS	High time 100 ms, period 1s. Rising edge synchronised with 1s epochs in simulated GPS time
UPDATE	High time 195.5ns (2 p-chips), Period: 1, 2, or 4 ms. All velocity changes occur at the mid point of this pulse
10PPS	10PPS Sync: High time 1 ms, period 100 ms. Rising edge synchronised with 100 ms epochs in simulated GPS time
100PPS	100PPS Sync: High time 1 ms, period 10 ms. Rising edge synchronised with 10 ms epochs in simulated GPS time
HIGH / PHIGH	Permanently high
PLOW / DISABLED	Permanently low
GATED	Any of the above signals may be “gated”, that is, present only when a scenario is running
STRIG	Identical to 1PPS, except in Immediate trigger mode if the unit is a master an additional pulse is produced on receipt of the trigger pulse 4 clocks before the 1PPS to trigger the auxiliary unit using the SYNC IN input.
ATRIG	In delayed and immediate trigger modes ATRIG transitions low-high on receipt of a trigger. Used to cascade a Delayed trigger mode trigger signal from a master unit to an auxiliary by connecting the ATRIG signal to the TRIG IN input of the auxiliary. Both STRIG and ATRIG connections may exist at the same time. When trigger is Disabled, this signal is permanently high.
XTRIG (only use for backwards compatibility with 2760 signal generators)	In trig modes 1 and 2 transitions low-high on receipt of trigger. In Immediate trigger mode it is timed to trigger an auxiliary STR2760 in synchronisation with the GSS7700/STR47n0 master unit (XTRIG signal applied to STR2760 TRIG IN input). If any output is set to XTRIG, the Immediate trigger mode pulse to start delay is increased by 200 ns. When trigger is Disabled, this signal is permanently high.
TPOP	A user-definable update pulse for use in SimINERTIAL implementations.

7.8.1.1 Consideration for 4 ms iteration rate

The 4 ms iteration rate uses one of the Timer Outputs set to GATED UPDATE to provide a 1 ms interrupt to SimGEN. Normally Timer Output 2 is used, but any available output may be used.

For example; a dual-box GSS7xxx-based simulator using Delayed trigger mode has Timer Output 2 on the Master signal generator normally set to ATRIG. In this case, you can use an unused output, such as Timer Output 2 on the Auxiliary signal generator, or Timer Output 1 on the Master signal generator.

7.8.2 Using the ‘sync in’ input

Notes:

1) Refer to reference c), for the timing of 1PPS IN and TRIG IN when you require precise synchronisation on 6560 signal generators.

2) All signal generators in the GSS77xx range and the superseded GSS47xx range sample SYNC IN and TRIG IN with a 10.23 MHz internal clock. This can result in an ambiguity in synchronisation of up to 98 ns.

3) *To prevent errors in the simulation caused by re-synchronising during a run, the SYNC IN input is internally disabled while the signal generator is running.*

To bring the signal generator timers into synchronisation with the remote system, you can supply a TTL-level, 1PPS Sync signal to the SYNC IN input of the master signal generator, when the external trigger is Disabled or when you are using Delayed trigger mode.

To keep the signal generator in synchronisation with the remote system, Spirent recommends that, if a 1PPS signal is supplied to SYNC IN, you should also supply an external reference source to the signal generator. This avoids the signal generator drifting in time with respect to the remote system over the duration of a run. If you do not supply an external reference source, or the 1PPS is not synchronous with the external reference, the BITE enunciator RESYNC EVENT will appear each time the external 1PPS causes the timer to resynchronise. This can occur occasionally while the signal generator is stopped, even with a synchronous 1PPS signal, as the required phase of 1PPS with respect to the external reference source is undefined so in certain circumstances a small amount of noise or drift can cause resynchronisation.

The SYNC IN input of the master signal generator is disabled for approximately three minutes after switching on to allow its reference oscillator to warm up and run close enough to nominal frequency to avoid constant re-synchronisation.

Contact Spirent for further information on synchronisation.

7.9 Synchronisation of GSS6560 inputs

Notes:

1) *Using TRIG IN (immediate mode) together with 1PPS IN is inappropriate, since both these inputs will attempt to control the timer.*

2) *TRIG IN (Delayed mode) can be used with 1PPS IN.*

The GSS6560 incorporates a number of input and output signal ports which you can use to synchronise time between the signal generator and your system. This section describes how to use the 1PPS IN and/or TRIG IN inputs to achieve synchronisation.

The GSS6560 maintains time internally by means of a time counter clocked by an internal 10 MHz clock. Simulations always start on a one-second rollover of this timer. Before starting a simulation, synchronise this timer to an external system by applying a rising edge to the 1PPS IN rear panel input. Now you can start simulations either by appropriate timing of the software run command (Disabled trigger mode) or by selecting Delayed trigger mode and applying a rising edge to the TRIG IN input. Both cause the simulation to start on the next one-second rollover of the timer. Alternatively, select Immediate trigger mode to force the timer to a point just before the one-second rollover and freeze it until a rising edge is detected on the TRIG IN input, the simulation starts running after a short delay.

If coarse synchronisation to your remote system is sufficient, these methods can be used with no additional considerations. However, certain fixed delays and uncertainties of the order of 100 ns will exist. For precise synchronisation, you must supply the signal generator with an external 10 MHz frequency reference and follow the timing requirements between the 1PPS IN and/or TRIG IN signals and the EXT REF IN signal detailed in sections 7.9.1 to 7.9.3.

7.9.1 1PPS IN

Notes

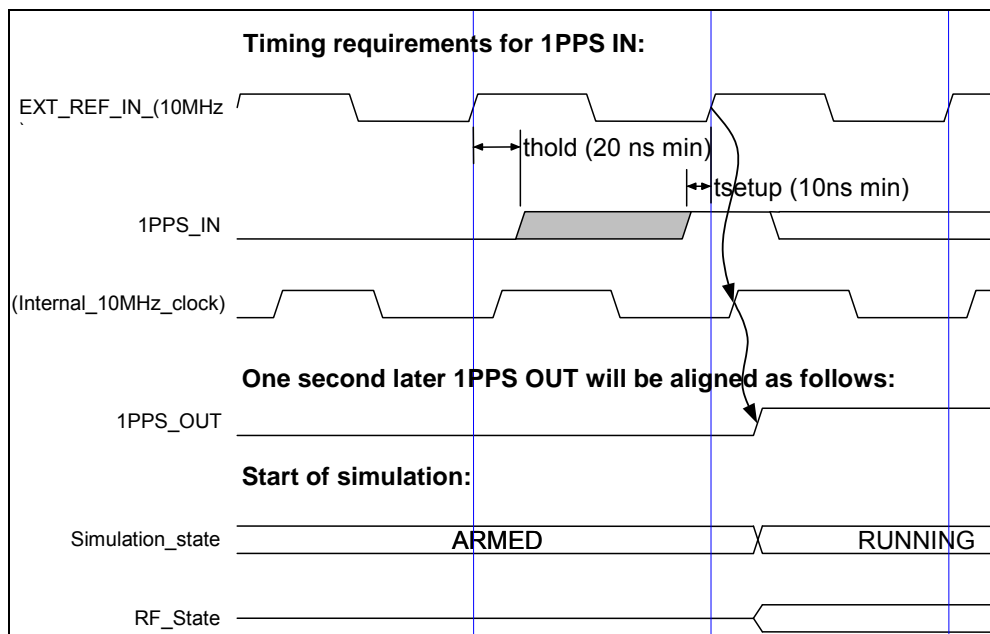
1) *Alignment of 1PPS OUT, as shown in Figure 7-9 occurs one second after detection of 1PPS IN and not immediately.*

2) *While a simulation is running, the 1PPS IN input is disabled.*

Figure 7-9 shows the required timing of the rising edge of 1PPS IN with respect to EXT REF IN, and the resulting timing of the start of simulation. Provided you meet these timing requirements, the RF signal timing will be fixed and repeatable, with respect to REF IN, every time you run a simulation.

The EXT REF IN signal may be a square wave as shown (for example a TTL/CMOS signal) or a sinusoid. Whatever the REF IN input waveform, the timing reference point is the ac zero crossing of the signal.

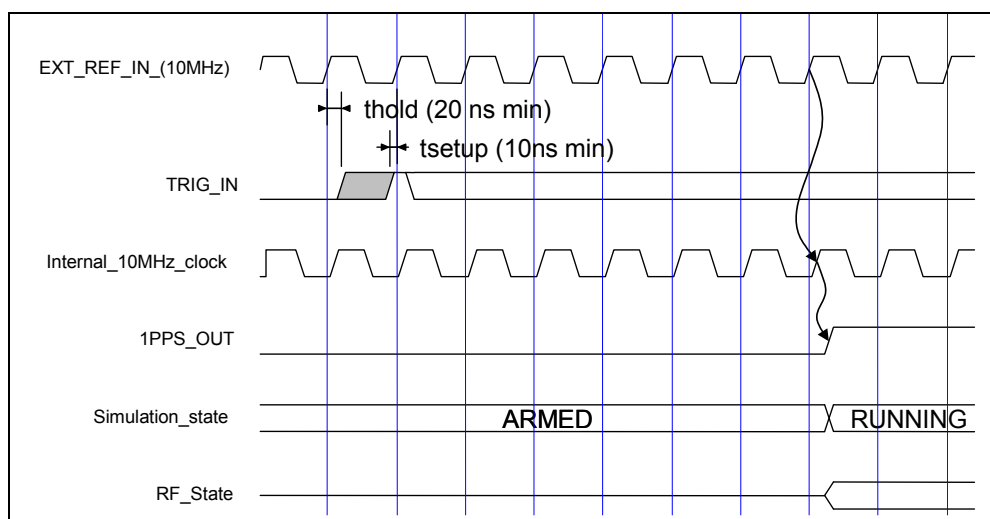
Figure 7-9: Timing requirements for 1PPS in and resultant start timing



7.9.2 TRIG IN - Immediate mode

When using the Immediate trigger mode, the timing requirements for the rising edge of TRIG IN with respect to EXT REF IN are the same as for the 1PPS IN input (that is 10 ns setup, 20 ns hold). However, there is a delay of six 10 MHz clock cycles after the recognizing the trigger, before the simulation starts, see Figure 7-10.

Figure 7-10: Timing requirements for TRIG IN (immediate mode) and resulting start timing



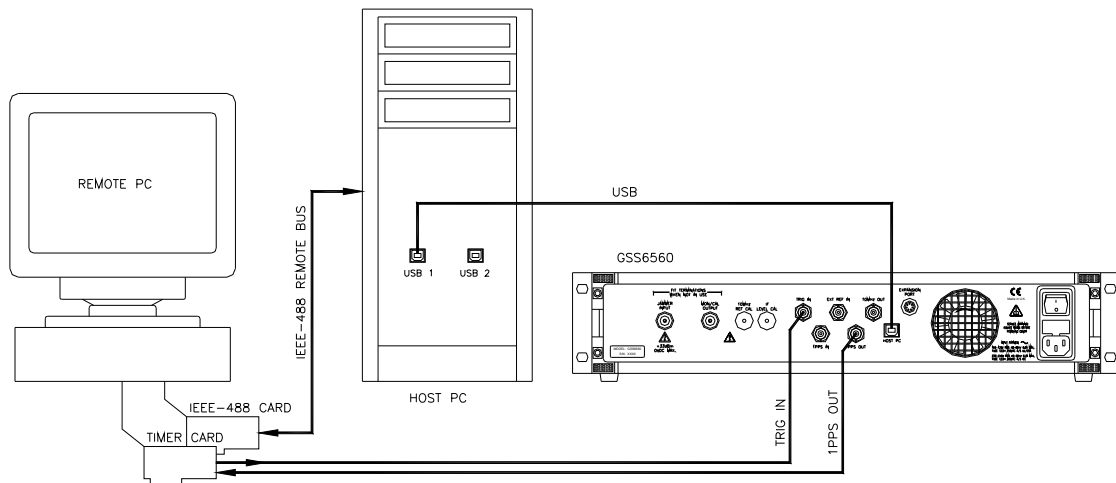
7.9.3 TRIG IN - Delayed mode

To start on a defined 1PPS event in Delayed trigger mode, the rising edge of TRIG IN must occur at least 1.1 milliseconds before the 1PPS event.

7.10 Configuring the GSS6560 for IEEE-488 remote control

Figure 7-11, shows how to configure the GSS6560 with a remote PC for remote control operation.

Figure 7-11: Configuring a GSS6560 for remote control operation



The remote simulation can be started in one of three ways:

- Trigger Disabled - software start
- Immediate - start immediately on detection of the external trigger pulse.
- Delayed - start on next 1PPS after detection of the external trigger pulse.

Select the mode you want to use using the remote command **TR,n**. If you need to, apply the external trigger pulse using the TRIG IN port.

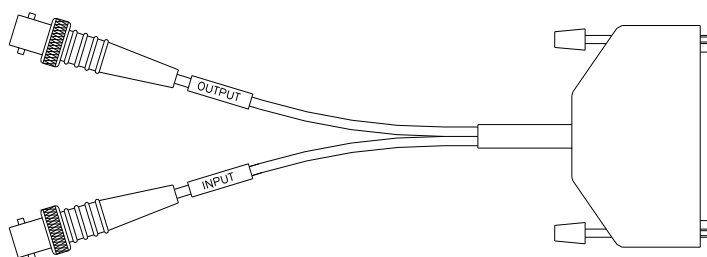
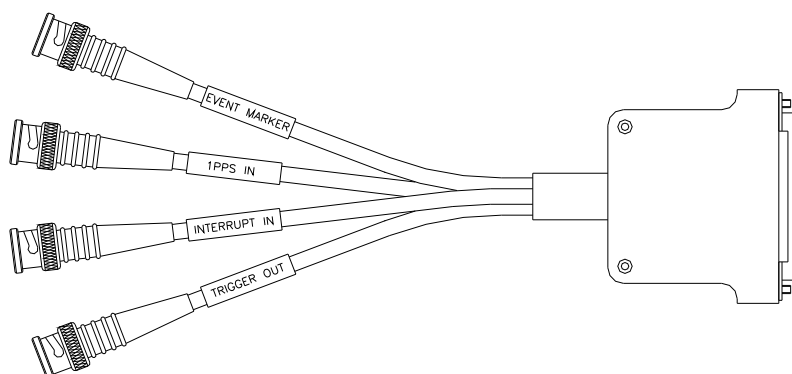
See section 7.9 for details on trigger mode timing constraints

7.11 Timer cable assembly (optional)

There are two styles of this optional cable assembly. Figure 7-12 shows the Old style cable assembly and Figure 7-13 shows the new style cable assembly, which provides an output pulse from the (optional) Timer card, see Table 7-6.

Table 7-6: Timer cable assembly - connector details

Label on OLD style cable assembly	Label on NEW style cable assembly	Description
(Cable not present)	EVENT MARKER	Provides timer card pulse when using TIMER_PULSE Command, see section 5.3.14
INPUT	1PPS IN	Connect to 1PPS timer output source
(Cable not present)	INTERRUPT IN	Used with SimINERTIAL and 250 Hz update rate, see section 7.8.1.1
OUTPUT	TRIGGER OUT	Connect to signal generator "Trig In" connector

Figure 7-12 Old style PCI Timer cable assembly**Figure 7-13 New style PCI Timer cable assembly**

Chapter 8: Data Streaming

Note: The Data Streaming facility supersedes the STR4761 Real Time Ethernet upgrade package supported in SimGEN for VMS.

8.1 Introduction

In addition to logging data for real time analysis in SimGEN, logging to file for post processing, or using the SimREMOTE facilities to request data from a remote controller, SimGEN lets you stream data over Ethernet using the UDP/IP transport protocol. You have control over where the data is sent, how fast the data is sent and what data is sent.

Section 8.3.3 describes Enhanced Data Streaming, which is only available for authorised users.

8.2 Data streaming definition file

The data streaming definition file holds the settings for the data streaming output. This is an XML formatted file. You can edit the file within SimGEN by selecting **Data streaming definition file** in the Scenario tree, see Figure 8-1. The **Data streaming definition file** has a default filename (extension *.dsd) you can change.

Figure 8-1: Data streaming definition file selection

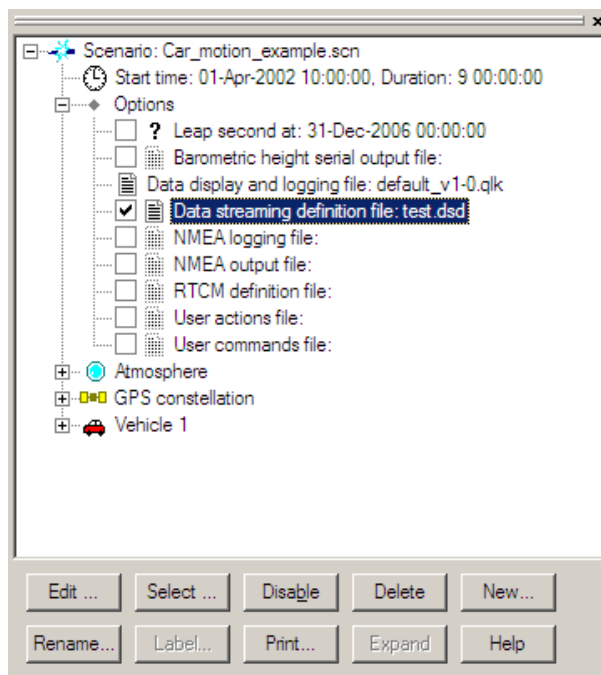
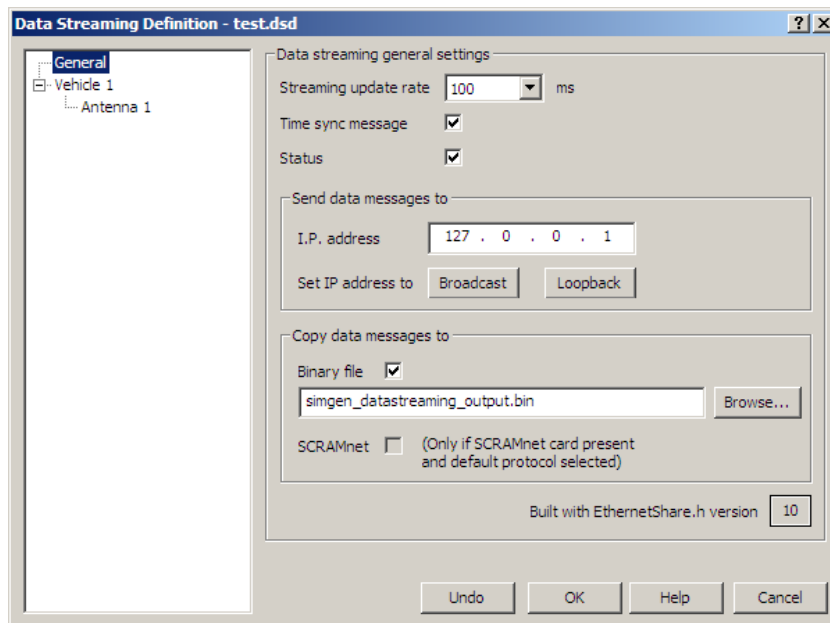


Figure 8-2 shows the **Data Streaming Definition** dialog.

Figure 8-2: Data streaming definition dialog**Streaming update rate**

Controls the update rate producing UDP/IP datagrams. If **Streaming update rate** is greater than the simulation iteration rate, the simulation iteration rate takes precedence. Some UDP/IP datagram rates are not controlled by this rate, such as the truth aiding EGI message and a note will be made by the message selection. Choose from the rates (in milliseconds) in the drop-down list.

Time sync message

Select (default): While a scenario is running, SimGEN sends a time sync message datagram at one-second intervals, starting at zero seconds into run. It helps the remote system synchronise to SimGEN's time.

The **time_into_run** parameter of the time sync message has a resolution of 1 millisecond and SimGEN determines this value at the point where it constructs the message. However, a variable amount of transmission delay is unavoidable.

The sync message also contains the start time, duration in seconds of the scenario and the GPS UTC offset in seconds.

Deselect: SimGEN does not send a time sync message

Notes:

1) Spirent advises you use the time sync message as a close approximation of the current time. This message can be late due to overheads from: processing, transmission of the UDP datagram and reading and interpreting the datagram.

The time sync message field time_into_run is the time when the message was constructed. Latency is the difference between this field and the time sync message field time_of_validity and tells you when the real 1 second epoch took place.

2) Running in Turbo mode requires extra processing overheads so time_into_run has no real meaning in this mode.

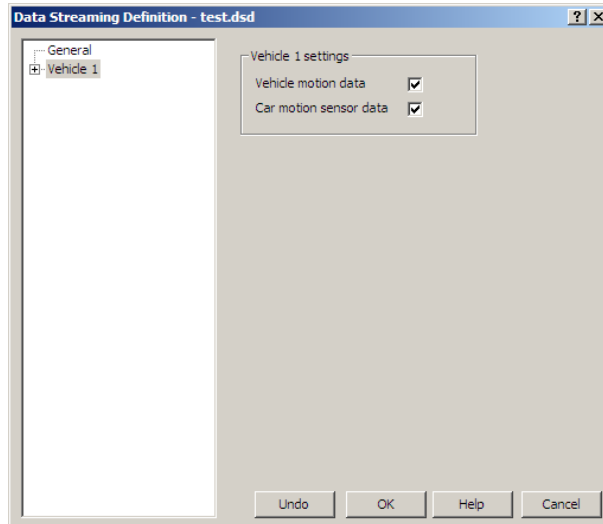
3) The latency between the time_into_run and time_of_validity has no bearing on any hardware timing.

Status	<p>Select (default): Sends a status message datagram whenever the status of SimGEN changes, such as changing state from READY to ARMING.</p> <p>The status datagram also contains the time step SimGEN is using (between 10 and 100 ms).</p> <p>If you have selected the truth aiding EGI message, SimGEN automatically selects status</p> <p>Deselect: Status message datagram not sent</p>
IP address	<p>Type the IP address to which you want to send the UDP/IP datagrams.</p>
Set IP address to	<p>Click on the appropriate button:</p> <p>Broadcast - The broadcast IP address collectively identifies all the interfaces on the local sub-network. The broadcast address is specified as 255.255.255.255.</p> <p>The network topology determines the range of interfaces that will receive the broadcast messages. In general the sub-network extent corresponds to the range of interfaces served by the closest router device. The router will pass broadcast messages to directly connected interfaces but will not pass them to another router.</p> <p>For details of the extent of your sub-network, and whether the target computers will be able to receive the broadcast messages, contact your network manager.</p> <p>All interfaces connected to the sub-network receive Broadcast Datagrams, including the sender.</p> <p>Loopback - The unicast address is a single IP address that identifies a single interface on a single machine.</p> <p>The localhost address is 127.0.0.1.</p> <p>Multicast</p> <p>Note: <i>Spirent has not tested data streaming with multicast addresses and the examples installed with SimGEN do not support this option.</i></p>
Binary file	<p>Select to copy data messages to a file.</p> <p>The default file <i>simgen_datastreaming_output.bin</i> is stored in the current scenario folder.</p> <p>Click Browse to navigate to a folder of your choice and type a filename. Click OK to save the file.</p>
Copy to SCRAMNet	<p>Disabled unless your SimGEN controller uses a SCRAMNet card, when all messages sent as UDP/IP datagrams are copied into a ring buffer in the SCRAMNet memory.</p>

8.2.1 Vehicle motion, antenna motion and signal data

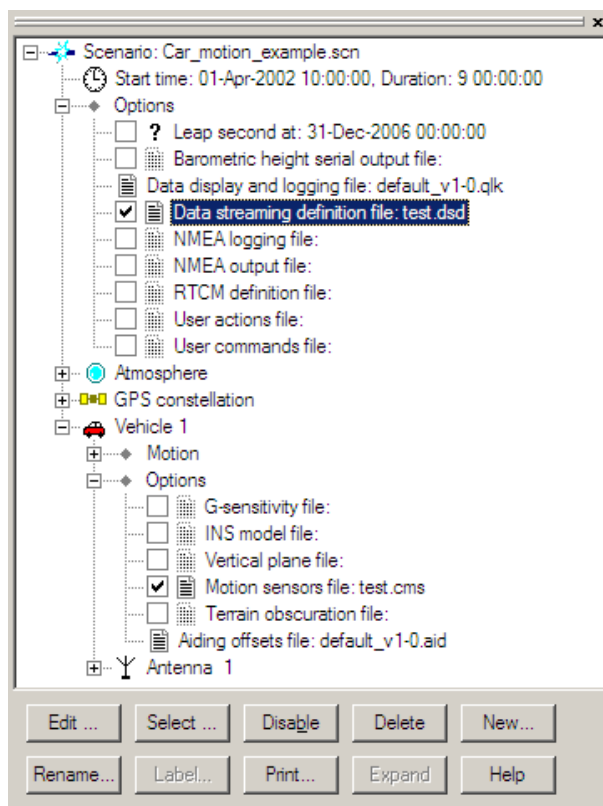
The vehicles and antennas listed in the data streaming definition settings dialog match the vehicles and antennas you define for the scenario. To stream the motion of a particular vehicle, click on the required vehicle in the **Data Streaming Definition** dialog and select **Vehicle motion data**, see Figure 8-3.

Figure 8-3: Data Streaming Definition - vehicle motion data dialog

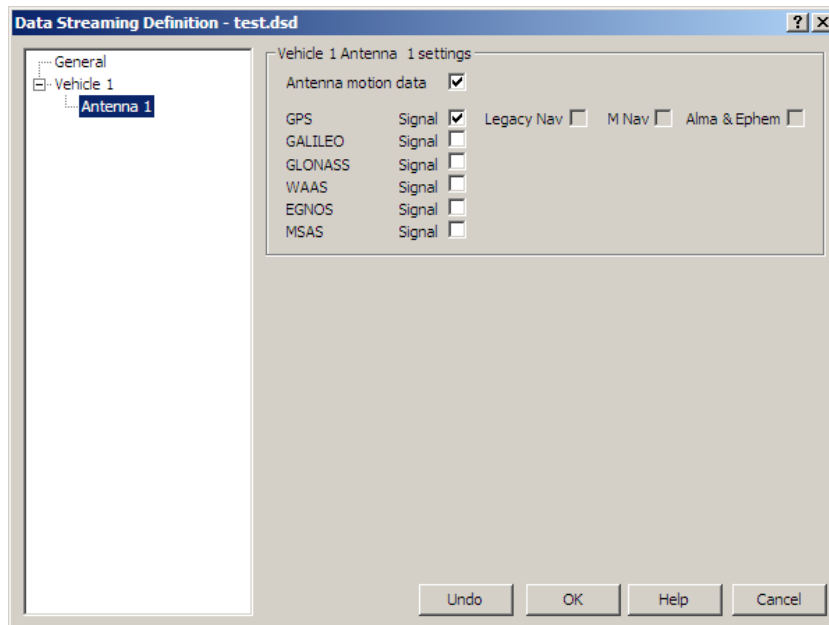


To stream data from the vehicle_n car motion sensor, select **Vehicle_n - Car motion sensor data**. You must select **Scenario - Vehicle_n - Options - Motion sensors file** and ensure the check box is ticked to enable **Car motion sensor data** see Figure 8-4.

Figure 8-4 Scenario Options to use Car motion sensor data



To stream the motion of a particular antenna, expand the vehicle and select **Antenna Motion Data**, see Figure 8-5.

Figure 8-5: Data Streaming Definition - Antenna_n settings dialog

For each antenna, select the signal type you want from:

GPS / GALILEO / GLONASS / WAAS / EGNOS / MSAS

With the correct dongle installed in the PC running SimREMOTE, authorised users will also be able to select from the enabled **Legacy nav**, **M Nav** and **Alma & Ephem**(eris).

The numbers of datagram streams generated by SimGEN depend on the selections within the data streaming definition file:

- One vehicle motion datagram stream is generated for each vehicle selected.
- One antenna motion datagram stream is generated for each antenna selected.
- One signal data datagram stream is generated for each signal type selected for an antenna.

These datagram streams are sent at the rate you select in **General-Streaming update rate**.

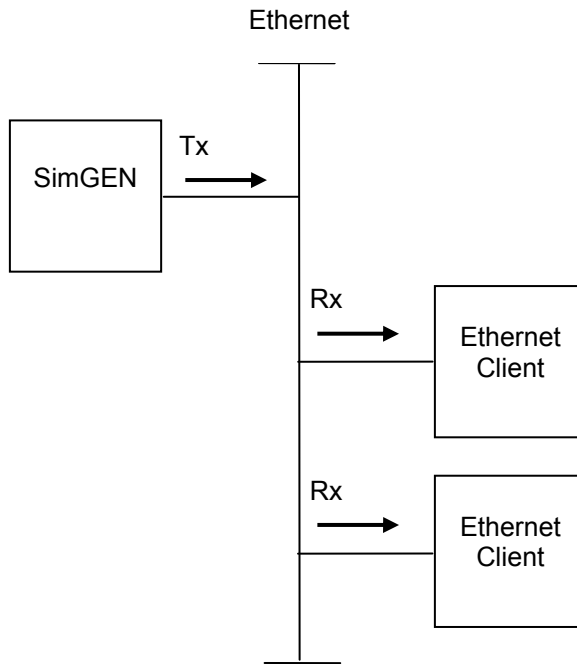
The status datagram messages are only sent when the SimGEN state changes.

One 'sync' datagram is sent every second.

8.3 Data streaming client

The UDP/IP datagrams target Data Streaming clients. It is your responsibility to implement Data Streaming clients. Spirent supplies example clients you can use as the basis of the user's client.

The client software may run on the same machine as SimGEN, normally the client software will run on a different machine. This machine should be at least as fast in processing network connections as the PC running SimGEN. The client machine does not necessarily need to be running Windows. However the example client Spirent supplies with SimGEN is a Windows program.

Figure 8-6: Ethernet client setup

8.3.1 Implementation details

Note: Currently SimGEN does not support *Space_Vehicle*

A C++ header file, ***EthernetShare.h***, is supplied with SimGEN, which, along with your own C/C++ code, allows you to implement data streaming clients. Section 8.3.2 describes the header file, which specifies:

- a) The datagram interface, defined as a union structure.
- b) UDP/IP port number.
- c) DLL interface specification.

All data is transmitted in little endian format aligned on an eight-byte boundary.

Sections 8.3.2.1 to 8.3.2.12 describe the sections of the header file. Section 8.3.3 describes features of Enhanced Data Streaming that are only available for authorised users.

A datagram comprises a fixed length section that identifies the message, followed by a variable length section that holds the message data. Section 8.3.2.7 defines the Message structure and reflects this requirement. It is a sequence of common declarations that define the fixed part and a union of different structures that define the various messages.

The common declarations for the fixed length part are:

type indicates the type of message and thereby which element of the union 'data' must be used to decode the message. Clearly it is critical that the correct message structure definition is used, for a particular message, otherwise the message content will be misinterpreted. The valid types are listed in the **Message::Type** enumeration.

version indicates the version number of the ***EthernetShare.h*** header file that should be used to decode the datagram. The datagram client should check this at least once per execution to confirm compatibility with the Issue of SimGEN generating the datagrams. Spirent advises you check the compatibility of the datastreaming DLL. From version number 3 of the DLL there is a function that returns the version number.

As outlined above, the **type** indicates which element of the **data** union describes the message body. The **data** union is an overlay of the Vehicle, Antenna, Signal, Truth_Aiding_EGI, Truth_Aiding (for legacy support), INS_Error_Model, Space_Vehicle, Car Motion Sensors, Status_info, M_Nav, L_Nav, IMU_Data and Sync message body structures.

8.3.2 EthernetShare.h explained

Note You need separately programmed dongles to use L Nav, M Nav, Almanac and Ephemeris and IMU.

This section explains the **EthernetShare.h** header file. You will need an understanding of C/C++ to use the **EthernetShare.h** header file.

The **EthernetShare.h** header file declares a namespace to avoid naming conflicts; this means that to use any declarations within the header file you must precede the declarations with the namespace EthernetData.

For example, to access the select_time declaration, you use EthernetData::select_time.

EthernetShare.h contains the following declarations:

```
static const unsigned int    select_time      = 1000;
static const unsigned int    version          = 10;
static const unsigned int    port             = 15660;
```

select_time is the time in milliseconds that elapse between sync messages being sent.

version is the version number that is sent within a datagram.

port is the port number that the client must bind to, to receive datagrams.

8.3.2.1 Vehicle structure

When **type** equals **EthernetData::Message::vehicle** use the union part **data.vehicle** with the following structure.

```

/*****
Vehicle Message Data
*****/
struct Vehicle
{
    unsigned int    vehicle;
    unsigned int    spare;
    double          posn           [ 3 ];
    double          velocity       [ 3 ];
    double          acceleration   [ 3 ];
    double          jerk           [ 3 ];
    double          latitude;
    double          longitude;
    double          height;
    double          attitude       [ 3 ];
    double          angular_velocity [ 3 ];
    double          angular_acceleration [ 3 ];
    double          angular_jerk    [ 3 ];
    double          baro_height;
    double          ground_speed;
    double          velocity_ned    [ 3 ];
    double          angular_velocity_inertial [ 3 ];
    double          angular_acceleration_inertial [ 3 ];
    double          angular_jerk_inertial [ 3 ];
};

```

Where:

vehicle is the vehicle number, starting from one

posn is the X,Y,Z position in ECEF

velocity is the velocity in the ECEF X,Y,Z axes with respect to ECEF frame, m.s^{-1}

acceleration is the acceleration in the ECEF X,Y,Z axes with respect to ECEF frame, m.s^{-2}

jerk is the jerk in the ECEF X,Y,Z axes with respect to ECEF frame, m.s^{-3}

latitude in rad

longitude in rad

height in metres (WGS84 Ellipsoid)

attitude in rad

angular_velocity is the angular velocities about the body axes (x,y,z) with respect to ECEF frame, rad.s^{-1}

angular_acceleration is the angular accelerations about the body axes (x,y,z) with respect to ECEF frame, rad.s^{-2}

angular_jerk is the angular jerks about the body axes (x,y,z) with respect to ECEF frame, rad.s^{-3}

baro_height is the barometric altitude in metres

ground_speed is the speed of the vehicle over the surface of the Earth, m.s^{-1}

local_velocity_ned, velocity with respect to ECEF and resolved in the local geographic frame.

angular_velocity_inertial is the angular velocity with respect to ECI, rad.s^{-1}

angular_acceleration_inertial, is the angular acceleration with respect to ECI, rad.s^{-2}

angular_jerk_inertial is the angular jerk with respect to ECI, rad.s^{-3}

Only one vehicle datagram per vehicle per streaming update is transmitted.

You must select **Vehicle-Options-INS model** to produce data for **angular_velocity_inertial**, **angular_acceleration_inertial** and **angular_jerk_inertial**. If you do not select **Vehicle-Options-INS model**, data for the three **_inertial** parameters is set to zero.

8.3.2.2 Antenna structure

When **type** equals **EthernetData::Message::antenna** use the union part **data.antenna** with the following structure:

```

/*****
*
Antenna Message Data
*****/
/
struct Antenna
{
    unsigned int    vehicle;
    unsigned int    antenna;
    double          posn           [ 3 ];
    double          velocity       [ 3 ];
    double          acceleration   [ 3 ];
    double          local_velocity_ned [ 3 ];
    double          latitude;
    double          longitude;
    double          height;
    double          dop;

    enum Type
    {
        gps           = 0,
        waas          = 1,
        egnos         = 2,
        msas          = 3,
        glonass       = 4,
        jammer        = 5,
        base_station  = 6,
        gsm           = 7,
        laas          = 8,
        galileo       = 9,
        ground_tx     = 10,
        qz            = 11,
        gps_2         = 12,
        gps_3         = 12,
        gps_4         = 14,
        gps_5         = 15,
        gps_6         = 16,
        gps_7         = 17,
        gps_8         = 18,
        gps_9         = 19,
        gps_10        = 20,
        gps_11        = 21,
        gps_12        = 22,

        max_type

    };
    unsigned int num_signals_of_type [ Antenna::Type::max_type ];
};

```

Where:

vehicle represents the vehicle number starting from one

antenna represent the antenna number starting from one

posn as X,Y,Z in ECEF

velocity is the velocity in the ECEF X,Y,Z axes with respect to ECEF frame, m.s^{-1}

acceleration is the acceleration in the ECEF X,Y,Z axes with respect to ECEF frame, m.s^{-2}

local_velocity_ned, velocity with respect to ECEF and resolved in the local geographic frame.

latitude in rad

longitude in rad

height in metres (WGS84 Ellipsoid)

dop (Dilution Of Precision) DOP type as selected for the current scenario

Only one antenna datagram per antenna per vehicle per streaming update is transmitted

num_signals_of_type is an array that carries the number of modelled satellites for a given type at the **time_of_validity** for an antenna. If antenna motion is used in conjunction with signal data this attribute will let you know how many signal data datagrams to expect of any one type at the **time_of_validity**.

8.3.2.3 Frequency structure

When type equals **EthernetData::Message::signal_data** use the union part **data.signal[n]** with the following structure:

```

/*****
*
Frequency Specific Data
*****/
/
struct Frequency
{
    static const int max_freq = 3;

    enum Type
    {
        gps_11,
        gps_12,
        gps_15,
        glonass_f1,
        waas_f1,
        waas_15,
        egnos_f1,
        egnos_15,
        msas_f1,
        msas_15,
        jammer_f1,
        base_station_f1,
        gsm_bcch,
        laas_vdb,
        galileo_11,
        galileo_e5,
        galileo_e6,
        ground_tx_f1,
        qz_f1,
        gps_2_11,    gps_2_12,    gps_2_15,
        gps_3_11,    gps_3_12,    gps_3_15,
        gps_4_11,    gps_4_12,    gps_4_15,
        gps_5_11,    gps_5_12,    gps_5_15,
        gps_6_11,    gps_6_12,    gps_6_15,
        gps_7_11,    gps_7_12,    gps_7_15,
        gps_8_11,    gps_8_12,    gps_8_15,
        gps_9_11,    gps_9_12,    gps_9_15,
        gps_10_11,   gps_10_12,   gps_10_15,
        gps_11_11,   gps_11_12,   gps_11_15,
        gps_12_11,   gps_12_12,   gps_12_15
    };

    enum Code
    {

```



```

        ca_code      =      1,
        p_code       =      2,
        y_code       =      4,
        m_code       =      8,
        mspot        =      16,
        mearth       =      32
    };
    Type              type;
    unsigned int      spare;
    double            pseudorange;
    double            pseudorange_rate;
    double            pseudorange_error;
    double            delta_range;
    double            iono_delay;
    double            signal_level;
    unsigned int      code_types; // composite of Code
    unsigned int      spare2;
};

```

Where:

type defines the signal type and frequency for which the frequency specific data applies, as defined by the enum “Type”.

pseudorange is the pseudorange code value for the relevant carrier frequency, m.

pseudorange_rate is the pseudorange-rate value for this signal component, m.s^{-1} .

pseudorange_error is the total non-deterministic pseudorange offset, that is this includes clock noise (ISCN model), clock divergence (delta af0 etc), pseudorange ramp error, remote offsets (Mod command), m.

delta_range is determined as the rate of change of pseudorange averaged over the data update interval, m.s^{-1} .

iono_delay is the pseudorange offset resulting from ionospheric delay, m.

signal_level is the received signal level (after applying antenna pattern gain), dB.

Note: The *pseudorange*, *iono_delay* and the *signal_level* are given as frequency specific. The different frequencies are L1CA, L2C and L5.

code_types defines the signal components that are valid for the given carrier frequency. This is a bitwise composite of the various code components as defined by the enum “Code”.

8.3.2.4 Signal structure

```

/*****
 *
Signal Message Data
 *****/
struct Errors
{
    double          divergence_terms;
    double          relativistic_effects;
    double          ramp_error;
    double          clock_noise;
    double          spare;
};

struct Signal
{
    unsigned int     vehicle;
    unsigned int     antenna;
    unsigned int     channel;
    unsigned int     satellite;
    Antenna::Type    type;
    unsigned int     prn;
    unsigned int     multi_path_index;
    unsigned int     spare;
    Frequency        frequency_specific [ Frequency::max_freq ];
    double           posn                [ 3 ];
    double           velocity            [ 3 ];
    double           acceleration        [ 3 ];
    double           elevation;
    double           azimuth;
    double           true_range;
    double           true_range_rate;
    double           tropo_delay;
    double           antenna_elevation;
    double           antenna_azimuth;
    Errors           errors;
};

```

Where:

The structure **Errors** includes the constituent elements of any applied pseudorange error in the signal as follows:

divergence_terms is the error resulting from clock divergence terms (delta af0 etc)

relativistic_effects is the pseudorange offset which results from the application of the relativistic clock correction as defined in ICD-GPS-200.

ramp_error is the pseudorange offset applied by activation of the SimGEN ramp/step facility.

clock_noise is the pseudorange error introduced when the intentional satellite clock noise model is enabled.

The structure **Signal** contains information which defines the characteristics of the RF signal for a simulated satellite as follows:

vehicle is the vehicle number to which the signal applies (starting from 1)

antenna is the antenna number to which the signal applies (starting from 1)

channel is the Channel number used for this signal (starting from 1)

satellite is the satellite id (SVID)

type where GPS = 0, WAAS = 1, EGNOS = 2, MSAS = 3, GLONASS = 4

prn is the satellite prn

multi_path_index where 0 is no multi-path and 1, 2,... represent the incident transmission plus 1, 2, ... multipath transmissions.

frequency_specific is a structure of type Frequency, as defined above, containing signal data for each used frequency, up to the maximum available for the given signal type.

posn is the satellite position at the time of signal transmission, expressed in the ECEF frame at the time of signal reception, m.

velocity is the satellite velocity in ECEF X,Y,Z axes with respect to the ECEF frame, m.s^{-1}

acceleration is the satellite acceleration in ECEF X,Y,Z axes with respect to the ECEF frame, m.s^{-2}

elevation is the satellite elevation with respect to the local geographic frame, rad.

azimuth is the satellite azimuth angle in the local geographic frame, rad.

true_range is the distance between the transmit and receive positions, m.

true_range_rate is the rate of change in range m.s^{-1}

tropo_delay is the pseudorange offset due to tropospheric delay, m

antenna_elevation signal arrival angle at antenna pattern, rad

antenna_azimuth signal arrival angle at antenna pattern, rad

errors is a structure of type Errors as defined above.

One datagram is produced at each streaming update for each satellite signal received by the antenna.

8.3.2.5 Truth_Aiding_EGI structure

```

struct Truth_Aiding_EGI
{
    unsigned int    vehicle;
    unsigned int    rate;
    double          acceleration_inertial    [ 3 ];

    // Jerk data is no longer needed so this
    // location can be used for baro data. However
    // we need to retain the old format for SimEGI
    union
    {
        struct old_egi_aiding old_format;
        struct new_egi_aiding new_format;
    } alt_data;

    double          posn_ecef                [ 3 ];
    double          velocity_ecef            [ 3 ];
    double          acceleration_ecef        [ 3 ];
    double          jerk_ecef                [ 3 ];
    double          angular_velocity_body_ecef [ 3 ];
    double          angular_acceleration_body_ecef [ 3 ];
    double          angular_jerk_body_ecef   [ 3 ];
    double          attitude                 [ 3 ];
};

```

Where:

vehicle represents the vehicle number starting from one

rate represents the simulation iteration rate of SimGEN (in the range 4 to 100 ms)

acceleration_inertial in ECI axes with respect to the ECI frame, m.s^{-2}

posn_ecef in ECEF axes with respect to the ECEF frame, m

velocity_ecef in ECEF axes with respect to the ECEF frame, m.s^{-1}

acceleration_ecef in ECEF axes with respect to the ECEF frame, m.s^{-2}

jerk_ecef in ECEF axes with respect to the ECEF frame, m.s^{-3}

angular_velocity_body_ecef in body axes with respect to the ECEF frame, rad.s^{-1}

angular_acceleration_body_ecef in body axes with respect to the ECEF frame, rad.s^{-2}

angular_jerk_body_ecef in body axes with respect to the ECEF frame, rad.s^{-3}

attitude in rad

Spirent's SimEGI and SimINERTIAL use this type of data. Enabling the SimGEN INS model file outputs the Truth_Aiding_EGI datagram.

To give backward compatibility with SimEGI, Spirent uses type equals

EthernetData::Message::truth_aiding_egi and the union part **data.truth_aiding_egi** with the following structure:

```

struct old_egi_aiding
{
    double          jerk_inertial[3];
};

```

Where:

jerk_inertial in ECI axes with respect to the ECI frame, m.s^{-3}

SimINERTIAL does not use `acceleration_inertial` and `jerk_inertial` parameters and Spirent uses the redundant `jerk_inertial` field to transmit Baro data.

```
struct new_egi_aiding
{
    double baro_height;
    double spare[ 2 ];
};
```

Where:

baro_height is the mean seal level (MSL) altitude inclusive of baro error modelling

8.3.2.6 Truth_Aiding structure

Note: Spirent provides this structure for legacy purposes.

```
/*
 *
 * Truth aiding at ins sensor offset
 */
/
struct Truth_Aiding
{
    unsigned int    vehicle;
    unsigned int    body_valid;
    unsigned int    local_valid;
    unsigned int    spare;
    double          latitude;
    double          longitude;
    double          height;
    double          velocity_body_inertial    [ 3 ];
    double          acceleration_body_inertial [ 3 ];
    double          jerk_body_inertial        [ 3 ];
    double          velocity_local_inertial   [ 3 ];
    double          acceleration_local_inertial [ 3 ];
    double          jerk_local_inertial       [ 3 ];
};
```

Where:

vehicle represents the vehicle number starting from one

latitude with respect to the WGS84 ellipsoid, rad

longitude with respect to the WGS84 ellipsoid, rad

height with respect to the WGS84 ellipsoid, metres

velocity_body_inertial body axes with respect to inertial frame, m.s^{-1}

acceleration_body_inertial body axes with respect to inertial frame, m.s^{-2}

jerk_body_inertial body axes with respect to inertial frame, m.s^{-3}

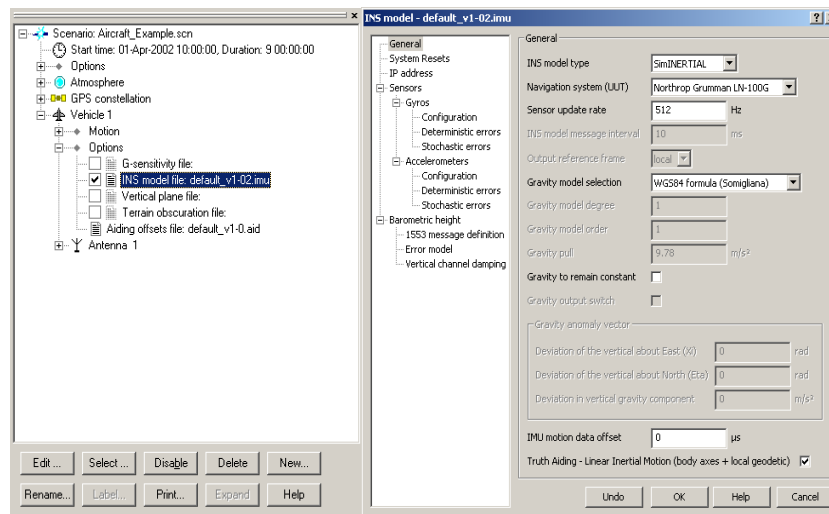
velocity_local_inertial local geographic axes with respect to inertial frame, m.s^{-1}

acceleration_local_inertial local geographic axes with respect to inertial frame, m.s^{-2}

jerk_local_inertial local geographic axes with respect to inertial frame, m.s^{-3}

Select **Truth Aiding-Linear Inertial Motion** in the **INS model-General** dialog to output the legacy Truth_Aiding datagram see Figure 8-7.

Figure 8-7 Using Truth_Aiding with SimEGI



8.3.2.7 INS_Error_Model structure

Note: SimGEN version 2-52, or higher, is required to access this data output.

This structure is used when you select the Space stabilised INS model.

When type equals **EthernetData::Message:: ins_error_model** use the union part **data.ins_error_model** with the following structure:

```

/*****
*
*   INS aiding error model
*****/
/
struct INS_Error_Model
{
    unsigned int    vehicle;
    unsigned int    spare;
    double          attitude                [ 3 ];
    double          latitude;
    double          longitude;
    double          height;
    double          velocity                [ 3 ];
    double          acceleration            [ 3 ];
    double          angular_rate            [ 3 ];
    double          angular_acceleration    [ 3 ];
    double          cnex                    [ 3 ];
    double          latitude_acceleration;
    double          longitude_acceleration;
    double          height_acceleration;
    double          baro_height;
    double          magnetic_heading;
};

```

Where:

vehicle represents the vehicle number starting from one

attitude, rad

latitude with respect to the WGS84 ellipsoid, rad

longitude with respect to the WGS84 ellipsoid, rad

height with respect to the WGS84 ellipsoid, m

velocity in local/body axes with respect to the ECEF frame, m.s^{-1}

acceleration in local/body axes with respect to the ECEF frame, m.s^{-2}

angular_rate rotation rate about body axes with respect to the ECI frame, rad.s^{-1}

angular_acceleration rotation acceleration about body axes with respect to the ECI frame, rad.s^{-2}

cnex direction cosine elements [X-X, X-Y, X-Z]

latitude_acceleration, rad.s^{-2}

longitude_acceleration, rad.s^{-2}

height_acceleration, m.s^{-2}

baro_height as modified by barometric model, m

magnetic_heading currently same as true heading, rad

8.3.2.8 Space_vehicle structure

Note: Currently not implemented

8.3.2.9 Sync structure

When the type equals EthernetData::Message::sync use the union part **data.sync_info** with the following structure:

```
struct UTC_params
{
    double      A0;
    double      A1;
    int         t_ot;
    int         wn_t;
    int         wn_lsf;
    int         dn;
    int         delta_t_lsf;
    int         delta_t_ls;
};

struct Sync
{
    char         start_time      [ 25 ];
    char         spare          [ 7 ];
    unsigned int  duration_seconds;
    int          utc_offset_seconds;
    unsigned int  GPS_time_s;
    unsigned int  GPS_week;
    unsigned int  GPS_tow_s;
    unsigned int  spare2;
    UTC_params    utc_params;
};
```

Where:

start_time is the start time as an ASCII string in the following format: DD-MMM-YYYY HH:MM:SS

duration_seconds is the duration of the scenario in seconds.

utc_offset_seconds is the UTC offset as defined in the GPS constellation file in seconds.

GPS_time_s is the GPS time in seconds

GPS_week is the GPS week number

GPS_tow_s is the time of week in seconds

One sync message is produced every second, including the start of the scenario where **time_of_validity** and **time_into_run** equal zero.

8.3.2.10 Car motion sensor structure

```

/*****
*
*   Car Motion Sensor Data
*****/
/
struct Car_Motion_Sensors
{
    enum AnalogueCard
    {
        max_num_outputs = 4
    };

    unsigned int    vehicle;
    unsigned int    odometer_enabled;
    unsigned int    gyrorate_enable;
    unsigned int    ratetable_enable;
    unsigned int    ratetable_speed_enable;
    unsigned int    can_enabled;
    unsigned int    gyrotemp_enable;
    unsigned int    ratetable_north_align;
    double          gyrorate_sens;
    double          gyrorate_offset;
    double          ratetable_sens;
    double          ratetable_offset;
    double          ratetable_speed_sens;
    double          ratetable_speed_offset;
    double          gyrotemp_sens;
    double          gyrotemp_offset;
    double          wheelbase;
    double          wheeltrack;
    double          deadband_threshold;
    double          ratetable_north_offset;
    double          cog_pulse_rate;
    double          front_odo_rate;
    double          rear_odo_rate;
    double          fl_wheel_rate;
    double          fr_wheel_rate;
    double          rl_wheel_rate;
    double          rr_wheel_rate;
    unsigned int     odos_all_same;

    unsigned int     log_cms_data;
    double           log_cms_data_every_ms;

    // The temperature model is: temperature starts at and stays
    // at 'start_temperature' for 'seconds_at_start_temperature'
    // seconds, then ramps slowly to 'steady_state_temperature', taking
    // another 'seconds_to_ramp_temperature' to do so. Then it stays
    // at this steady state temperature for the rest of the run.
    double           start_temperature;
    double           steady_state_temperature;
    double           seconds_at_start_temperature;
    double           seconds_to_ramp_temperature;
    double           ana_sig_map[ AnalogueCard::max_num_outputs ];

    // Other useful information about the car motion
    double           turn_radius;
    double           cog_speed;
    double           cog_accel;

```

```

double    latitude;
double    longitude;
double    height;
unsigned int    fwd_rev;
unsigned int    spare;
};

```

Where:

vehicle represents the vehicle number, starting from one

odometer_enabled true=>1, false=>0

gyrorate_enable true=>1, false=>0

ratetable_enable true=>1, false=>0

ratetable_speed_enable true=>1, false=>0

can_enabled true=>1, false=>0

gyrotemp_enable true=>1, false=>0

ratetable_north_align true=>1, false=>0

gyrorate_sens volts.rad⁻¹.sec⁻¹ about car vertical axis

gyrorate_offset voltage representing zero rate of change of heading

ratetable_sens volts.rad⁻¹ position about car vertical axis

ratetable_offset voltage representing zero heading

ratetable_speed_sens volts.rad⁻¹.sec⁻¹ about car vertical axis

ratetable_speed_offset voltage representing zero rotation rate

gyrotemp_sens volts.°C⁻¹

gyrotemp_offset voltage representing 0°C

wheelbase distance between centres of fore and aft axles, m

wheeltrack distance between centres of contact patches of left and right wheels, m

deadband_threshold minimum speed for wheel sensor output, m.s⁻¹

ratetable_north_offset offset in degrees of turntable zero position from true north

cog_pulse_rate pulses per km

front_odo_rate pulses per km

rear_odo_rate pulses per km

fl_wheel_rate pulses per km

fr_wheel_rate pulses per km

rl_wheel_rate pulses per km

rr_wheel_rate pulses per km

odos_all_same true=>1, false=>0

log_cms_data true=>1, false=>0

log_cms_data_every_ms true=>1, false=>0

start_temperature °C

steady_state_temperature °C

seconds_at_start_temperature s

seconds_to_ramp_temperature s

ana_sig_map V

turn_radius m

cog_speed m.s⁻¹

cog_accel m.s⁻²

latitude rad

longitude rad

height m

fwd_rev true=>1, false=>0

8.3.2.11 Status structure

When the type equals EthernetData::Message::status use the union part **data.status_info** with the following structure:

```
Struct Status
{
    enum Type
    {
        no_scenario_loaded    = 0,
        loading                = 1,
        ready                  = 2,
        arming                 = 3,
        armed                  = 4,
        running                 = 5,
        paused                 = 6,
        ended                  = 7
    };

    Type      status;
    int       simulation_update_rate; // 10 or 100 (milliseconds)
    int       UDP_output_rate_ms; // obviously in milliseconds
    unsigned int spare;
};
```

Where:

status represents one of the status types as shown in the above enum.

simulation_update_rate is set to the SimGEN simulation iteration rate (range 4 to 100 ms).

A status message is generated each time SimGEN changes state, for example from ARMING to ARMED.

8.3.2.12 Ethernet message structure

The Message structure is the main definition within the EthernetData namespace. The Message structure defines the various forms of datagram that SimGEN can transmit.

```

/*****
*
Ethernet Message Structure
*****/
/
struct Message
{
    /*****
    *****
    Type of Messages that can exist
    *****/
    enum Type
    {
        vehicle                = 0,
        antenna                 = 1,
        signal_data              = 2,
        truth_aiding              = 4,
        truth_aiding_egi          = 5,
        ins_error_model           = 6,
        space_vehicle             = 7,
        status                    = 8,
        m_nav_data                = 9,
        legacy_nav_data           = 10,
        keplerian_data            = 11,
        imu_data                  = 12,
        car_motion_sensor_data    = 13,

        // Always make sync the last one.
        sync                      = 29
    };
    enum
    {
        max_signal = 1
    };
    /*****
    *****
    Message Data
    *****/
    unsigned int    type;
    unsigned int    version;
    unsigned int    time_into_run;
    unsigned int    time_of_validity;

    union
    {
        Vehicle      vehicle;
        Antenna       antenna;
        Signal        signal[ max_signal ];
        Truth_Aiding  truth_aiding;
        Truth_Aiding_EGI truth_aiding_egi;
        INS_Error_Model ins_error_model;
        Space_Vehicle space_vehicle;
        Status         status_info;
        MNav           m_nav;
    }
}

```

```

        LNav                legacy_nav;
        IMU_data             imu;
        Keplerian            keplerian;
        Car_Motion_Sensors   cms;

        Sync                 sync_info;
    } data;
};

```

Where:

time_into_run indicates the time into the scenario the datagram was generated.

This time will precede the **time_of_validity** as datagrams are generated two scenario time steps in advance.

Units are in milliseconds.

time_of_validity indicates the time into the scenario at which the data in the datagram is valid.

Units are in milliseconds.

8.3.3 Enhanced Data Streaming

Note: Enhanced Data Streaming is only available to authorised users.

To use Enhanced Data Streaming, you must insert the dongle, supplied by Spirent to authorised users, in a USB port on the PC before you start SimREMOTE.

8.3.3.1 IMU_data structure

For more details, see the section on Aiding in reference a).

SimGEN generates data packets at the sensor update rate and transmits them at the simulation iteration rate. For example, if you set the sensor update rate to 500 Hz and set the simulation update rate to 100 Hz, SimGEN produces five IMU_data packets every 10 ms (that is, at the simulation iteration rate). The IMU_data packet contains a separate time_of_validity field that SimGEN time-stamps at 500 Hz epochs. As with all data streaming packets, SimGEN transmits each set of IMU_data packets in advance.

IMU (Inertial Measurement Unit) data represents changes in velocity and attitude of the inertial system. The double precision **time_of_validity** is used.

```

/*****
IMU data
*****/

struct IMU_data
{
    enum Sensors
    {
        max_sensors = 6
    };

    unsigned int vehicle; // 1 -> N
    unsigned int model_number; // INS model number 1 -> 4
    double time_of_validity;
    double delta_no_errors_velocity[ max_sensors ];
    double delta_no_errors_theta [ max_sensors ];
    double delta_errored_velocity[ max_sensors ];
    double delta_errored_theta [ max_sensors ];
};

```

Where:

vehicle is the vehicle number starting from one

antenna is the antenna number starting from one

time_of_validity is the actual time of validity of the data.

delta_no_errors_velocity is the perfect (no errors) change in body velocity, m.s^{-1}

delta_no_errors_theta is the perfect (no errors) change in body angle, rad

delta_errored_velocity is the change in body velocity, with errors modelled by SimGEN, m.s^{-1}

delta_errored_theta is the change in body angle, with errors modelled by SimGEN, rad

8.3.3.2 Keplerian structure

```

/*****
Keplerian
*****/
/
struct Almanac
{
    int            time_of_almanac_s;
    int            week_number;
    double         eccentricity;
    double         deviation_in_inclination_angle_sc;
    double         rate_of_right_ascension_scps;
    double         square_root_sma_sqm;
    double         longitude_acending_node_sc;
    double         argument_of_perigee_sc;
    double         mean_anomaly_sc;
    double         af0_s;
    double         af1_sps;
};

```

Where:

time_of_almanac_s is the time of almanac

week_number is the truncated almanac week number

eccentricity is the eccentricity of the orbit

deviation_in_inclination_angle_sc is the deviation in inclination angle, semi-circles

rate_of_right_ascension_scps is the rate of right ascension, semicircles per second

square_root_sma_sqm is the square root of the semi major axis, m,

longitude_acending_node_sc is the longitude of the ascending node, semi-circles

argument_of_perigee_sc is the argument of the perigee of the orbit, semi-circles

mean_anomaly_sc is the anomaly at reference time, semi-circles

af0_s is the zero order clock correction term, seconds

af1_sps is the first order clock correction term, seconds per second

time_of_almanac_s is the time of almanac

```

struct Ephemeris
{
    enum Amplitude_harmonic_index
    {
        cuc_rad = 0,
        cus_rad,
        crc_m,
        crs_m,
    }
};

```

```

        cic_rad,
        cis_rad,
        max_cxx
    };

    int          time_of_ephemeris_s;
    int          iode;
    double       mean_anomaly_sc;
    double       mean_motion_difference_scps;
    double       eccentricity;
    double       square_root_sma_sqm;
    double       longitude_acending_node_sc;
    double       inclination_angle_sc;
    double       argument_of_perigee_sc;
    double       rate_of_right_ascension_scps;
    double       rate_of_inclination_angle_scps;
    double       amplitude_harmonic_correction[ max_cxx ];
};

```

Where:

time_of_ephemeris_s is the time of ephemeris, seconds

iode is the issue of data ephemeris

mean_anomaly_sc is the anomaly at reference time, semi-circles

mean_motion_difference is the difference from the computed value

eccentricity is the eccentricity of the orbit

square_root_sma_sqm is the square root of the semi major axis, m,

longitude_acending_node_sc is the longitude of the ascending node, semi-circles

inclination_angle_sc is the inclination angle, semi-circles

argument_of_perigee_sc is the argument of the perigee of the orbit, semi-circles

rate_of_right_ascension_scps is the rate of change of the right ascension, semicircles per second

rate_of_inclination_angle_scps is the rate of change of the inclination angle, semicircles per second

amplitude_harmonic_correction[max_cxx] is the maximum of the harmonic correction terms

```

struct Ionospheric
{
    enum Alpha_index
    {
        alpha_0_s          = 0,
        alpha_1_spsc        = 1,
        alpha_2_spsc2       = 2,
        alpha_3_spsc3       = 3,
        alpha_max
    };

    enum Beta_index
    {
        beta_0_s           = 0,
        beta_1_spsc        = 1,
        beta_2_spsc2       = 2,
        beta_3_spsc3       = 3,
        beta_max
    };

    double                alpha_coefficients[ alpha_max ];
};

```

```
double      beta_coefficients [ beta_max];
};
```

Where:

alpha_coefficients is the alpha coefficients used in the ionospheric model

beta_coefficients is the beta coefficients used in the ionospheric model

```
struct Clock
{
    double      af0;
    double      af1;
    double      af2;
    double      toc;
    int         iodc;
};
```

Where:

af0 is the zero order clock correction term, seconds

af1 is the first order clock correction term, second per second

af2 is the second order clock correction term, second per (second)²

toc is the time of clock

iodc is the issue of data, clock

```
/******
 *
 * Almanac, Ephemeris, Ionospheric & Clock data
 * *****/
/
struct Keplerian
{
    enum Generation
    {
        from_computed_data = 0,
        from_nav_data= 1
    };
    unsigned int      vehicle;
    unsigned int      antenna;
    unsigned int      channel;
    unsigned int      satellite;
    unsigned int      prn;
    unsigned int      gps_time_s;
    unsigned int      gps_tow_s;
    unsigned int      week_number; // includes rollover
    Almanac            almanac;
    Ephemeris          ephemeris;
    Ionospheric         ionospheric;
    Clock              clock_terms;
    Generation         bit_mods;
    Antenna::Type      type;
};
```

Where:

vehicle is the vehicle number starting from one

antenna is the antenna number starting from one

channel is the Channel number starting from one

satellite is the satellite id (SVID)

prn is the satellite prn

gps_time_s is the GPS time in seconds

gps_tow_s is the GPS time of week

week_number is the untruncated GPS week number

SimGEN produces the Keplerian data streaming message at time 0, when a new satellite appears and then every 12 ½ minutes until the satellite disappears from view.

At time 0, and when a new satellite appears, the Keplerian data is derived from data computed by SimGEN and shown by the bit_mods field containing the value "from_computed_data". At the next 12 ½ minute epoch, the data is derived from the generated navigation data and shown by the bit_mods field containing the value "from_nav_data". Generated navigation data could contain user bit modifications and errors

Note: *The navigation data has been through a scaling process and is less precise than the computed data.*

8.3.3.3 L nav structure

```

/*****
*
Legacy Nav Data
*****/
/
struct LNav
{
    Unsigned int    vehicle;
    unsigned int    antenna;
    unsigned int    channel;
    unsigned int    satellite;
    unsigned int    prn;
    char            nav_bits[ 38 ]; // 300bits = 37.5 bytes
    Antenna::Type   type;
};

```

Where:

vehicle is the vehicle number starting from one

antenna is the antenna number starting from one

channel is the Channel number starting from one

satellite is the satellite id (SVID)

prn is the satellite prn

8.3.3.4 M nav structure

```

/*****
*
M Code Nav Data
*****/
/
struct MNav
{
    unsigned int    vehicle;
    unsigned int    antenna;
};

```

```

        unsigned int      channel;
        unsigned int      satellite;
        unsigned int      prn;
        char              nav_bits[ 50 ]; // 400bits
        Antenna::Type      type;
};

```

Where:

vehicle is the vehicle number starting from one

antenna is the antenna number starting from one

channel is the Channel number starting from one

satellite is the satellite id (SVID)

prn is the satellite prn

8.4 Implementing a client

There are two ways you can implement a data streaming client to receive SimGEN transmitted datagrams. You can utilise the DLL supplied by Spirent, see section 8.4.1, then it is only necessary to implement the controlling framework; or you can implement your own custom software using the details in section 8.4.3, which you can base on the Spirent example, see section 8.4.4.

The example source code below (supplied as a Microsoft Visual C++ project), the header file (**EthernetShare.h**), the DLL (**simgendatastreaming.dll**) and the example clients are supplied with SimGEN, in the pre-installed folder, **SimGEN/Data Streaming**.

The example **Data Streaming MFC Client** was constructed with the Microsoft Foundation Classes. It uses the DLL for all datagram communications.

The other example **Data Streaming Client** is a console mode program that shows how to use both the DLL and direct datagram reception. You choose the actual method at run time.

8.4.1 Implementing the supplied DLL

The supplied DLL (**simgendatastreaming.dll**) takes care of the connection and delivery of datagrams to your software. First, you must make sure that the file **simgendatastreaming.dll** exists in the **Windows\System32** folder.

You must also reference the **EthernetShare.h** header file at the head of your program.

8.4.1.1 Loading the DLL into memory

```

#include "EthernetShare.h"
#include <Windows.h>
HMODULEedll = LoadLibrary( EthernetData::ethernet_dll );

```

You must check the value returned is not NULL and exit if it is.

If the DLL is not present then LoadLibrary will return NULL.

8.4.1.2 Version of loaded DLL

The function prototype for this is defined within **EthernetShare.h** as:

```

typedef int (*ethernet_version_func)( void )

```

Where:

The return value is the version number of **EthernetShare.h** that the DLL was compiled with, that is, `EthernetData::version`.

To obtain the address of this function within the DLL use the following code:

```
EthernetData::ethernet_version_func version_eth_p;
version_eth_p = ( EthernetData::ethernet_version_func )
                GetProcAddress( edll, EthernetData::version_dll_func );
```

To use the function use the following code:

```
int version = version_eth_p();
```

8.4.1.3 Start DLL listening for incoming datagrams

The function prototype is defined in **EthernetShare.h** as:

```
typedef void ( *ethernet_start_func ) ( bool multi_threaded );
```

Where:

multi_threaded is a Boolean flag that indicates to the DLL to use its own thread or not when receiving datagrams. If the user of this function supplies this parameter as **true** then this function will create a new thread and return immediately. If the user supplies this parameter as **false** then the function will block indefinitely until the program is terminated externally.

To obtain the address of this function within the DLL use the following code:

```
EthernetData::ethernet_start_func start_eth_p;
start_eth_p = ( EthernetData::ethernet_start_func )
              GetProcAddress( edll, EthernetData::start_dll_func );
```

To use the function use the following code:

```
start_eth_p( false );
```

8.4.1.4 Stop DLL listening for incoming datagrams

The function prototype is defined within **EthernetShare.h** as:

```
typedef void ( *ethernet_stop_func ) ( void );
```

To obtain the address of this function within the DLL use the following code:

```
EthernetData::ethernet_stop_func stop_eth_p;
stop_eth_p = ( EthernetData:: ethernet_stop_func )
             GetProcAddress( edll, EthernetData::stop_dll_func );
```

To use the function use the following code:

```
stop_eth_p();
```

8.4.1.5 Register interest callbacks for incoming datagrams

When the DLL receives a datagram it will call the function registered for the datagram type. Your program must declare each function to the DLL by calling the DLL function **ethernet_register_func**. It is not necessary to register a callback function for every datagram type as the DLL will discard unwanted datagrams. You can register only one callback function for each datagram type.

Before the DLL is commanded to start receiving datagrams, register all desired callback functions. This is mandatory if the DLL is not set to operate in its own thread, as the DLL does not then return control to the main program.

The function prototype for this is defined in **EthernetShare.h** as:

```
typedef void ( *ethernet_register_func )
```

```
( EthernetData::Message::Type type,
EthernetData::Callback_p      callback_p,
EthernetData::DeliveryType    delivery_type,
void*                          client_data_p );
```

Where:

type indicates which datagram type the registration is for that is **EthernetData::Message::vehicle**

callback_p indicates the user supplied function which must be of type

EthernetData::callback_p to call when the supplied **type** of datagram is received. The callback function is invoked with a pointer to the datagram of type **EthernetData::Message** and any supplied client data pointer. Only one callback will be active at a time. The action of calling a callback function will block the reception of any new datagrams until that callback function completes. The memory for the datagram is freed after the call to the callback therefore if the datagram needs to persist after the call then a copy must be made by the callback.

delivery_type indicates to the DLL when to call the callback. If **delivery_type** is set to **EthernetData::now** then the callback is invoked immediately. If **delivery_type** is set to **EthernetData::tov** then the callback is invoked at the time of validity stated in the datagram. The time of validity clock maintained by the DLL is synchronised to the times of receipt of the stream of **sync** datagrams and therefore there will be a slight lag relative to the clocks maintained by SimGEN and the signal generator(s).

client_data_p is a pointer to client data and is of type void. This pointer is passed with each callback function call.

To obtain the address of this function within the DLL use the following code:

```
EthernetData::ethernet_register_func register_eth_p_;
register_eth_p_ = ( EthernetData::ethernet_register_func )
GetProcAddress( edll, EthernetData::register_dll_func );
```

To use the function use the following code:

```
void messageCB( EthernetData::Message* msg_p,
                void* data_p )
{
    printf( "Received message type %d at %d (ms)\n",
            msg_p->type, msg_p->time_into_run );
}

registerEth_p_( EthernetData::Message::vehicle,
                messageCB,
                EthernetData::now,
                NULL );
```

8.4.1.6 Anatomy of a delivered data packet

Table 8-1 shows the anatomy in memory of the UDP data packet for the Signal type, see section 8.3.2.4. All data packets contain 408 bytes of data packed on an 8-byte boundary. They are delivered in little endian format. Packets that are smaller than 408 bytes are padded at the end with zeros to 408 bytes.

Table 8-1 UDP data packet for signal type

Offset	Field	Number of bytes
0x000	type	4
0x004	version	4
0x008	time_into_run	4
0x00C	time_of_validity	4
0x010	vehicle	4

Offset	Field	Number of bytes
0x014	antenna	4
0x018	channel	4
0x01C	satellite	4
0x020	type	4
0x024	prn	4
0x028	multi_path_index	4
0x02C	spare	4
0x030	frequency_specific[0].type	4
0x034	frequency_specific[0].spare	4
0x038	frequency_specific[0].pseudorange	8
0x040	frequency_specific[0].pseudorange_rate	8
0x048	frequency_specific[0].pseudorange_error	8
0x050	frequency_specific[0].delta_range	8
0x058	frequency_specific[0].iono_delay	8
0x060	frequency_specific[0].signal_level	8
0x068	frequency_specific[0].code_types	4
0x06C	frequency_specific[0].spare2	4
0x070	frequency_specific[1].type	4
0x074	frequency_specific[1].spare	4
0x078	frequency_specific[1].pseudorange	8
0x080	frequency_specific[1].pseudorange_rate	8
0x088	frequency_specific[1].pseudorange_error	8
0x090	frequency_specific[1].delta_range	8
0x098	frequency_specific[1].iono_delay	8
0x0A0	frequency_specific[1].signal_level	8
0x0A8	frequency_specific[1].code_types	4
0x0AC	frequency_specific[1].spare2	4
0x0B0	frequency_specific[2].type	4
0x0B4	frequency_specific[2].spare	4
0x0B8	frequency_specific[2].pseudorange	8
0x0C0	frequency_specific[2].pseudorange_rate	8
0x0C8	frequency_specific[2].pseudorange_error	8
0x0D0	frequency_specific[2].delta_range	8
0x0D8	frequency_specific[2].iono_delay	8
0x0E0	frequency_specific[2].signal_level	8
0x0E8	frequency_specific[2].code_types	4
0x0EC	frequency_specific[2].spare2	4
0x0F0	posn[0]	8
0x0F8	posn[1]	8

Offset	Field	Number of bytes
0x100	posn[2]	8
0x108	velocity[0]	8
0x110	velocity[1]	8
0x118	velocity[2]	8
0x120	acceleration[0]	8
0x128	acceleration[1]	8
0x130	acceleration[2]	8
0x138	elevation	8
0x140	azimuth	8
0x148	true_range	8
0x150	true_range_rate	8
0x158	tropo_delay	8
0x160	antenna_elevation	8
0x168	antenna_azimuth	8
0x170	errors.divergence_terms	8
0x178	errors.relativistic_effects	8
0x180	errors.ramp_error	8
0x188	errors.clock_noise	8
0x190	errors.spare	8

8.4.2 Putting it together

Below is an example from the codes given in sections 8.4.1.1 to 8.4.1.6.

```

/*****
*
We register one callback for this example - but you can register a
separate callback for each different type if needed.
*****/

/
void messageCB( EthernetData::Message* msg_p,
                void* data_p )
{
    printf( "Received message type %d at %d (ms)\n",
msg_p->type, msg_p->time_into_run );
}

/*****
*
This function implements an example of using the DLL to obtain the
data stream from the host software.
*****/

/
void doDll( void )
{
    HMODULE          edll;
    EthernetData::ethernet_start_func start_eth_p;
    EthernetData::ethernet_stop_func stop_eth_p;
    EthernetData::ethernet_register_func register_eth_p;

    // Load the DLL into memory
    if (( edll = LoadLibrary( EthernetData::ethernet_dll )) != NULL )
    {
        // Obtain the function pointers from the DLL
        start_eth_p = ( EthernetData::ethernet_start_func )
GetProcAddress( edll, EthernetData::start_dll_func );
        stop_eth_p = ( EthernetData::ethernet_stop_func )
GetProcAddress( edll, EthernetData::stop_dll_func );
        register_eth_p = ( EthernetData::ethernet_register_func )
GetProcAddress( edll, EthernetData::register_dll_func );

        // Register the callbacks we could register some of our own data
        // to be passed back to the callback if we wanted to but in this
        // example we set the client data to NULL.
        register_eth_p( EthernetData::Message::vehicle,
messageCB,
EthernetData::now,
NULL );

        register_eth_p( EthernetData::Message::antenna,
messageCB,
EthernetData::now,
NULL );

        register_eth_p( EthernetData::Message::signal_data,
messageCB,
EthernetData::now,
NULL );

        register_eth_p( EthernetData::Message::sync,
messageCB,
EthernetData::now,
NULL );

        // For this example we dont want to create a new thread so
        // just start the DLL waiting for messages. When a message is
        // received the callback will be called.
    }
}

```

```

start_eth_p( false );
}
else
{
printf( "ERROR: Could not load %s, please check that it is within\n"
        " the same directory as this executable\n\n",
        EthernetData::ethernet_dll );
}
}
}

```

8.4.3 Custom client

You can implement a custom client from using the supplied **EthernetShare.h** file and the example C++ code. To implement a custom client you must perform, as a minimum, the following steps:

- a) Include the supplied **EthernetShare.h** file.
- b) Create a datagram socket that will be used to receive the SimGEN transmitted datagrams. For example:

```

SOCKET socket_fd;
socket_fd = socket( AF_INET, SOCK_DGRAM, 0 );

```

- c) Bind to the port number that SimGEN will be transmitting upon. For example:

```

SOCKADDR_IN sock_addr;
memset( &sock_addr, 0, sizeof( sock_addr ) );

sock_addr.sin_family      = AF_INET;
sock_addr.sin_port        = htons( EthernetData::port );
sock_addr.sin_addr.s_addr = htonl( INADDR_ANY );

// Bind to the above port
if( bind(socket_fd, (struct sockaddr*) &sock_addr, sizeof(
sock_addr )) < 0 )
{
    printf( "Bind failed for port %d\n", EthernetData::port
);
}

```

- d) Set the receiving buffer space to a size larger than **EthernetData::receive_size**.

```

if ( setsockopt( socket_fd,
                SOL_SOCKET,
                SO_RCVBUF,
                ( const char* ) &rx_buff_size,
                sizeof( rx_buff_size ) ) == SOCKET_ERROR )
{
    printf( "failed for to set sockets receive buffers\n" );
}

```

- e) Read datagrams from the socket as they arrive. For example:

```

int bytes_read = 0;
EthernetData::Message msg;

if ( ( bytes_read = recvfrom( socket_fd,
                             (char *) &msg,
                             sizeof( msg ),
                             0,
                             (struct sockaddr *) NULL,
                             (int *) NULL) ) == SOCKET_ERROR )

```



```

    {
        printf( "Error receiving data from UDP port" );
    }
    else
    {
        // Make sure we have read the correct amount of data
        // and there have been no under runs.
        if ( bytes_read != sizeof( msg ) )
        {
            printf( "Error size of message inconsistent" );
        }
    }
}

```

8.4.3.1 Example custom code

```

/*****
*
* Selects which windows socket version to use. If the function
* returns false then it has failed.
*****/
/
bool select_WinSock( int major_version, int minor_version )
{
    WORD version = MAKEWORD( major_version, minor_version );
    bool result = true;
    WSADATA wsa_data;

    if ( WSAStartup( version, &wsa_data ) != 0 )
    {
        result = false;
    }
    else
    {
        if( LOBYTE( wsa_data.wVersion ) != major_version ||
            HIBYTE( wsa_data.wVersion ) != minor_version )
        {
            result = false;
            WSACleanup();
        }
    }

    // Test for any errors
    if ( result == false )
    {
        printf( "ERROR: Failed to start-up the windows socket"
            " DLL version %d.%d\n",
            major_version, minor_version );
    }

    return( result );
}

/*****
*
* Creates a UDP socket and binds to the known port, returns false if the
* socket could not be created.
*****/
/
bool create_socket( void )
{
    static const int rx_buff_size = EthernetData::receive_size;
    bool result = true;

    // Create the socket

```

```

        if (( socket_fd = socket( AF_INET, SOCK_DGRAM, 0 )) !=
INVALID_SOCKET )
        {
            memset( &sock_addr, 0, sizeof( sock_addr ));

            sock_addr.sin_family      = AF_INET;
            sock_addr.sin_port        = htons( EthernetData::port );
            sock_addr.sin_addr.s_addr = htonl( INADDR_ANY );

            // Bind to the above port
            if ( bind( socket_fd,
                      (struct sockaddr *) &sock_addr,
                      sizeof( sock_addr )) < 0 )
            {
                result = false;
            }

            // Set the receiver buffer size to something that
            // is large enough to buffer up the incoming messages
            if ( setsockopt( socket_fd,
                            SOL_SOCKET,
                            SO_RCVBUF,
                            ( const char* ) &rx_buff_size,
                            sizeof( rx_buff_size )) == SOCKET_ERROR )
            {
                result = false;
            }
        }
    else
    {
        result = false;
    }

    // Test for any errors
    if ( result == false )
    {
        printf( "ERROR: Failed to create and setup the socket\n" );
    }

    return( result );
}

/*****
*
* Process the incoming messages
*****/
/
bool process_messages( EthernetData::Message* msg_p, bool* new_msg_p )
{
    bool ok          = true;
    int  highest_fd  = 0;
    int  result       = 0;
    int  bytes_read   = 0;
    fd_set read_set;

    // Clear the file descriptor set
    FD_ZERO ( &read_set );

    // Set the socket we are listening upon
    FD_SET( socket_fd, &read_set );

    // Set the highest fd to one more than then fd
    // we will be reading from as this will be used in the select.

```

```

highest_fd = ( socket_fd + 1 );

// We sleep for timeout waiting for any new messages
if (( result = select( highest_fd,
                      &read_set,
                      NULL,
                      NULL,
                      &timer)) != SOCKET_ERROR )
{
    // Check that we have something to read.
    if ( FD_ISSET( socket_fd, &read_set ))
    {
        // Read from the UDP port and place the result
        // data into the message supplied.
        if (( bytes_read = recvfrom( socket_fd,
                                    (char *) msg_p,
                                    sizeof( EthernetData::Message ),
                                    0,
                                    (struct sockaddr *) NULL,
                                    (int *) NULL)) == SOCKET_ERROR )
        {
            printf( "ERROR: Receiving data from UDP port" );
            ok = false;
        }
        else
        {
            // Make sure we have read the correct amount of
data
            // and there have been no under runs.
            if ( bytes_read != sizeof( EthernetData::Message
))
            {
                printf("ERROR: Size of message
inconsistent");
                ok = false;
            }
            else
            {
                // Received a good new message
                (*new_msg_p) = true;
            }
        }
    }
    else
    {
        // Didn't get a new message
        (*new_msg_p) = false;
    }
}

return( ok );
}

/*****
*
This function implements an example custom way of obtaining the
streamed
data from the host software.
*****/
/
void do Custom( void )
{
    EthernetData::Message msg;

```

```

bool  new_msg;

// Before we start output the header for the counts
// that we are going to display upon the console
outHeader();

// Try to start up the windows sockets dll
if ( select_WinSock( winsock_major, winsock_minor ))
{
    // Create the socket and bind to the port that
    // that data will be sent to.
    if ( create_socket() )
    {
        // Setup the timer to wait for at least one millisecond
        timer.tv_sec= 0;
        timer.tv_usec = MS_WAIT( 1 );

        // Sit looping and processing all the incoming messages
        while ( process_messages( &msg, &new_msg ))
        {
            if ( new_msg )
            {
                printf( "Received message type %d at %d
(ms)\n",
                        msg_p->type, msg_p->time_into_run );
            }
        }
    }
}

```

8.4.3.2 Example of implementing a client in Java

You must remember that Java uses the big endian format. This example uses a `DataInputStream` that can read little endian and convert it to big endian for use.

```

import java.io.*;
import java.net.*;

public class SimGENUDPMsg
{
    public SimGENUDPMsg( DatagramPacket buff )
    {
        msg = buff;
        bin = new ByteArrayInputStream( msg.getData() );
        din = new LEDataInputStream( bin );
    }

    private String getType( int type )
    {
        String stype = "Unknown";

        switch( type )
        {
            case 0 : stype = "Vehicle"; break;
            case 1 : stype = "Antenna"; break;
            case 2 : stype = "Signal"; break;
            case 3 : stype = "Vehicle COG Additional"; break;
            case 4 : stype = "Truth Aiding"; break;
            case 5 : stype = "Truth Aiding EGI"; break;
            case 6 : stype = "INS Error Model"; break;
            case 7 : stype = "Space Vehicle"; break;
            case 8 : stype = "Sync"; break;
        }

        return( stype );
    }

    private void outSync()
    {
        try
        {
            byte[] startTime = new byte[ 25 ];
            din.read( startTime, 0, 25 );
            String st = new String( startTime, 0, 25 );

            din.read( startTime, 0, 7 );

            System.out.println( "Start Time: " + st );
            System.out.println( "Duration: " + din.readInt() );
            System.out.println( "UTC Offset: " + din.readInt() );
        }
        catch ( Exception e )
        {
            System.err.println( e );
        }
    }

    private void outAntenna()
    {
        try
        {
            System.out.println( "Vehicle ID: " + din.readInt() );
            System.out.println( "Antenna ID: " + din.readInt() );

            System.out.println( "Position X: " + din.readDouble() );
            System.out.println( "Position Y: " + din.readDouble() );
            System.out.println( "Position Z: " + din.readDouble() );
        }
    }
}

```

```

        catch ( Exception e )
        {
            System.err.println( e );
        }
    }

    private void outVehicle()
    {
        try
        {
            System.out.println( "Vehicle ID: " + din.readInt() );
            din.readInt(); // Spare

            System.out.println( "Position X: " + din.readDouble() );
            System.out.println( "Position Y: " + din.readDouble() );
            System.out.println( "Position Z: " + din.readDouble() );
        }
        catch ( Exception e )
        {
            System.err.println( e );
        }
    }

    public void out()
    {
        try
        {
            int type = din.readInt();

            System.out.println( "===== " );
            System.out.println( "Type: " + getType( type ) );
            System.out.println( "Version : " + din.readInt() );
            System.out.println( "Time in to run (ms) : " + din.readInt() );
            System.out.println( "Time of validity (ms) : " + din.readInt() );

            switch( type )
            {
                {
                    case 0 : outVehicle(); break;
                    case 1 : outAntenna(); break;
                    case 8 : outSync(); break;
                }
            }
            catch ( Exception e )
            {
                System.err.println( e );
            }
        }
    }

    private DatagramPacket msg;
    private ByteArrayInputStream bin;
    private LEDataInputStream din;
}

public class UDPReceive
{
    public static void main( String args[] )
    {
        try
        {
            {
                if ( args.length != 1 )
                {
                    throw new IllegalArgumentException( "Wrong number of arguments" );
                }
            }
        }
    }
}

```

```

int port= Integer.parseInt( args[ 0 ] );
DatagramSocketdsocket = new DatagramSocket( port );
byte[]buffer= new byte[ 2048 ];

for ( ;; )
{
    DatagramPacket packet = new DatagramPacket( buffer, buffer.length );

    dsocket.receive( packet );

    SimGENUDPMsg msg = new SimGENUDPMsg( packet );
    msg.out();
}
catch ( Exception e )
{
    System.err.println( e );
    System.err.println( "Usage: java UDPReceive <port>" );
}
}
}

```

8.4.4 Spirent example client

Spirent supplies SimGEN with a Microsoft Foundation Classes based Data Streaming client. This client is to aid in verifying network connectivity and SimGEN data generation.

The client needs *simgendatastreaming.dll* to run. This *.dll* file must be in the same folder as the Data Streaming executable file, *EthernetClient.exe*.

8.4.4.1 Running the client

See reference a) for details of the SimGEN **Data Streaming Definition** file.

Select **Scenario - Options - Data streaming definition file - General - Time sync message**. Then, for each Vehicle and Antenna that you want to stream data, select:

Data streaming definition file - Vehicle_n - Vehicle_n motion data

Data streaming definition file - Antenna_n - Antenna_n motion data

Data streaming definition file - Antenna_n - GPS signal

On the SimGEN PC, select a Windows PC on your network.

Note: *The PC you select should be on the same sub-network, and at least as powerful, as your SimGEN controller.*

Then:

- a) Get the IP address of the selected Windows PC (open a **Command Prompt** window and type "*ipconfig*").
- b) Enter this IP address in **Data streaming definition file - Send data messages to - I.P. address**.
- c) Working on the Windows PC you selected, install the **Data Streaming** client and *simgendatastreaming.dll* into a suitable folder.
- d) Double-click the **Data Streaming** client. Figure 8-8 shows the **Data Streaming** dialog.

Figure 8-8: SimGEN Data Streaming Client

SimGEN Data Streaming Client

Vehicle | Antenna | Signal Data | Sync Time 0

		X	Y	Z
Vehicle	0	Position	0	0
Latitude	0	Velocity	0	0
Longitude	0	Acceleration	0	0
Height	0	Jerk	0	0
T.O.V	0	Attitude	0	0
T.I.R	0	Angular Velocity	0	0
Baro height	0	Angular Acceleration	0	0
Grnd speed	0	Angular Jerk	0	0

Scenario Data

Start Time: Duration (s): UTC Offset (s):

Displayed Packet Counts

Vehicle 0 Antenna 0 Signal Data 0 Total 0

Start Stop Cancel

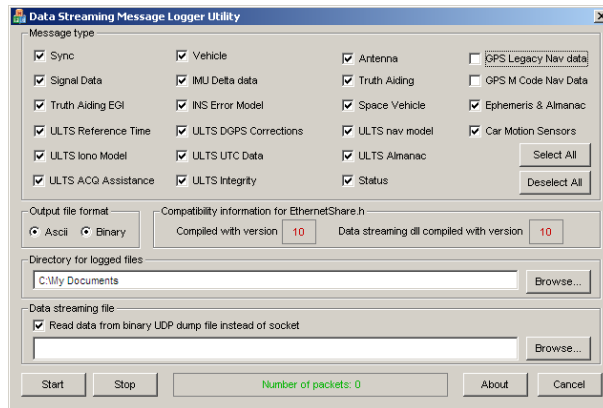
- e) Click **Start** to start the **Data Streaming** client listening for datagrams.
- f) Start your SimGEN simulation.
- g) The **Data Streaming** client refreshes its contents every second.
- h) Use the tabs at the top of the **Data Streaming** dialog to select the data view.
- i) The updated data is dependant upon data selected for transmission in the SimGEN **Data Streaming** dialog.
- j) Click **Stop** to stop the **Data Streaming** client listening for datagrams.
- k) Click **Start** to start listening again.
- l) The **Data Streaming** client can tolerate interruptions from SimGEN, letting you stop and start the scenario running on the SimGEN PC at anytime.

8.5 Data streaming message logger utility

Note: The logger utility needs the file *simgendatastreaming.dll* to be present in the Utilities folder or in the folder *Windows\System32*.

This Utility, see Figure 8-9, checks UDP data sent over Ethernet by SimGEN's Data Streaming option.

Figure 8-9: Data Streaming Message Logger Utility dialog



Select **Output file format - binary** or **- ASCII** to choose the format of the output file. Click **Browse** to navigate to a folder where you can save the output file under a name you enter. The filename corresponds to the message type being logged. For example, if you select **Message type - Sync**, Sync messages will be logged as **sync.csv** (ASCII format) or **sync.bin** (binary format). To start the logger utility, navigate to your PosApp folder (default location is *C:\Program Files\Spirent Communications\PosApp*) and double-click *Data Streaming Message Logger Utility.exe*. To use the logger utility:

- Select each **Message Type** you want to log.
- Select the **Output file format** and choose the **Directory for logged files**, which is where the log files are written.
- Click **Start**.
- The number of packets received is shown in the central area. between the buttons.
- Click **Stop** to stop the logger.

For data streaming files, you can select **Read data from binary UDP dump file instead of socket** and click **Browse** to navigate to the dump file.

If the logger utility does not start receiving packets, check SimGEN is running and:

- You have selected **Scenario-Options-Data streaming definition file**:
- You have selected **Vehicle and Antenna motion data**, as required, in the appropriate **Data Streaming Definition-Vehicle** and **Antenna** dialog boxes
- You have selected the appropriate **Signal data** in **Vehicle_n Antenna_n Settings**
- The IP address in the **Data Streaming Definition-General** dialog is the same as the IP address of the machine running the **Data Streaming Message Logger Utility**.

Finally, check you have selected the correct messages in the **Data Streaming Message Logger Utility** dialog.

This page is intentionally blank

Chapter 9: Acronyms and abbreviations

1CPS	One Character Per Second
1PPS	One Pulse Per Second
<CR><LF>	Carriage Return and Line Feed characters
AGP	Advanced Graphics Port – a type of graphics card socket found in a PC
ASCII	Standard character set used for serial communications
CD-ROM	Term for a Compact Disk when used as a PC storage medium
CofG	Centre of Gravity
DLL	Dynamically Linked Library – Microsoft promoted mechanism for sharing common code between applications.
ECEF	Earth Centred Earth Fixed
EGI	Embedded GPS Inertial
EOI	End or Identify
GPIO	General Purpose Interface Board – term for the National Instruments IEEE-488 bus interface cards.
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronic Engineers – standards body for many widely used interfaces
I/O	Input Output
IP	Internet Protocol – generically the set of standards that define the Internet
IRQ	Interrupt Request Level – a resource option for PC interface devices, normally set automatically by devices that meet ‘Plug and Play’ standards
LAAS	Local Area Augmentation System
PC	Personal Computer
PCI	A standard type of interface board socket found in PCs
PCMCIA	A standard interface commonly supplied for portable computers. Also known as PC Card it is Hot Pluggable enabling devices to be added/removed as needed.
PPM	Parts Per Million
PRN	Pseudo Random Number
STR47n0	Refers to a STR4750, STR4760, STR4780 or a STR4790 signal generator
TCP/IP	Transmission Control Protocol – the main interface standard used for reliable connection mode communications using the Internet
Toa	Time of Application
UDP/IP	User Datagram Protocol – interface standard for connection less communications using the Internet
USB	Universal Serial Bus – Hot pluggable interface for PC peripheral devices
VMS	Acronym for the Operating System used with DEC Alpha Workstations.
XML	eXtensible Markup Language – a standard syntax and semantics for passing information between computers as a text based vendor independent file.

This page is intentionally blank

Chapter 10: Configuring remote operation

10.1 SimGEN controller

Note: *The test programs described in section B.3.3 require the SimREMOTE timer card to be installed in the Remote PC. You cannot use these test programs to confirm correct operation of the remote control interface if you do not use the SimREMOTE hardware package.*

If the real-time, remote IEEE-488 option was ordered with a new Spirent signal generator, the SimGEN controller has been pre-configured by Spirent with an additional IEEE-488 Interface card and all necessary software. Skip to section B.5 of this document.

Spirent pre-configures its Remote PC's with interface cards and software. If you are using a Remote PC not supplied by Spirent, you must install the IEEE-488 (GPIB) card and the PCI-215 timer card Spirent supplies in your Remote PC. Refer to the manual supplied with your SimGEN controller for details on installing additional cards. Section B.2 describes the installation procedure.

When you get SimREMOTE as an upgrade, separately from your main signal generator, you may have to install additional hardware to the SimGEN controller. These will vary according to your specific system. This appendix contains installation instructions, use the section(s) appropriate to your system. If you are unsure which system you have, contact Spirent for assistance.

If your system is to operate remotely without the SimREMOTE hardware package, see section B.5 for details on enabling remote control in SimGEN.

10.2 Optional IEEE-488 and Timer cards

Notes:

1) *Spirent pre-installs the IEEE-488 and Timer cards in Remote PC it supplies and you can skip to section B.5.*

2) *To prevent possible problems with drivers and software, Spirent recommends you install the Timer card before the IEEE-488 card.*

Spirent supplies a National Instruments PCI-488/AXP (IEEE-488) card and an Amplicon Liveline PCI-215 Timer card as part of the SimREMOTE upgrade. These cards are intended for use in a PC acting as the Remote PC, or as a test tool capable of evaluating and/or developing the interface.

10.3 Optional Timer card

The Timer card provides a method of precise time synchronisation between the Remote PC and the signal generator.

10.4 Install the card



SAFETY WARNING

Before you install the Timer Card, completely isolate mains power from the PC by removing the plug from the mains socket



Use anti-static handling precautions when you handle the timer card

Notes:

1) If the PC has a Graphics Display (Video) card fitted in the AGP or PCI Express port, do not use the adjacent PCI slot as this PCI slot has an address that is often common to the AGP port.

2) AGP and PCI Express ports are colour-coded to distinguish them from PCI slots; and their positions are offset.

Before installing the Timer card, the PC has sufficient slots for the Timer card and the optional IEEE-488 card. Take into account other boards or adapters that may be installed when assessing physical space and the power requirements. The Timer card is a half-length card.

PCs supplied by Spirent have their graphics controller built into the motherboard so the AGP/PCI Express port is unused. In this case, all PCI slots will be available for additional cards.

Refer to the PC manufacturer's hardware manual for instructions on how to install devices into a PCI slot. Refer to the hardware manual for any restrictions on installing cards to PCI slots.

10.5 Install the drivers

Refer to the documentation supplied with the Timer card for details on using the installation software.

10.6 Testing

Notes:

1) You must install the IEEE-488 card before you can test the Timer card. If the IEEE-488 card is not installed, return to this test after installing the IEEE-488 card, see section B.4.

2) You cannot test the Timer card without using the IEEE-488based remote motion demonstration program, which requires an IEEE-488 card to be fitted to the PC. Currently, Spirent does not offer an alternative test procedure.

If you have an IEEE-488 card installed, test the Timer card using the following procedure:

- a) Connect the Counter-Timer Adapter to the 78-way D connector on the back of the Timer card
- b) Connect a BNC to BNC co-axial cable from the STRn7xx or GSS77xx TIMER OUTPUT 3 socket or the GSS6560 1PPS output socket as appropriate, to the IN lead of the Counter-Timer Adapter,
- c) Switch on the hardware platform and turn on the SimGEN PC,
- d) Start the SimGEN remote motion demonstration program *remote_motion_ieee.exe*, see C.2.
- e) Run the Timing test command, described in section C.3.5.

10.7 Optional IEEE-488 card

Note: *This IEEE-488 bus is separate from the bus used to control the signal generator.*

You must connect the Remote PC to the SimGEN PC using the IEEE-488 bus.

10.7.1 Install the drivers

Before installing the IEEE-488 card, Spirent recommends installing the National Instruments drivers and support software from the supplied CD-ROM. Refer to the documentation supplied with the IEEE-488 card for details on using the installation software.

10.7.2 Install the card

SAFETY WARNING



**BEFORE YOU INSTALL THE IEEE-488 CARD
COMPLETELY ISOLATE MAINS POWER FROM THE PC BY
REMOVING THE PLUG FROM THE MAINS SOCKET**



Use anti-static handling precautions when you handle the IEEE-488 card

Notes:

- 1) A system using only GSS6560 signal generators uses the USB bus to control the signal generators. If this system is fitted with an IEEE-488 card, use this card for connection to the Remote PC.
In all other systems, you must use a second IEEE-488 card for connection to the Remote PC.
- 2) The first IEEE-488 card is given the identifier GPIB0. Spirent recommends marking the card's endplate with a '0' for future reference and also arranging the cards such that the GPIB0 card is situated closer to the AGP port, which will usually be toward the centre of the PC.
- 3) The PCI ports are numbered with the AGP port as address 0. By default, the National Instruments software allocates GPIB identities in sequence with the PCI port addresses, so the IEEE-488 card in the lower address becomes GPIB0 and the other card GPIB1.
- 4) A second National Instruments IEEE-488 card needs no additional software.

Refer to the PC manufacturer's hardware manual for instructions on how to install devices into a PCI slot. Refer to the hardware manual for any restrictions on installing cards to PCI slots.

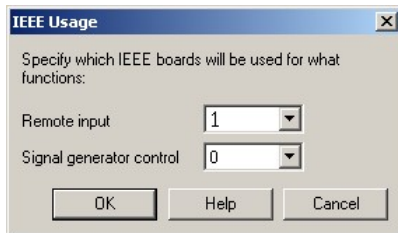
Install the IEEE-488 card following the instructions provided by National Instruments.

Re-start the PC and complete the final steps in the installation process. You should accept the default options at each stage that will allow Windows to complete the installation automatically.

Follow the instructions provided by National Instruments to complete the verification stage.

After fitting the National Instruments card, give it the identifier GPIB1. After installing the card and re-booting the PC, you must complete the National Instruments card setup script.

Finally, open SimGEN and set IEEE card 0 for the signal generator controller and set IEEE card 1 for the remote input. Click on **Options - IEEE usage**, see .

Figure 10-1: Set IEEE usage

Select **1** for **Remote input**.

Select **0** for **Signal generator control**.

Click **OK**.

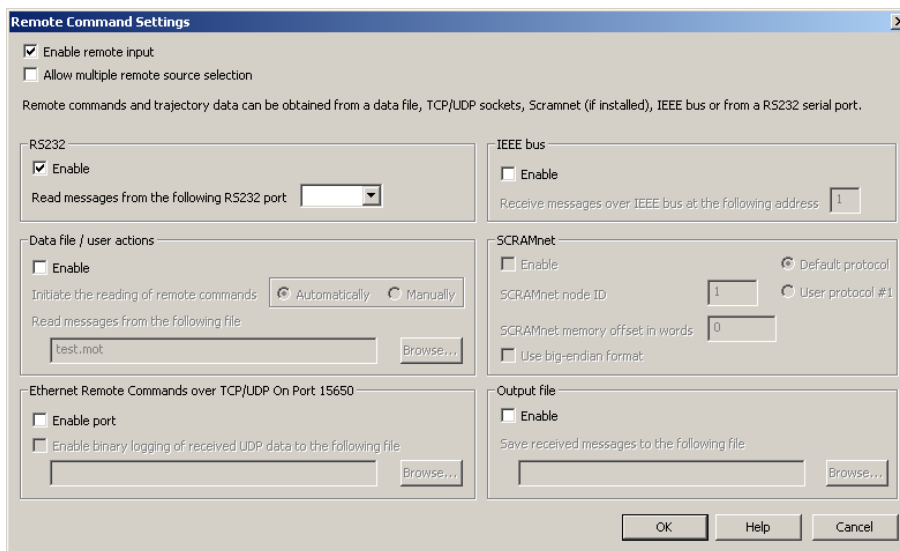
10.8 Enable SimGEN for remote control

Notes

1) Only use manual mode when reading remote commands from a file. In 'automatic' mode the external program initiates control. In either mode, a remote run command, 'RU' (see section 5.2.8) is required to start the scenario. If a run command is not present, you must start the run by clicking **Run** on the SimGEN toolbar.

2) When SimGEN is under remote control, Spirent recommends you suppress message pop-ups as they require acknowledgement before the application resumes. You can suppress message pop-ups using **Options - Message Reporting - Disable message popups**.

To enable SimGEN to use a remote command source, select **Options-Remote Command Settings**, see Appendix Figure B-2.

Figure 10-2: Remote Command Settings dialog

You must select **Enable remote input** to activate remote control.

You can select one or more sources. For each source, you must complete the associated items to enable that source.

Selecting **Data file / user actions – Initiate the reading of remote – Automatically**, begins reading the remote commands immediately you close the Remote Command Settings dialog.

Selecting **Data file / user actions – Initiate the reading of remote – Manually**, requires you to click the **Remote** button on the SimGEN toolbar to begin reading remote commands.

Selecting **Output file - Enable** will record all incoming remote commands to a file. You can use this option to replay remotely controlled scenarios that were originally generated in real-time, for example in closed loop operation.

Section 4.1 describes the options for the Remote Control sources.

This page is intentionally blank

Chapter 11: Confidence tests

Note: The test program is intended only for systems configured with a Remote PC.

These tests use the IEEE-488 option. You must select **Remote Command Setting -IEEE Bus – Enable**.

You must ensure the IEEE-488 bus address matches the address used in the remote control program; the example supplied by Spirent assumes a bus address of 1.

Spirent supplies demonstration software and its source code with SimGEN. This software lets you to test the basic installation integrity and experiment with the control messages forming part of the interface protocol. The source code will assist you to understand the implementation of the control messages.

The remote control demonstration software is installed on the SimGEN PC as part of the normal SimGEN installation, in the folder

C:\Program Files\Spirent Communications\PosApp\Remote Control Demo Software.

You can also install this software to a separate PC used as the remote controller, see section C.1.

Unless pre-installed by Spirent, you must install the test software following the instructions in this document before running the test program.

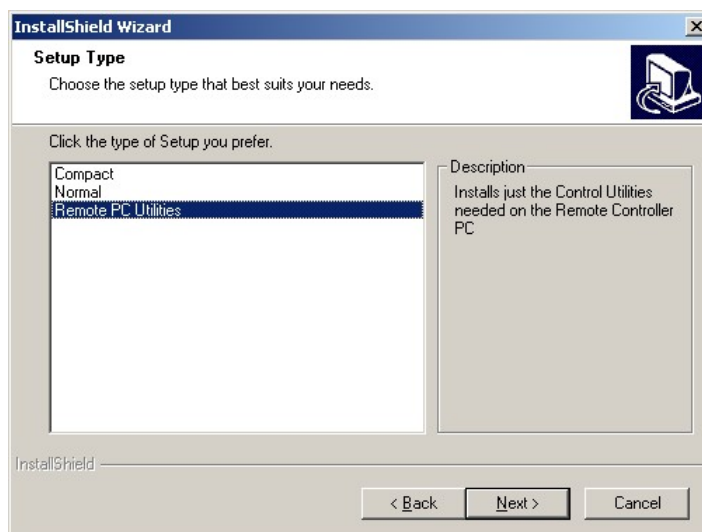
11.1 Install the IEEE test program on the remote PC

Note: This does not install the full SimGEN package, just the items needed on the Remote PC.

Install the Demonstration Software to the Remote PC from the standard SimGEN Installation CD, as follows:

- a) Insert the CD into the CD Drive of the Remote PC; the installation script starts automatically.
- b) Select the default options by clicking Next until you see the InstallShield Wizard window, see Appendix Figure C-1.

Figure 11-1: Remote PC Utilities installation dialog



- c) Select **Remote PC Utilities**.
- d) Click **Next** to install the Remote Control software on the remote PC.

The default installation folder for the Remote PC Utilities is
C:\Program Files\Spirent Communications\PosApp\Remote Control Demo Software.

Three Demonstration packs are installed:

<i>Remote Logging TCPIP</i>	-Log scenario progress remotely using Ethernet
<i>Remote Motion TCPIP</i>	-Supply scenario motion data using Ethernet
<i>Remote Motion IEEE</i>	-Supply scenario motion data using IEEE-488

The first two packs are configured to operate with the 'localhost' Ethernet/TCPIP network address. They must be executed on the main SimGEN PC alongside SimGEN. They may be modified to operate from a remote PC.

11.1.1 Using the IEEE bus

Note: *Many IEEE instruments use <CR> or <LF>, or both, to denote end of message, this is not acceptable for use with SimGEN.*

Both IEEE-488 cards on the bus should be set to generate the EOI (End Or Identify) signal with the last byte of each message. The software must terminate message reads on receipt of the EOI signal, and terminate all messages sent with EOI.

SimGEN and the Remote Motion IEEE example program apply these settings by software command.

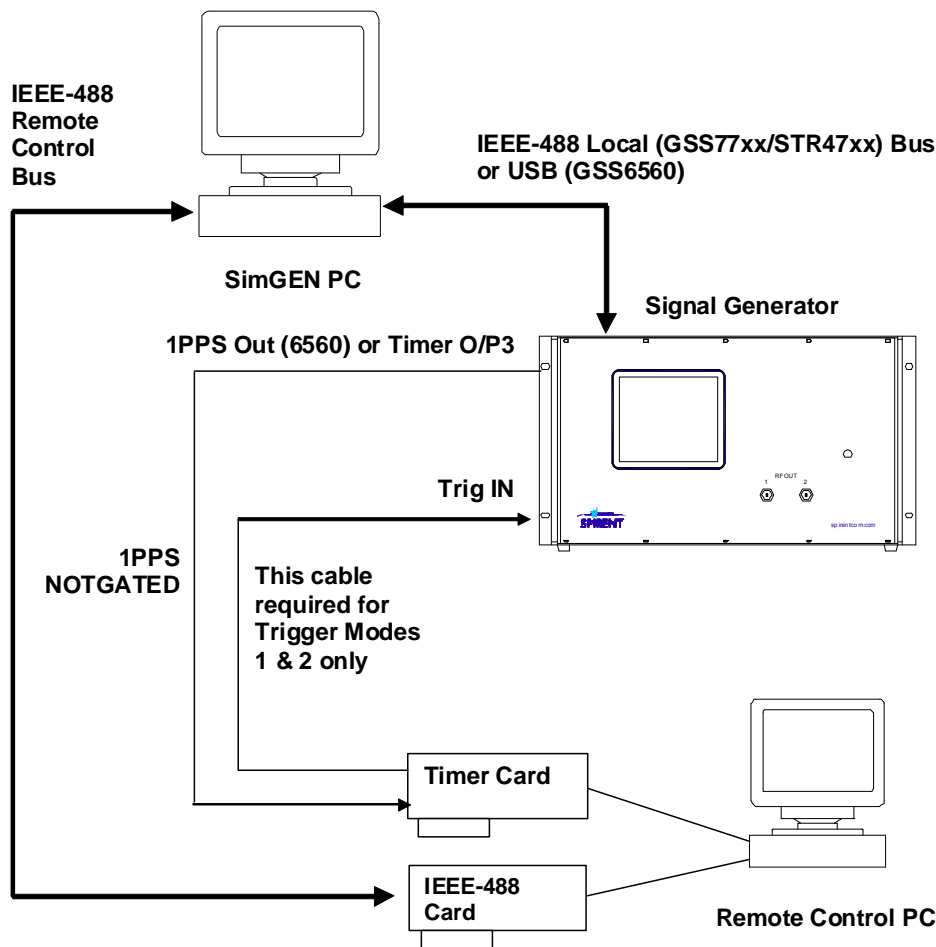
11.1.2 Test using the remote motion IEEE software

Note: *The 1PPS NOTGATED cable must be connected correctly for the test/demonstration programs to function correctly. If the programs fail to start then check the 1PPS NOTGATED cable connections and the signal generator timer output configuration.*

The remote motion IEEE pack includes the source code and Microsoft Visual C++ project files used to build the *port_ieee_test.exe* test program. A release build of the program is installed with the pack and this should be used for the following test sequence.

Before running the test program, configure the system as shown in Appendix Figure C-2.

Figure 11-2: IEEE_488 test configuration



Start SimGEN on the main PC, select or create a scenario and ensure this is compatible with the signal generator(s) available.

To run the `port_ieee_test.exe` program on the remote PC, open a Command Prompt window and navigate to the Remote motion IEEE\Release folder using the `CD` command (that is, using the short folder names type: "`CD progra~1\spiren~1\PosApp\remote~1\remote motion ieee\release`").

Type the program name `port_ieee_test.exe` and press **Return**.

Figure 11-3: IEEE test program menu

```

C:\ Command Prompt - port_ieee_test.exe
2 Dir(s) 2,850,459,648 bytes free
C:\PROGRA~1\SPIREN~1\SimGEN\REMOTE~1\Remote Motion IEEE\Release>port_ieee_test.exe
** SimGEN Remote IEEE488 example program. Iss 1.04 2/6/04 **
1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>

```

The port_ieee_test.exe program displays a list of the available actions.

Type "4" and press **Return** to start the scenario running, see Appendix Figure C-4.

Figure 11-4: Running a simulation with the IEEE test program



```

C:\> Command Prompt - port_ieee_test.exe
4
Data read: <msg>
          <status> 2 </status>
          <
Sim response="<msg>
          <status> 2 </status>
          <" target is SimGEN

Card status = ok Simulation status = 2
Calling init timer card fn
Board 0 Type 215 IRQ 17 BASE -8928 RESULT 0
Timer initialised
Trig command sent
ARM command sent
System Armed
Run command sent

1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>5

Card status = ok Simulation status = 5
END command sent

1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>6

Card status = ok Simulation status = 7
Rewind command sent

1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>

```

This will instruct SimGEN to Arm and Run the scenario.

To stop the scenario type **5** and press return.

To rewind or reset the scenario, type "6" and press **Return**.

If the sequence completes as shown in Appendix Figure C-4 then the Remote PC and software are installed OK.

11.2 Options using the IEEE remote test program

11.2.1 Change current scenario

To change the current scenario from using the Remote control demo, software:

- a) Type "2" and press **Return**.
- b) Type the full path including disk letter and the file name for the scenario, and press **Return**.
- c) Type "1" and press **Return** to change the scenario.
- d) SimGEN will close the old scenario and open the new scenario.

Figure 11-5: Changing the scenario

```

Command Prompt - port_ieee_test.exe
>2
Card status = ok Simulation status = 2
Enter scenario name>c:\progra~1\spiren~1\simgen\scenar~1\geoff_remote\geoff_remote.scn
1. Select scenario c:\progra~1\spiren~1\simgen\scenar~1\geoff_remote\geoff_remote.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>1
Card status = ok Simulation status = 2
1. Select scenario c:\progra~1\spiren~1\simgen\scenar~1\geoff_remote\geoff_remote.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>

```

11.2.2 Check SimGEN simulation status

SimGEN responses carry a code number that indicates the state of the current simulation.

You can get the state value using a specific command:

Type "7" and press **Return**, SimGEN gives one of the responses listed in Appendix Table C-1.

Table 11-1: SimGEN status responses

State Code	SimGEN state
0	No Scenario loaded
1	Not completed loading a scenario
2	Idle, ready to run a scenario
3	Arming the scenario
4	Completed arming; or waiting for a command or trigger signal to start the scenario
5	Scenario running
6	Current scenario is paused.
7	Active scenario has stopped and has not been reset. Waiting for further commands.

11.2.3 Change to another trigger mode

SimGEN enables simulations to be started in one of three time sequences:

- a) Trigger mode Disabled No trigger required; start at the next 1PPS event time.
- b) Trigger mode Immediate Trigger is required; start immediately the trigger pulse is detected.
- c) Trigger mode Delayed Trigger is required; start at the next 1PPS event after the trigger pulse.

To change the trigger mode setting:

- a) Type "3" and press **Return**, SimGEN responds with the message:
- b) *"Enter trig mode <0=sw, 1=ext immediate, 2=ext delayed> >"*
- c) Type "0", "1" or "2" as required and press **Return**.

The command list will be displayed and shows the new trigger mode code in the descriptions of commands 4 and 8.

Figure 11-6: Changing the trigger mode

```

Command Prompt - port_ieee_test.exe
1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>3

Card status = ok Simulation status = 2
Enter trig mode <0=sw, 1=ext immediate, 2= ext delayed>>2

1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 2, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 2 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program

```


11.2.4 Change motion message rate

The demonstration program sends Motion Messages at a default rate of 10 Hz. You can change this rate:

Type “12” and press **Return**

SimGEN responds with a message that shows the current rate selection and prompts for a new rate.

Enter your desired rate as a period in milliseconds and press **Return**.

Command 12 in the command list will show the new rate.

Figure 11-7: Changing the motion message rate

```

C:\> Command Prompt - port_ieee_test.exe
>12
Card status = ok Simulation status = 2
Current message rate = 100ms
Enter new message rate in ms: 10

1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 10ms
0. Quit program
>8

```

11.2.5 Confidence test of the timer card

This test can be used to confirm that the PCI-215 Timer card is operating correctly and that it has effective cable connections to the signal generator. Before running this test, you must ensure:

- The signal generator is powered up
- The Timer Card Input cable is connected to the 1PPS output of the signal generator
- The Timer Card Output cable is connected to the Trig In input on the signal generator.

To confirm trigger pulses are being received, first select a modelled motion scenario in SimGEN by setting the trigger mode for the signal generator to **Delayed** in the **Hardware Configuration** menu.

Start the Scenario by clicking the SimGEN **Run** button. SimGEN progresses to the stage where the Status message flashes Red with the symbol 'Trigger Wait'.

At the remote PC start the test by typing “11” and press **Return**.

The test sequence checks that appropriate clock signals are being received and then sends a trigger pulse to the signal generator, which starts running the scenario.

The SimGEN status changes to Green with the message 'Running'.

At the remote PC press the **Space Bar** to stop the trigger pulses.

The test displays the simulation time until you press the **Space Bar** again.

Figure 11-8: Testing the timer card

```

C:\Program Files\SimGEN\Remote IEEE488\Release>
** SimGEN Remote IEEE488 example program. Iss 1.04 2/6/04 **

1. Select scenario my.scn
2. Enter new scenario name
3. Enter new trig mode
4. Set trig mode 0, then Run scenario
5. Stop scenario
6. Rewind
7. Read from simulator
8. Set trig mode 0 then run with motion
9. Set trig mode then run with dual motion
10. Set trig mode then run with triple motion
11. Test timer
12. Change remote message rate from 100ms
0. Quit program
>11
Data read: <msg>
          <status> 2 </status>
</
Sim response="<msg>
          <status> 2 </status>
</", target is SimGEN

Card status = ok Simulation status = 2
Board 0 Type 215 IRQ 17 BASE -8928 RESULT 0
PCI215 Tests...
Testing for clock ticking
...Passed
Testing ms clock rate
highest ms count = 998
...Passed
Testing 1 second clock rate
Time before= 2495, after= 3494, diff= 999
...Passed
Testing 1PPS IN rate
...Passed
Sending Start <trigger> pulses, hit a key to end
Pulses stopped
Time= 18495 Diff= 18495
Reading time at about 1000ms intervals, hit a key to end
Time= 19495 Diff= 1000
Time= 20495 Diff= 1000
Time= 21495 Diff= 1000
Time= 22495 Diff= 1000
Time= 23495 Diff= 1000
Time= 24495 Diff= 1000
Time= 25495 Diff= 1000
Time= 26495 Diff= 1000
Time= 27495 Diff= 1000
Time= 28495 Diff= 1000

```

11.2.6 Run a remote motion demonstration

The *port_ieee_test.exe* program has the facility to generate three examples of remote motion. These are for a single vehicle and for 2 or 3 vehicles.

Note: You can only run the 2 and 3 vehicle examples if the installed hardware has the necessary multiple output capability.

In each case, the vehicle motion starts at Latitude 0 degrees, Longitude 0 degrees and proceeds in a straight line, at low velocity, in the z-direction.

Ensure you run SimGEN using a scenario using the Remote Motion Vehicle type. To run the dual or triple motion examples, you must specify 2 or 3 Remote Motion vehicles.

To run the single motion example:

Type "8", or "9" for the 2 vehicle example or "10" for 3 vehicles, and press **Return**.

The signal generator will be configured and the initial motion will be sent before arming the signal generator. The motion data is displayed and the ARM and RUN commands sent, together with a trigger pulse (if necessary) for the current trigger mode.

A 'Running' message is displayed when the scenario has started and the mark at the bottom of the screen is twirling, indicating the test is proceeding.

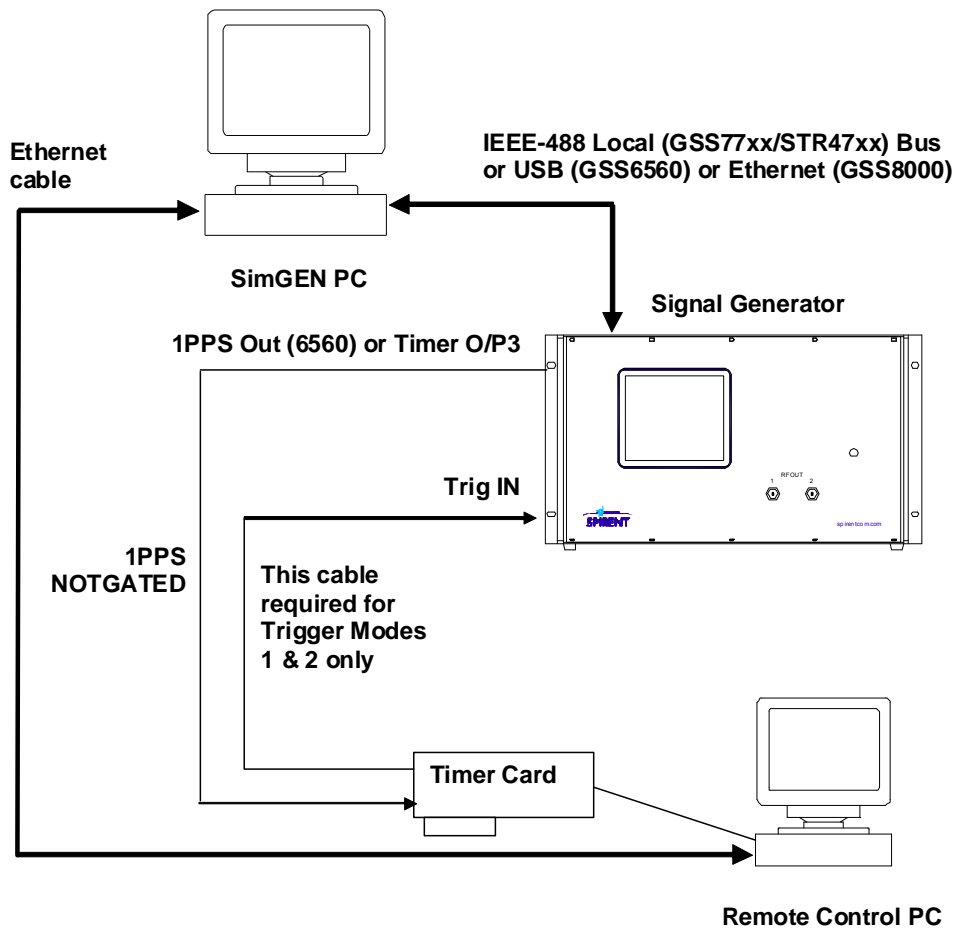
The motion example will continue until stopped by pressing any key. The scenario terminates and resets.

11.3 Test using the remote motion TCP/IP software

Note: The 1PPS NOTGATED cable must be connected correctly for the test/demonstration programs to function correctly. If the programs fail to start then check the 1PPS NOTGATED cable connections and the signal generator timer output configuration.

This is an example program, similar to the IEEE demonstration program. It communicates with the SimGEN PC using an Ethernet network, rather than an IEEE-488 bus see Appendix Figure C-10.

Figure 11-10: TCP/IP test configuration



The program is called `remote_socket_test.exe` and is installed in the default folder `C:\Program Files\Spirent Communications\PosApp\Remote Control Demo Software\Remote Motion TCP/IP\Release`.

You can run this program on the SimGEN controller in parallel with SimGEN or it may be run on a separate remote PC. When the program is run on the SimGEN controller the program can use the local host IP address (127.0.0.0).

The test program operates as a socket client with SimGEN as the server so it is essential you run SimGEN first. Before starting `socket_remote_logging`, start the server by selecting **Options-Remote Command Settings**, then select **Enable remote input** and **Enable BSD sockets**. The test program can then connect to the server.

To operate the program from a remote PC you must get the IP address of the SimGEN controller by typing the command `'ipconfig'` in a Command Prompt window on the SimGEN controller. The `ipconfig` command displays details of the TCP/IP configuration of the PC, including the current IP address.

Note: Some networks use IP addresses that are assigned dynamically and the address changes each time the PC is booted.

The program keeps a record of the IP address in the text file server_ip_address.txt.

The first time the program runs, it prompts you to enter the IP address of the SimGEN controller. If the IP address allows SimGEN to be located, the program saves this address and continues with that address.

Subsequently, after starting the program, it initially uses the saved address to locate the SimGEN controller.

If the stored address does not reveal the SimGEN controller, the program will prompt for the IP address.

Figure 11-11: Menu of remote motion software TCPIP

```

C:\PROGRAM~1\SPIREN~1\SimGEN\REMOTE~1\Remote Motion TCPIP\Release>remote_socket_t
est.exe
Remote Socket Test V1.06 26/03/04 Selecting winsock 1.1
Creating socket
Connecting to server... Can't connect to server

Please enter the dotted ip address of the SimGEN server and press Return
Use 127.0.0.1 to connect to the local copy of SimGEN
Example, Note there should be no leading zeros!
123.456.1.12
:10.24.1.12
...connected!
Calling init timer card fn
Board 0 Type 215 IRQ 17 BASE -8928 RESULT 0
Timer Card initialised

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario my.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
X. Quit program
>

```

The command sequence necessary to perform a confidence test with this program is the same as that described for the IEEE test program in section C.2.

Type "4" and press **Return** to start the test. Type "5" to stop the test and type "6" to reset SimGEN.

11.4 Commands

The remote motion program TCPIP provides commands to:

- a) Change the trigger mode.
- b) Change the current scenario loaded into SimGEN.
- c) Run remote motion scenarios with one or two vehicles.

These commands operate in the same manner as the equivalent commands described for the IEEE demonstration software.

Note: With this program, the direction of motion reverses every 2 minutes.

11.4.1 Change the RF power level settings

Three commands enable you to:

- a) Turn the Power On/Off for either All Channels or a selected Channel,
- b) Toggle between Absolute and Relative power level assignment,
- c) Sets the Power level for either All Channels or for a selected Channel.

For example, to turn the power off for Channel 5:

Type “9” and press **Return**.

SimGEN responds with the prompt: *All channels [N]*

Press **Return** to change only one Channel and SimGEN responds with: *By channel, not TXID [Y]*

Press **Return** and at the *channel >* prompt type “5” and press **Return**.

SimGEN will prompt with *on [Y]*, type “n” and press **Return** to turn the Channel off.

Figure 11-12: Changing the RF power level

```

Command Prompt - remote_socket_test.exe
X. Quit program
>9
All channels [N] >
by channel, not TXID [Y] >
channel > 5
on [Y] > n

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario my.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
X. Quit program
>

```

11.4.2 Stop/Start PRN code transmission

The command may be used to stop and re-start transmission of the PRN component of the RF signal for All Channels or for a selected Channel.

To turn off the PRN code generation for Channel 2:

Type “c” and press **Return**(or C, case is not important)

At the prompt *all channels [N]* press enter for not all channels

At the prompt *channel >* type “2” and press **Return**

At the prompt *PRN code on [Y]* > type “n” and press **Return**

11.4.3 Set start location for a vehicle

The command implemented in the example sets the start location for vehicle 1 to 20 Deg N, 30 Deg E at an altitude of 50 m.

To set this position:

Type “d” and press **Return**.

The command is executed without a response.

11.5 Log simulation progress to a remote PC

Note: *This has been superseded by the Data Streaming capability. The Data Streaming method is more efficient and puts less computing strain on the SimGEN controller. The facility described remains supported, but Spirent does not recommend it.*

A demonstration program that reads data from SimGEN as a simulation run progresses and writes the data to a log file is distributed with SimGEN.

The program is *socket_remote_logging.exe* and is installed in the Remote Logging TCPIP sub-folder of Remote Control Demo Software. The source code and project files for Microsoft Visual C++ are provided in addition to the executable.

The program as supplied is configured to operate on the same PC as SimGEN. To change the program to connect to an instance of SimGEN running on a separate PC it is necessary to change the program and re-compile. You will need to know the network name of the PC running SimGEN. At the SimGEN controller open a command prompt window and enter the command 'hostname', Windows will respond with the PC's network name.

To change the program, open the workspace using Visual C++ and search for the variable 'hostname'. Edit the associated string to be the name of the SimGEN controller and re-compile.

The program has no commands. To log data from a SimGEN scenario run:

Start the *socket_remote_logging.exe* program either in a Command Prompt window or by double clicking the filename in Windows Explorer.

The program is a simple Console application.

The test program operates as a socket client with SimGEN as the server so you must run SimGEN first. Before starting *socket_remote_logging*, select socket remote control **Options-Remote Command Settings** then select **Enable remote input** and **Enable BSD sockets** to start the Server. The test program will then be able to connect to the server.

The program prompts you to run a scenario on SimGEN, the program then logs a set of data items from 1 second into run to 15 seconds into run, logging data every 0.5 seconds. At 15 seconds the program terminates, writing the data to a file *data.txt*.

Each data item is obtained by issuing a data logging command. A set of commands must be issued every time a set of items is required (such as every 0.5 seconds or whatever logging rate is desired).

Each logging command may be prefaced with a timestamp (the time when the data is required) or "- " ("give me the data now")

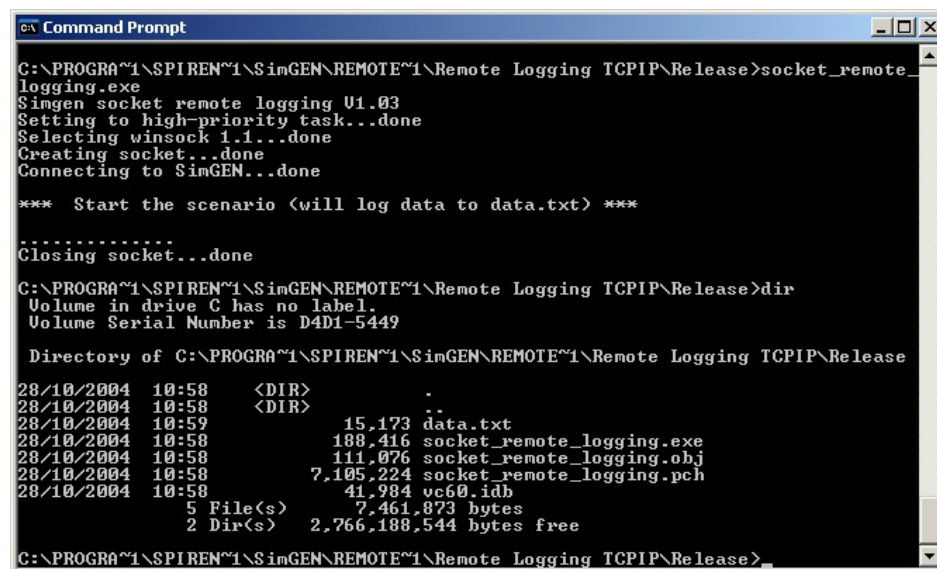
If you use "- " your program must have some way of determining when "now" is, to correctly attribute a time-of-validity to the returned data. Therefore using the timestamp option is preferred.

If data is requested with a timestamp (for example) of "0 00:00:30.00" at a time BEFORE 30 seconds into the run, SimGEN will not respond until that data is available (the calling program hangs on the read function until the appropriate time). If the request were made at a time LATER than 30 seconds into the run, the call will return the latest available data, with no error indication.

In the example program all data items in the list are requested prefaced by the appropriate timestamp. At the end of the list an additional command "TIME" is sent which returns the current scenario time. If this does not correspond to the requested time it indicates that not all the items were collected within the required time-period. Spirent recommends you perform this check when logging long lists of items and you have set a higher rate, such as 10 ms, for the simulation period.

As supplied, the program writes the data to a file called *data.txt*. You can read this text file using Windows Notepad.

Figure 11-13: Running the remote logging example program



```

C:\PROGRA~1\SPIREN~1\SimGEN\REMOTE~1\Remote Logging TCPIP\Release>socket_remote_
logging.exe
Singen socket remote logging U1.03
Setting to high-priority task...done
Selecting winsock 1.1...done
Creating socket...done
Connecting to SimGEN...done

*** Start the scenario <will log data to data.txt> ***

*****
Closing socket...done

C:\PROGRA~1\SPIREN~1\SimGEN\REMOTE~1\Remote Logging TCPIP\Release>dir
Volume in drive C has no label.
Volume Serial Number is D4D1-5449

Directory of C:\PROGRA~1\SPIREN~1\SimGEN\REMOTE~1\Remote Logging TCPIP\Release

28/10/2004  10:58    <DIR>          .
28/10/2004  10:58    <DIR>          ..
28/10/2004  10:59             15,173 data.txt
28/10/2004  10:58           188,416 socket_remote_logging.exe
28/10/2004  10:58           111,076 socket_remote_logging.obj
28/10/2004  10:58           7,105,224 socket_remote_logging.pch
28/10/2004  10:58             41,984 vc60.idb
                5 File(s)              7,461,873 bytes
                2 Dir(s)          2,766,188,544 bytes free

C:\PROGRA~1\SPIREN~1\SimGEN\REMOTE~1\Remote Logging TCPIP\Release>

```


Chapter 12: Optional SCRAMNet card

You must install the optional SCRAMNet card and the SCRAMNet drivers and software in the SimGEN controller.

12.1 Install the SCRAMNet card

SAFETY WARNING



**BEFORE YOU INSTALL THE SCRAMNET CARD
COMPLETELY ISOLATE MAINS POWER FROM THE PC BY
REMOVING THE PLUG FROM THE MAINS SOCKET**



Use anti-static handling precautions when you handle the SCRAMNet card

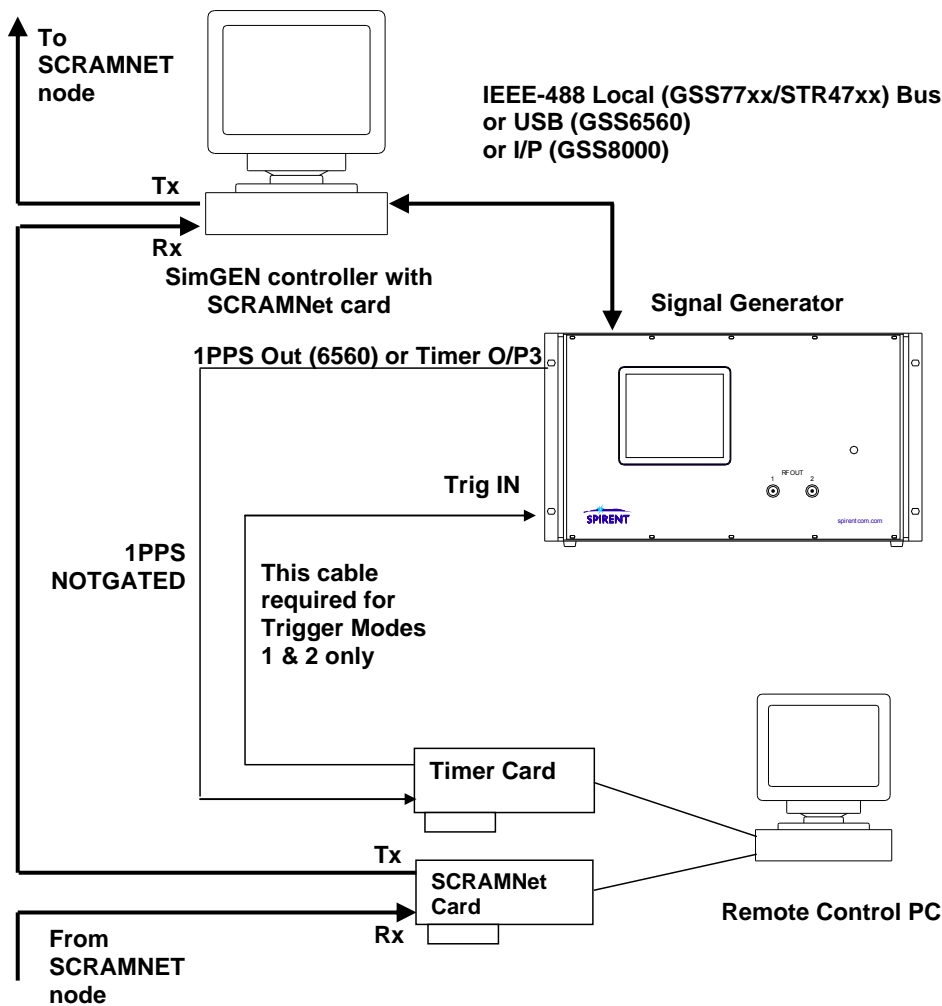
Notes:

- 1) Keep the ends of the fibre optic cables and the SCRAMNet card optical connector sockets clean. Always replace the cover on an optical fibre connector or socket when it is not in use.*
- 2) The 1PPS NOTGATED cable must be connected correctly for the test/demonstration programs to function correctly. If the programs fail to start then check the 1PPS NOTGATED cable connections and the signal generator timer output configuration.*

Refer to the PC manufacturer's hardware manual for instructions on how to install devices into a PCI slot. Refer to the hardware manual for any restrictions on installing cards to PCI slots.

Install the SCRAMNet card following the instructions provided by the manufacturer.

Nodes in a SCRAMNet network are connected together in a "daisy chain" fashion. The transmit side of the SCRAMNet card fitted to the SimGEN controller should be connected to the receive side of another node in the network, and the receive side of the SimGEN SCRAMNet card should be connected to the transmit side of the SCRAMNet card in another node see Figure 12-1

Figure 12-1 SCRAMNet test configuration

When connecting the SimGEN controller to the rest of the SCRAMNet network, handle the fibre optic cables with care. Fibre optic cables can be broken by being struck or bent through small radius turns.

Further detail on the installation of the SCRAMNet card can be found in chapter 4 of the SCRAMNet Hardware Reference, reference d), which you will receive with your SCRAMNet card.

12.2 Install SCRAMNet software and drivers

Notes:

- 1) There is a known problem using the a4 version of the SCRAMNet driver together with interrupts on some PCs that takes the form of the PC crashing when interrupts are enabled. If you encounter this problem, contact SCRAMNet support to obtain an earlier version of the driver.
- 2) The default SCRAMNet protocol does not use interrupts.

See reference e) for a detailed description of how to install the SCRAMNet software and drivers.

After installing the software and drivers, carry out a confidence test with applications from Systrans. First check the values stored in the PC registry for the SCRAMNet card's node ID, timeout and memory size. Click **Start-Programs-SCRAMNet+-SCRAMNet+ Windows Installation**, and then click on **Edit**.

Figure 12-2: SCRAMNet Windows installation dialog

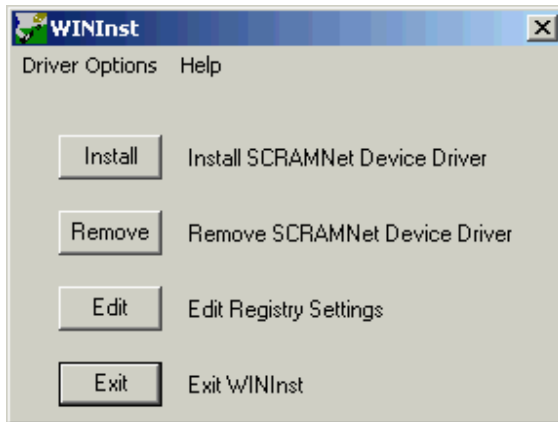
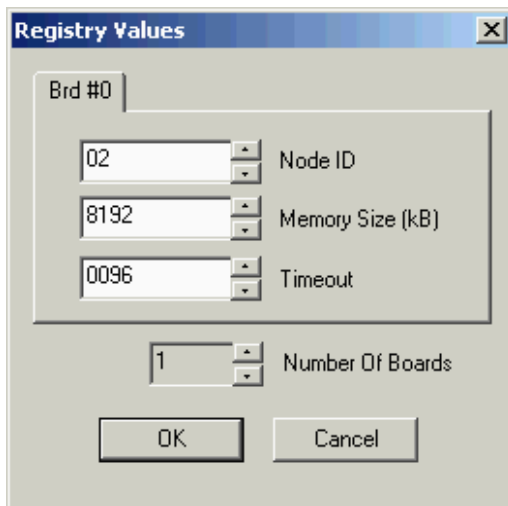


Figure 12-3: SCRAMNet registry values



The node ID must be unique within the SCRAMNet network. A node ID of zero specifies the control node. The SimGEN controller should not have a node ID of zero. Check that the memory size matches that of your SCRAMNet card.

If any of the following tests fail, check that these registry values are correct and the SimGEN controller is connected correctly to the rest of the network. If the problem is still not rectified, consult SCRAMNet technical support. Details of how to contact them are available in their documentation.

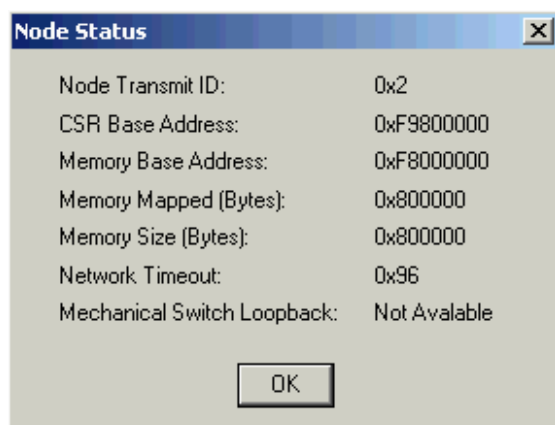
12.3 Confidence test 1

Note: The tests can take ten minutes to complete.

This will test that it is possible to write to the memory in your SCRAMNet card and to read from it.

Open the windiags application by selecting **Start-Programs-SCRAMNet+ - SCRAMNet+ - Diagnostics**. A window shows the status of the node. Click **OK** to proceed.

Figure 12-4: Node status window



In the **WinDiags** dialog, see Figure 12-5, select **All Memory Tests** and click **Run**. After you click **Run**, you must select a folder for the results to be written to.

Figure 12-5: WinDiags dialog

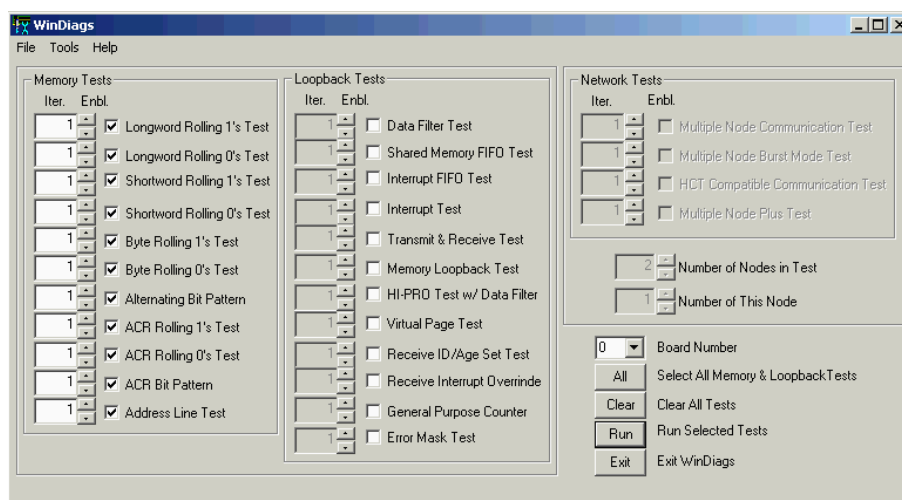


Figure 12-6 shows the window you will see while **WinDiags** is running the tests

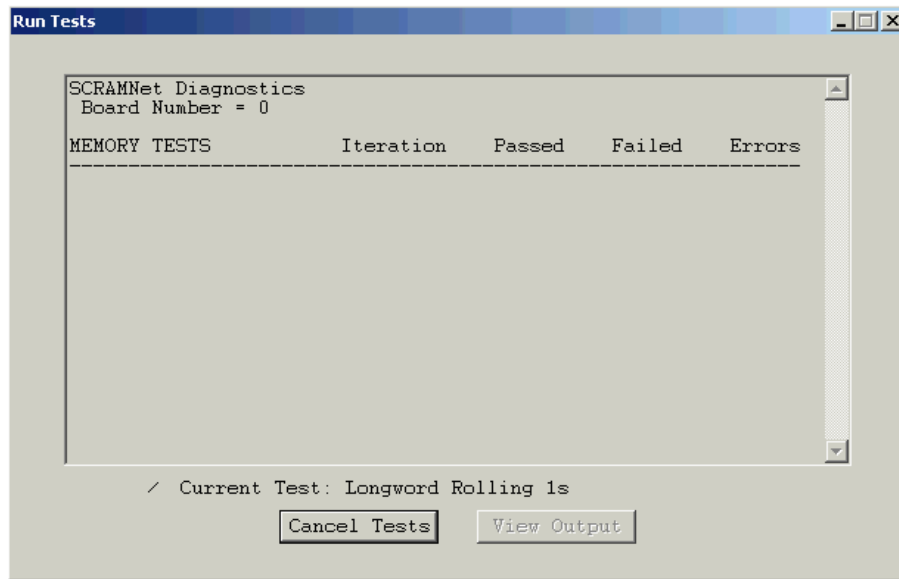
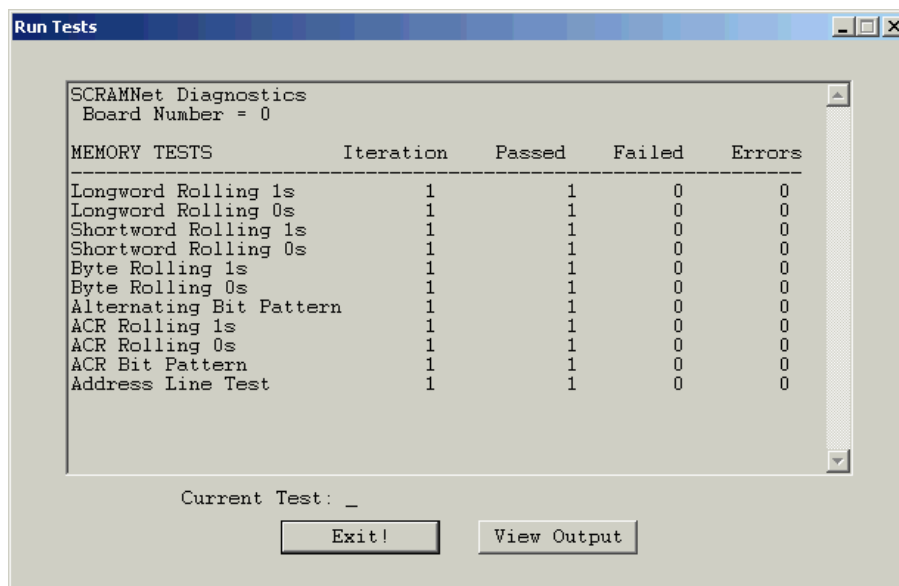
Figure 12-6: WinDiags Run Tests window

Figure 12-7 shows the window you will see after the tests have finished.

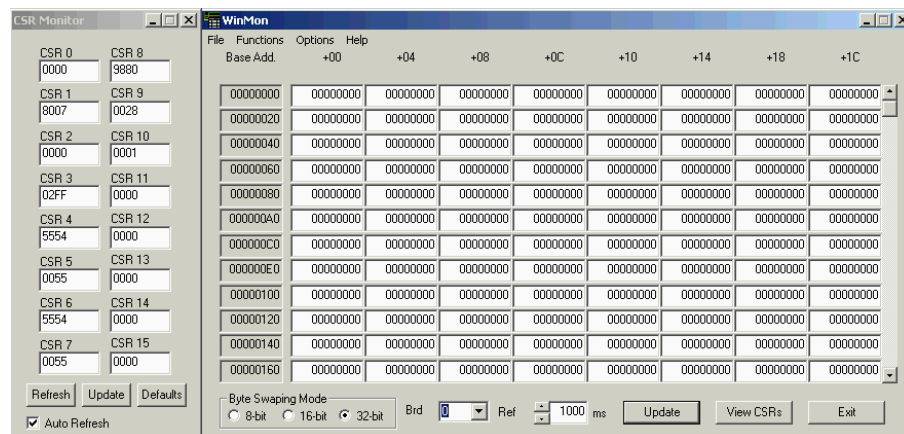
To exit the test click **Exit!**.

Figure 12-7: WinDiags Run Tests - tests completed window

12.4 Confidence test 2

Open the **WinMon** utility (**Start-Programs-SCRAMNet+-SCRAMNet+ -Monitor**), see Figure 12-8.

Figure 12-8: SCRAMNet WinMon dialog



WinMon comprises two areas:

The larger area shows the contents of the SCRAMNet memory

The smaller area shows the state of the CSRs (Control Status Registers).

See appendix B of the SCRAMNet Hardware Reference, reference d), for a full description of CSRs.

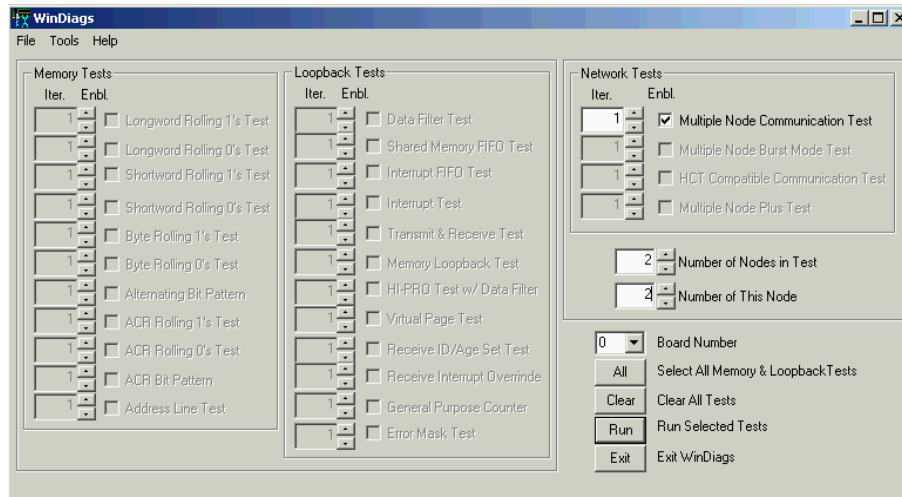
Now, open another instance of this program on another PC in the SCRAMNet network. Check that, when values are written to the memory of one card, they are copied to the other.

12.5 Confidence test 3

Note: The program expects node 0 to be started second.

Run **WinDiags** and run the multiple node communication test on the SimGEN PC. You will need to run this test on node 0 in the network at the same time.

Figure 12-9: WinDiags communication test



If these tests were successful, try using the program supplied by Spirent, *Scramnet_std_remote_QPM939.exe* to send SimGEN some basic commands. Section D.6.1 describes this program.

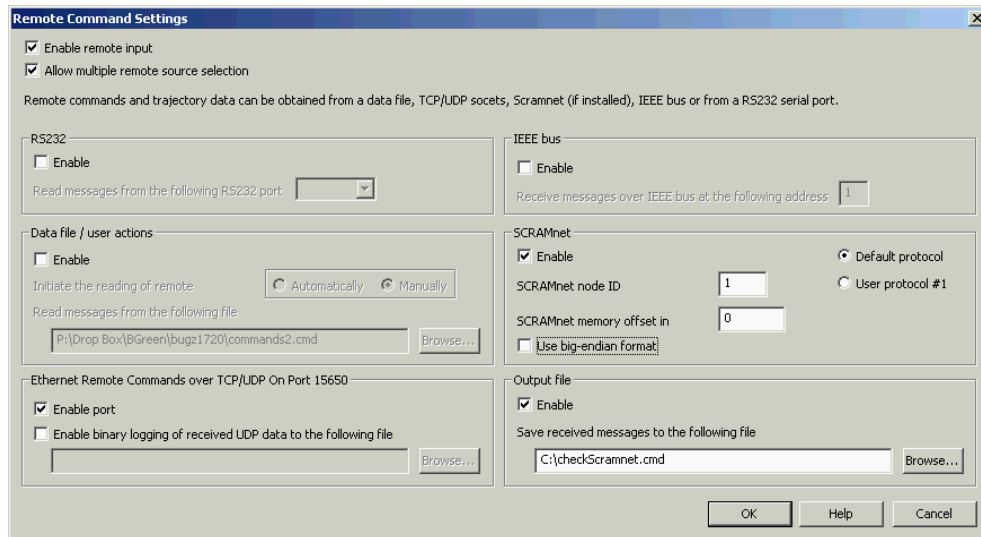
12.5.1 Scramnet_std_remote_QPM939 test program

In order to send SimGEN commands using SCRAMNet, you must select **SCRAMNet** from **Options-Remote Commands Settings**.

In the SCRAMNet area, select **Enable** and type a unique **SCRAMNet node ID**.

To use the Scramnet_std_remote_QPM939 test program, it is necessary to use a **SCRAMNet memory offset** of zero. If you need to use a non-zero memory offset, the code for the test program must be altered accordingly (see file Scramnet.cpp). If you use the big-endian format in SimGEN, you must also use this format in the *Scramnet_std_remote_QPM939* test program.

Figure 12-10: Enable the SCRAMNet interface for SimGEN



The test program is located in the installation folder *C:\Program Files\Spirent Communications\PosApp*. The name of the executable file is *scramnet_remote_end.exe*.

Copy this file onto another PC in the SCRAMNet network and run it.

To use the “run with motion” feature, copy the optional file *motion.txt* to the same location in other PC.

Note: If you do not copy this file to the PC, a default motion file is created.

Scramnet_std_remote_QPM939 is a simple command line utility, see Figure 12-11.

Figure 12-11: Scramnet_std_remote utility

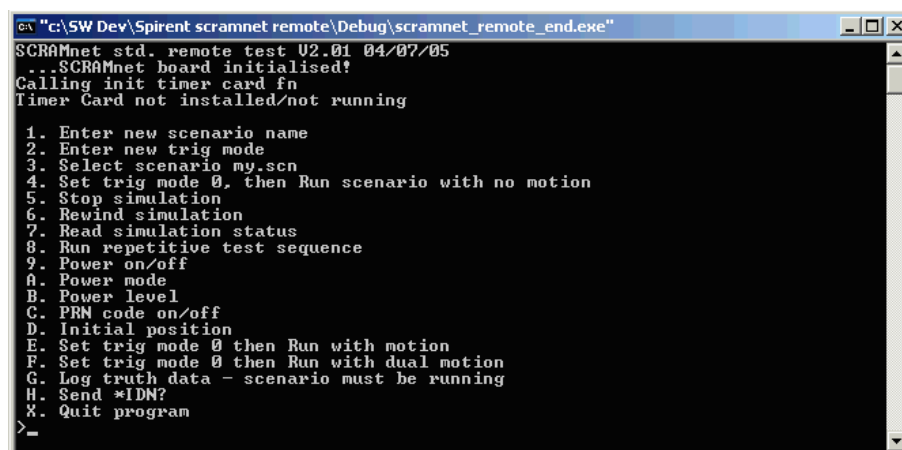


Figure 12-11 shows that no timer card is present in the SimGEN PC. You can run SimGEN with this hardware configuration, however there will be no 1PPS signal for synchronisation and SimGEN will use the PC clock.

12.6 Get SimGEN scenario status

Note: This is a useful way of determining whether the system is working correctly.

Get the scenario status by sending the NULL command (see section 5.2.13).

Choose option 7 from the menu. SimGEN returns an XML string containing the scenario status. Appendix Table D-1 details the scenario status number.

Table 12-1: Scenario status responses

State Code	Scenario state
0	Not loaded
1	Not completed loading
2	Idle, ready to run
3	Arming
4	Completed arming; or waiting for a command or trigger signal to start
5	Running
6	Paused
7	Stopped and not reset; waiting for further commands.

In Figure 12-12, "2" indicates that SimGEN is ready to run.

Figure 12-12: Get SimGEN scenario status

```

c:\SW Dev\Spirent scramnet remote\Debug\scramnet_remote_end.exe
SCRAMnet std. remote test U2.01 04/07/05
..SCRAMnet board initialised!
Calling init timer card fn
Timer Card not installed/not running

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario my.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>7

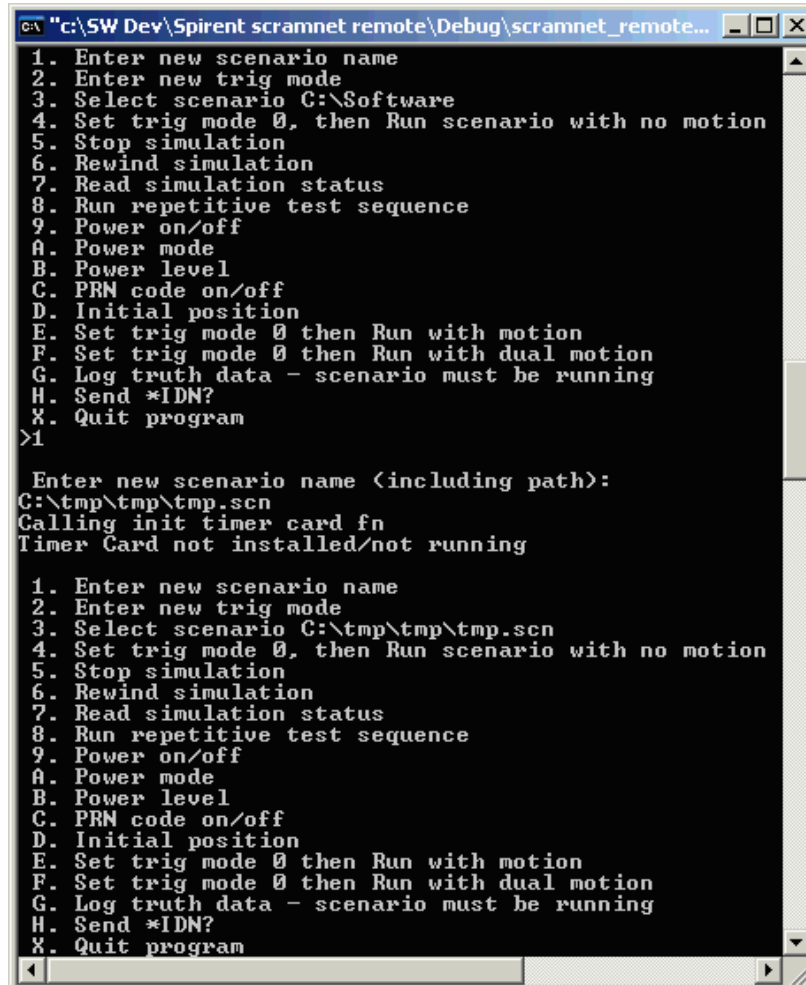
Status return:<msg>
      <status> 2 </status>
</msg>

```

12.7 Change scenario

To select a new scenario for SimGEN to run, select option 1 from the menu and enter the full path for the scenario you want to run, see Figure 12-13. The name of this scenario should replace the default *my.scn* in the text for option 3.

Figure 12-13: Entering a new scenario name



Select option 3. There will be no response from SimGEN, but the new scenario should load.

12.8 Run a scenario

Select option 4 from the menu shown in Figure 12-14.

The test program verifies that the trigger command has been sent and the run command has been sent.

There will be no response from SimGEN, but the scenario will start running. At this point, getting the simulation status should yield the result "5" (see Appendix Table D-1), which indicates a scenario is running.

Figure 12-14: Sending a run command to SimGEN

```

C:\SW Dev\Spirent scramnet remote\Debug\scramnet_remote...
1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>4
Trig command sent
Run command sent
Calling init timer card fn
Timer Card not installed/not running

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>7

Status return:<msg>
<status> 5 </status>
</msg>
?
Calling init timer card fn
Timer Card not installed/not running
  
```

To stop the scenario, select option 5 from the menu, see Figure 12-15.

There will be no response from SimGEN, but getting the simulation status will give the result "7", indicating that the scenario state is "ended" (see Appendix Table D-1).

Figure 12-15: Stopping SimGEN

```

c:\SW Dev\Spirent scramnet remote\Debug\scramnet_remote...
1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>5
Calling init timer card fn
Timer Card not installed/not running

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>7

Status return:<msg>
      <status> ? </status>
</msg>
!
Calling init timer card fn
Timer Card not installed/not running

```

Rewind the scenario so that SimGEN will be ready to run again by selecting 6 from the menu, see Figure 12-16.

Figure 12-16: Rewinding the SimGEN scenario

```

c:\SW Dev\Spirent scramnet remote\Debug\scramnet_remot...
1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>6
Calling init timer card fn
Timer Card not installed/not running

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>7

Status return:<msg>
      <status> 2 </status>
</msg>

```

Getting simulation status gives the result “2” (see Appendix Table D-1), showing SimGEN is ready to run

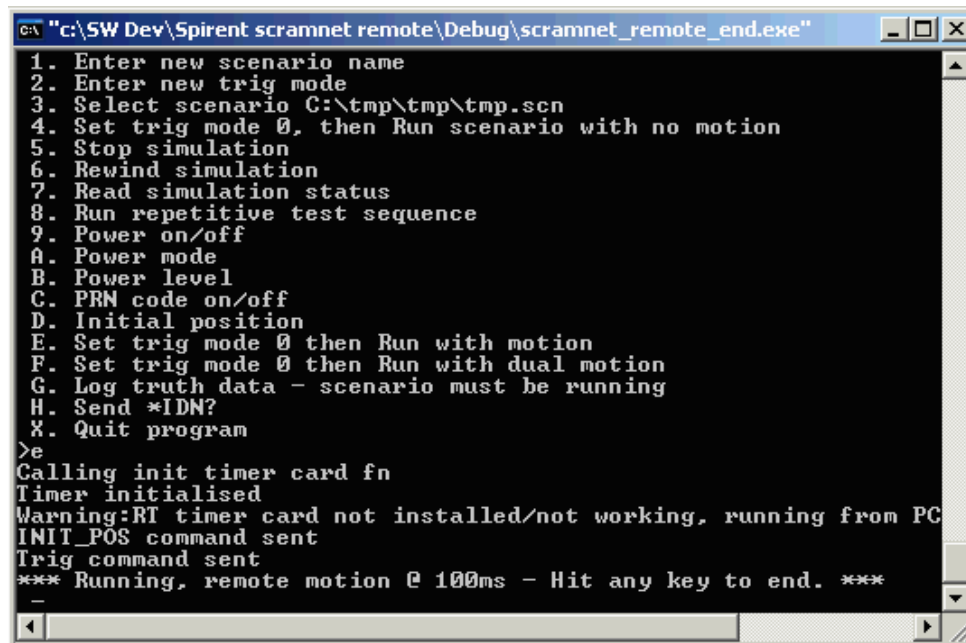
12.9 Run with motion

Note: The command **INIT_POS** is valid for moving remote vehicles, see section 5.2.6

In order to run with motion, the scenario loaded into SimGEN must have **vehicle** set to **remote vehicle**, see Figure 12-17. Select option E from the menu.

The mark at the bottom of the screen should be twirling, showing the SimGEN scenario is running and the vehicle is moving. Pressing any key automatically sends “end” and “rewind” commands to SimGEN.

Figure 12-17: Running with motion



The motion of the (remote) vehicle corresponds to the motion commands in the file *motion.txt*. In order to specify different motion, create a different *motion.txt* file.

Example motion.txt file:

```
RU  
2.125,mot,v1_m1,6378137,0,0.060000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
3.791,mot,v1_m1,6378137,0,0.050000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
. .  
. .  
. .  
-.EN
```

Section 5.2.8 details the “RU” command

Section 5.15.2 details the “MOT” command (all data after text “mot”).

Section 5.1.1 details the “TIMESTAMP” command (data before “MOT” command, this example uses floating-point seconds).

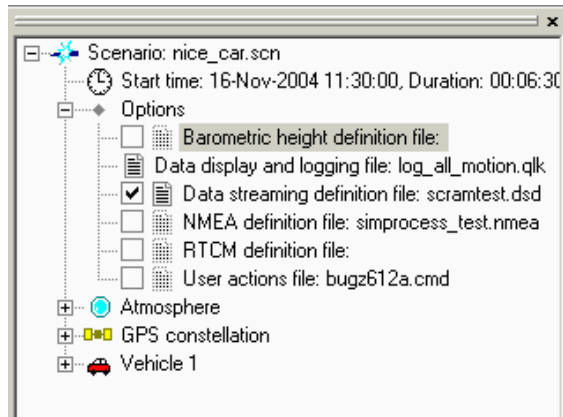
Section 5.2.10 details the “EN” command

12.10 Log truth data

Select and run a scenario as shown in sections D.8 and D.9.

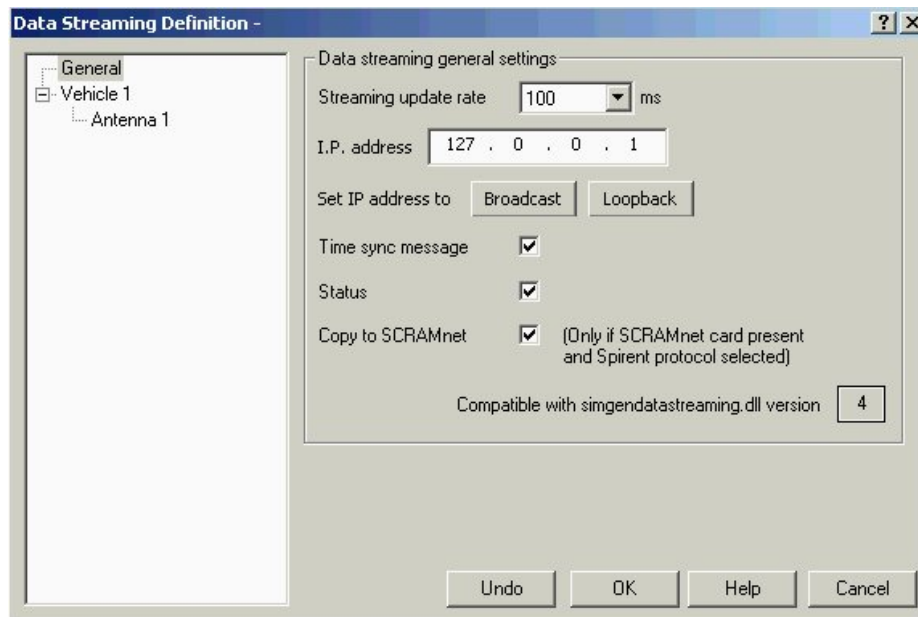
Select **Options -Data streaming definition file** in the scenario tree, see Figure 12-18.

Figure 12-18: Enable data streaming



In the **Data Streaming Definition** dialog, see Appendix Figure D-18, select **Copy to SCRAMNet**.

Figure 12-19: Copy to SCRAMNet



Once the scenario is running, select option **G** from the menu shown in Figure 12-20. The message shown appears and the cursor should spin. When a key is pressed to end the logging the total number of messages received is displayed. The truth data is logged to the file `scramnet_truthdata_log.csv`, located in the same directory as the executable file. At present all the different types of truth data message are logged to the same file. To change this or to change the location or name of the file the data is logged to, it is necessary to change the code of this utility.

Figure 12-20: Logging truth data

```

c:\SW Dev\Spirent scramnet remote\Debug\scramnet_remote_end.exe
END command sent
ReWind command sent
Calling init timer card fn
Timer Card not installed/not running

1. Enter new scenario name
2. Enter new trig mode
3. Select scenario C:\tmp\tmp\tmp.scn
4. Set trig mode 0, then Run scenario with no motion
5. Stop simulation
6. Rewind simulation
7. Read simulation status
8. Run repetitive test sequence
9. Power on/off
A. Power mode
B. Power level
C. PRN code on/off
D. Initial position
E. Set trig mode 0 then Run with motion
F. Set trig mode 0 then Run with dual motion
G. Log truth data - scenario must be running
H. Send *IDN?
X. Quit program
>g
*** Logging truth data - Hit any key to end. ***
!
```

12.11 Other commands available

It is also possible to turn the power of some or all of the satellites on or off, toggle the mode of some or all of the satellites between “absolute” and “relative”, set the power level for some or all satellites and turn off the PRN code for some or all satellites. These commands work like the corresponding options described in section C.3 and guide you using a series of simple questions.

It is possible to use this utility to run a repetitive test sequence that will load one scenario, run stop, rewind, load a second scenario, run, stop and rewind repeatedly. In order for this to work, there must be two scenarios called TEST1 and TEST2 adjacent to the current scenario.

This utility can set the trigger mode (option 2). See sections 7.3 to 7.5 for a description of how SimGEN will behave in various trigger modes.

Option D sets the initial position of the vehicle to 20 degrees latitude, 30 degrees longitude and 50 m above sea level. Send this command before SimGEN runs a scenario. It is only valid for static vehicles.

Option F, run with dual motion, works exactly the same way as running with motion. You must ensure the scenario uses remote vehicles for vehicle1 and vehicle2.

Option H sends SimGEN a “IDN?” message. The XML string returned should contain the data “Spirent, SimGEN for Windows,0,VX.XX”, where VX.XX is the version of SimGEN you are using.

12.12 Remote commands for SCRAMNet

This section details remote commands specifically for use with the SCRAMNet interface.

12.12.1 MOTBIN

Description	<p>This command is a SCRAMNet version of the MOT command in section 5.10.1.</p> <p>MOTBIN defines vehicle motion using binary motion data; whereas the MOT command converts motion data to and from ascii, which can be slow.</p> <p>The motion parameters for this command (and the units) are the same as the MOT command in section 5.10.1, except the parameters use binary data. For this reason, there is no example command.</p> <p>Note: <i>The first MOTBIN position message determines the motion of the vehicle at the start of the scenario, even if the timestamp of the message does not match the scenario start time. Subsequent messages will be used correspondingly.</i></p>
Format	<code><timestamp>,MOTBIN,<veh_mot>,<binary_data></code>
Where	<p><code>timestamp</code> ::= see section 5.1.1 "B" (uses the <code>binary_timestamp</code> field of <code><binary_data></code>)</p> <p><code>veh_mot</code> ::= <code>vehicle_id</code>"_"<code>motion_id</code></p> <p><code>vehicle_id</code> ::= "v"<v_number></p> <p><code>v_number</code> ::= vehicle number starting from 1</p> <p><code>motion_id</code> ::= "m"<m_number></p> <p><code>m_number</code> ::= "1" (motion model number always 1)</p> <p><code>binary_data</code> ::= a fixed sequence of binary data in IEEE 754 double (8 byte) format. You can set to zero any <code>binary_data</code> field not required.</p> <p>The <code>binary_data</code> fields must be in this order, with no separators (fields grouped for clarity):</p> <p><code>binary_timestamp</code></p> <p><code>x_pos</code> ::= x, y and z position, metres <code>y_pos</code> <code>z_pos</code></p> <p><code>x_vel</code> ::= x, y and z velocity, m.s^{-1} <code>y_vel</code> <code>z_vel</code></p> <p><code>x_acc</code> ::= x, y and z acceleration, m.s^{-2} <code>y_acc</code> <code>z_acc</code></p> <p><code>x_jerk</code> ::= x, y and z jerk, m.s^{-3} <code>y_jerk</code> <code>z_jerk</code></p> <p><code>heading</code> ::= heading, bank and elevation, radians <code>bank</code> <code>elevation</code></p> <p><code>x_ang_vel</code> ::= x, y and z angular velocity, rad.s^{-1} <code>y_ang_vel</code> <code>z_ang_vel</code></p> <p><code>x_ang_acc</code> ::= x, y and z angular acceleration, rad.s^{-2} <code>y_ang_acc</code></p>

Returns

```
z_ang_acc
x_ang_jerk ::= x, y and z angular jerk, rad.s-3
y_ang_jerk
z_ang_jerk
status see returned response format, section 5.1.2
```

Chapter 13: VMS remote command compatibility

This section details the remote command compatibility between SimGEN for Windows and SimGEN for VMS

Currently, SimGEN For Windows does not directly support all the legacy SimGEN for VMS remote control commands. Appendix Table E-1 compares the available remote commands.

Table 13-1: Comparison of remote commands used in SimGEN for Windows and SimGEN for VMS

VMS Cmd	Windows Cmd	Discussion
SC ,filename	SC (see 5.2.3)	On VMS logical path names are used. On the PC a full path name is used.
SP ,0,x,y,z SP ,1,lat,long,height (Note: the number after the command denotes the type of parameters)	INIT_POS (see 0)	SimGEN does not currently have an ECEF co-ordinates type for this command.
IN , DD-MMM-YYY HH:MM:SS, x,y,z, (antenna offsets) h,e,b (body axis) (Note that the offsets and the body axis can be repeated for each antenna)	START_TIME (see 5.3.5)	SimGEN currently does not have a command to set the antenna offsets from vehicle CofG or its orientation w.r.t body axis.
MC ,n1,n2,n3,n4 n1 – remote motion data intervals in ms. n2 – simulation data message in ms. n3 – linear motion integration level. n4 – angular motion integration level. n5 – simulation data advance in ms.	Partially implemented via OFFSET_PROC_TIME see 5.3.13	In SimGEN VMS the simulation data advancement can be specified by this command. In SimGEN the data is expected one time step in advance. In SimGEN the time steps are specified in the GUI (as simulation iteration rate) and currently cannot be remotely controlled.
MA ,time-tag, PX,PY,PZ, VX,VY,VZ, AX,AY,AZ, JX,JY,JZ, h,e,b, rollrate,pitchrate,yawrate, rollaccel,pitchaccel, yawaccel Where time-tag is a count of the 100 ms intervals. The MB message is the same apart from the position is expressed in lat, long and height.	MOT (see 5.8.1) MOTB (see 5.8.2)	SimGEN command can be used at any select rate.
DM ,<same parameters as MA/MB for vehicle 1>,<same parameters as MA/MB for vehicle 2>	Same as MOT command above.	SimGEN VMS can optimise motion control by sending a DM command, which sends two sets of motion data in one transaction. This can only be achieved by sending two mot commands with SimGEN.
TR ,mode	TR (see 5.2.7)	Same

VMS Cmd	Windows Cmd	Discussion
RU	RU (see 5.2.8)	Same
EN , loop, save_sim_data, save_rx_data	EN (see 5.2.10)	<p>These two commands do almost the same thing. However they take different values for their parameters.</p> <p>SimGEN does not need to use a NU message to indicate how long a scenario should run for as the EN command has an associated timestamp.</p>
AR	AR (see 5.2.12)	In SimGEN VMS the AR command is initiated by the first MA command i.e. MA,0 but this is not the case for the MOT command so an AR would have to be sent whatever for SimGEN.
RS	<p>NULL (see 5.2.13)</p> <p>This command returns the status and any pending data.</p> <p>TIME (see 5.3.1)</p> <p>This command returns the time into run.</p>	There is no single RS command for SimGEN. The status is returned for every command.
KI	No equivalent command.	This command kills the remote input task.
<p>WA,enable_state</p> <p>enable_state – if set to 1 informs SimGEN VMS to processes subsequent commands when SimGEN falls into the correct state.</p>	No equivalent command.	To implement this with SimGEN the client would have to poll the status with the NULL command.
<p>CO, [command_string], [sysin_string], [sysout_string], [wait_flag], [wait_time]</p> <p>command_string – string containing the command to be spawned.</p> <p>sysin_string – string assigned to the standard input of the spawned process.</p> <p>sysout_string – string assigned to the standard output of the spawned process.</p> <p>wait_flag – flag to make parent process wait for initialisation.</p> <p>wait_time – maximum wait time in seconds.</p>	No equivalent command.	.

VMS Cmd	Windows Cmd	Discussion
LO , logging_enabled logging_enabled – set to 1 to turn on logging and 0 to turn off logging.	No equivalent command.	Currently the only log control a remote user has in SimGEN is that of the EN command which allows the user to save the logs or not. Logging can however be timed control from SimGEN via the data display and logging file .
PR PA , op, svid, level op – indicates the frequency to be adjusted i.e. L1 or L2 or both. SVID – satellite id level – required power PR – power relative PA – power absolute The SVID and the level can be repeated for more than one satellite.	POW_LEV (see 5.5.5)	SimGEN command works on a satellite or channel number and can be applied to all satellites/channels simultaneously. If individual control is needed then more than one command must be sent, but they could all share the same timestamp.
ST , type, svid, error, [time] type – can be 0 for immediate and 1 for timed. svid – satellite id. error – size of pseudorange error in metres, time – if type 1 then the time in seconds into run to be applied.	No equivalent command.	The PR_RAMP (see 5.6.9) command in SimGEN will effectively let you accomplish a step error.
RA , type, svid, reset, ramp, dur, time type – 1 for timed 0 for immediate. svid – satellite id. reset – pseudorange error at the end of the ramp is held or set to zero. ramp – size of pseudorange ramp error in m/s dur – duration of ramp in seconds. time – if timed, time in seconds into run when applied. Note that the SVID etc can be repeated so this command can act on many satellites.	PR_RAMP (see 5.6.9)	The SimGEN command can only act on one or all satellites at any one time, but the commands can be preloaded at 0 time so that the start time for each satellite is the same; which would cause the same effect.

VMS Cmd	Windows Cmd	Discussion
SW,enable,channel_list enabled – if set to 1 turn the channel on 0 to turn it off. channel_list – list of channels to be turned off and on.	POW_ON (see 5.5.3)	SimGEN allows the turning off of satellites or channels. SimGEN VMS can list the channels it wants turned off in one command.
SA,ant_n ant_n – antenna and number to switch to.	ANT_PAT_NUM (see 5.4.1)	
DR,num[start_time][,repeat interval], [variable part] num – specifies the type of data required see commands 22 to 28. start_time – the requested data will be returned this number of seconds into run. repeat_interval – the requested data will be continually returned after this number of seconds. variable_part – see 22 to 28 in <> brackets.	SimGEN has a separate list of commands and does not support one command with various parameters.	
DR,0,<veh,ant,sig> 0 – requests a list of satellites being simulated for a specified antenna.	SIG_TXID (see 5.11.3)	The SimGEN version requires the remote user to go through a list of channels and check that the response is either a valid SVID or contains an “-1” meaning unallocated.
DR,1,<type,sig,svid> 1 – requests the almanac for a specified satellite.	No equivalent command.	
DR,2,<veh,ant,sig> 2 – requests the number of keyword sources available on a specified antenna.	No equivalent command.	
DR,3,<veh,ant,sig,source> 3 – requests the keyword source information. The name of the keyword and the number of keywords in the source are returned.	No equivalent command.	
DR,4,<veh,ant,sig,source,kwid> 4 – requests the name of the keyword.	No equivalent command.	
DR,5,<veh,ant,sig,source,svid_or_chan,use_svid,kwid> 5 – keyword data request.	Not one command.	
DR,6 6 – The message requests the motion mode for all vehicles. i.e. whether the motion is modelled, remote etc.	No equivalent command.	

VMS Cmd	Windows Cmd	Discussion
MP ,Mn,chan,svid,power,delay Mn – <ol style="list-style-type: none"> switch off multipath, reflector pattern multipath ground reflector fixed offset in power or delay chan –simulator channel number to be used for the multipath signal svid – satellite to be reflected power – power loss of selected signal delay – delay to be applied for reflected signal	SWITCH_SAT (See 5.6.2)	
NM ,sat_type,sv,start_time,end_time,word,subframe,page,bit_map sat_type – satellite type sv – satellite svid start_time – time at which change becomes effective end_time – time at which change ceases to apply word – word number to be modified subframe – subframe to be changed page – page number to be changed bit-map – 24 character string defining the changes to be made Note that GLONAS has a different format for the same message.	No equivalent command.	
Timestamp, NU	No equivalent command.	All timed commands in SimGEN take a timestamp, as their first parameter so there seems no need for this command in SimGEN.
Timestamp, PL ,veh,ant,type,svid,level veh – vehicle ant – antenna type – GPS,SBAS etc. svid – satellite ID level – power level	POW_LEV (see 5.5.5)	
Timestamp, PM ,veh,ant,type,svid,modelled_flag veh – vehicle ant – antenna id type – GPS,SBAS etc. svid – satellite id modelled_flag – 1 on, 0 off	POW_MODE (see 5.5.4) Note that the POW_LEV command can also be used to change the power mode.	

VMS Cmd	Windows Cmd	Discussion
<p>Timestamp, PR, type, svid, all_sats, start_time, use_start_time, start_state, offset, up_time, hold_time, down_time</p> <p>type – GPS, SBAS etc. svid – satellite id all_sats – 1 apply to all 0 don't start_time – start of modification use_start_time – 1 use 0 don't start_state – ramp start state offset – range offset value hold_time – seconds to hold down_time – end of ramp</p>	PR_RAMP (see 5.6.9)	
<p>Timestamp, SW, veh, ant, type, chan, chan_state, all_ants, all_vehs</p> <p>veh – vehicle ant – antenna type – GPS, SBAS etc. chan – channel number chan_state – on/off all_ants – apply to all antennas all_vehs – apply to all vehicles</p>	POW_ON (see 5.5.3)	SimGEN can currently turn off all channels but not all vehicles.
<p>Timestamp, SS, veh, ant, type, svid, sat_state, chan, multipath_type, power_loss, range_offset</p> <p>veh – vehicle ant – antenna type – GPS, SBAS etc. svid – satellite id sat_state – normal, forced, banned, multipath, include chan – channel number multipath_type – pattern, reflection, offset power_loss – multipath power loss range_offset – multipath range offset.</p>	SWITCH_SAT (See 5.6.2)	
<p>Timestamp, CM, veh, ant, type, svid, chan</p> <p>veh – vehicle ant – antenna type – GPS, SBAS etc. svid – satellite id chan – channel number</p>	No equivalent command.	
<p>Timestamp, TU, on_off</p> <p>on_off – turn turbo mode on or off</p>	No equivalent command.	

VMS Cmd	Windows Cmd	Discussion
<p>Timestamp,PN,toa,veh, type,svid,l1_code_offset, l1_carr_offset, l2_code_offset, l2_carr_offset</p> <p>toa – time into run that the data applies veh – vehicle type – GPS/SBAS etc. svid – satellite id l1_code_offset – pr offset for l1 code m@toa l1_carr_offset – pr offset for l1 carrier m@toa l2_code_offset – pr offset for l2 code m@toa l2_carr_offset – pr offset for l2 carrier m@toa</p> <p>svid onwards can be repeated for a number of satellites.</p>	<p>MP_SWITCH and MOD See 5.6.3 and 5.6.4</p>	
<p>Timestamp,AN,toa,veh, type,svid,l1_level_offset, l2_level_offset</p>	<p>MP_SWITCH and MOD See 5.6.3 and 5.6.4</p>	

This page is intentionally blank

Chapter 14: Spirent Applications Support

During normal office hours, you can contact Spirent Applications Support in the United Kingdom, Asia Pacific and the USA by e-mail, fax or telephone. The Applications Support contact details are on the next page.

Spirent Applications Support can provide general information on any aspect of Spirent simulator hardware or software; or they can answer specific questions. Spirent Applications Support is available to all customers under Warranty or with a maintenance agreement. Please note that Spirent Applications Support is available to customers without a maintenance agreement, but Spirent is not obliged to provide solutions to any customer without a maintenance agreement.

Spirent Applications Support aims to respond (or, where necessary, initiate a System Report) to any query within one working day (local and national holidays permitting).

Spirent requests you tell them when your simulator behaves in an unexpected, unusual or unacceptable way; including suspected faults or errors in the hardware, software or documentation.

Spirent's fault reporting system will ensure Applications Support can quickly return your simulator to normal working operation. After contacting Spirent Applications Support to report a fault, Applications Support staff will create a System Report (SR) and copy you with the SR reference number. Spirent Applications Support requests you use the SR reference number in all correspondence relating to that incident. All responses from Spirent in relation to your incident will contain the SR reference number.

When you first contact Spirent Applications Support to report a fault, please provide the following information:

- a) Your name
- b) Your e-mail address
- c) Your telephone or fax number
- d) The types and version numbers of your Spirent software
- e) The serial number of your signal generator (located on rear panel)
- f) A comprehensive description of the incident and, where appropriate, provide:
 - i) A copy of the *message_log.txt* file
 - ii) Copies of your scenario files, including all shared files. Provide a separate list, as a plain text (**.txt*) file, of all files you send Spirent.
 - iii) Details of the repeatability of the incident
 - iv) Details of changes to the system, including any new software added and all upgrades (including Windows, new hardware, new drivers, Spirent software and so on)

This will enable a swift response and improve the service to you.

Spirent Applications Support contact details:

The Americas (All Positioning Customers)

Spirent Federal Systems Inc.,
Applications Support Centre,
1331 Airport Freeway, Suite 304,
Eules,
TX 76040,
USA

Contact Applications Support:
Tel: +1 817 508 6095
Fax: +1 817 508 6096
E-mail: help@spirentfederal.com

Asia Pacific (All Positioning Customers)

Spirent Communications,
Shining Tower, No. 35, Xueyuan Road,
Room 1302,
Beijing 100083,
China

Contact Applications Support
Tel (Toll free mainland China): 4008109529
Tel (Outside mainland China): +86 4008109529
Fax: +86 1082330022
E-mail: help_asia@spirent.com

All other Regions and Positioning Customers

Spirent Communications Plc,
Aspen Way,
Paignton,
Devon
TQ4 7QR
United Kingdom

Contact Applications Support:
Tel: + 44 1803 546333
Fax: + 44 1803 546302
E-mail: help@spirent.com

Go to <http://support.spirent.com> for access to Spirent's Applications Support, FAQ and document databases.

Index and list of figures and tables

This page is intentionally blank

Index

1

1 CPS, **9-1**

1 PPS, 5-17, 5-18, 5-24, 5-35, 7-1, 7-2, 7-3, 7-4, 7-5, 7-6, 7-7, 7-8, 7-11, 7-12, 7-13, 7-14, 7-15, **9-1**, 10-2, 11-2, 11-6, 11-7, 12-9

A

AGP, **9-1**, 10-2, 10-3

Almanac, 5-3, 5-14, 5-21, 5-22, 13-4

Antenna gain pattern, 5-74

Antenna phase, 5-74

ASCII, 4-2, 5-4, 5-29, 5-30, 8-18, 8-43, **9-1**

B

BITE, 7-13

Broadcast, 4-5, 5-64, 8-3

C

Calibration, 5-6, 5-73

Closed-loop, 2-1, 3-1, 3-2, 7-3, 7-4, 10-5

csv, 8-43, 12-15

D

Datagram, 8-2, 8-3, 8-5, 8-6, 8-7, 8-8, 8-10, 8-13, 8-22, 8-23, 8-28, 8-29, 8-30, 8-34, 9-1

Delay, 5-4, 5-9, 5-29, 5-33, 5-83, 5-84, 6-4, 7-3, 7-6, 7-12, 7-13, 7-14, 8-2, 8-11, 8-13, 13-5

Dongle, 2-1, 8-23

DOP, 5-8, 5-81, 8-10

E

Earth Centred Earth Fixed, 5-6, 5-10, 5-67, 5-86, 6-1, 6-5, 8-8, 8-9, 8-13, 8-14, 8-17, 8-24, 8-25, 8-26, **9-1**, 13-1

ECI, 8-14, 8-17

EGNOS, 5-26, 5-44, 5-61, 8-12

Ellipsoid, 5-7, 5-79, 8-8, 8-10, 8-15, 8-17

Ethernet, 4-4, 4-5, 4-7, 7-3, 8-1, 8-28, 8-29, 8-43, 11-2, 11-10

G

Galileo, 6-2

GLONASS, 5-26, 5-60, 5-61, 5-62, 5-86, 6-2, 8-12

GPS, 5-4, 5-7, 5-10, 5-26, 5-29, 5-30, 5-31, 5-32, 5-38, 5-39, 5-40, 5-41, 5-53, 5-56, 5-61, 5-62, 5-66, 5-70, 5-74, 5-79, 5-83, 5-86, 6-4, 7-11, 7-12, 8-2, 8-12, 8-18, 9-1, 13-5, 13-6, 13-7

GSS7800, 6-2

GUI, **9-1**, 13-1

H

Heading, 5-7, 5-36, 5-67, 5-68, 5-69, 5-72, 5-78, 5-79, 5-80, 6-1, 8-17

I

IEEE-488, 1-1, 2-1, 2-2, 3-1, 4-3, 4-4, 5-1, 6-1, 7-3, 7-4, **9-1**, 10-1, 10-2, 10-3, 11-1, 11-2, 11-3, 11-10, 11-11

IEEE-488 card, 4-3, 10-1, 10-2, 10-3, 11-2

INS, 8-7

Integrity, 11-1

Ionosphere, 5-9, 5-84

L

L1, 5-55, 5-56, 13-3

L2, 5-55, 13-3

L2C, 8-11

L5, 8-11

LAAS, **9-1**

Latency, 3-1, 6-1, 6-4

LO, 13-3

M

Mean motion difference, 8-25

MHz, 5-96, 7-12

Model, 5-5, 5-17, 5-43, 5-68, 5-69, 8-7, 8-17, 12-17

Motion, 2-1, 3-1, 3-2, 3-3, 4-1, 4-4, 5-5, 5-6, 5-17, 5-33, 5-63, 5-65, 5-66, 5-67, 5-68, 5-69, 6-1, 6-2, 6-3, 6-5, 7-1, 7-3, 7-4, 7-5, 7-6, 7-7, 7-8, 8-4, 8-5, 8-10, 8-43, 10-2, 11-2, 11-3, 11-7, 11-8, 11-11, 12-8, 12-14, 12-16, 12-17, 13-1, 13-4

MSAS, 5-26, 5-44, 5-61, 8-12

MSL, 5-7, 5-79, 8-15

N

ns, 7-6, 7-11, 7-12

O

Obstacle, 5-8, 5-81

Offset, 4-4, 4-5, 5-4, 5-5, 5-6, 5-7, 5-29, 5-31, 5-32, 5-36, 5-40, 5-43, 5-54, 5-55, 5-61, 5-73, 5-74, 5-79, 8-2, 8-18, 12-8, 13-5, 13-6, 13-7

On, 5-24, 11-12, 13-1

P

PC, 2-1, 2-2, 2-3, 3-1, 3-2, 3-3, 4-1, 4-3, 4-4, 4-6, 5-54, 6-4, 7-3, 7-4, 7-5, 7-6, 7-8, 7-10, 7-15, 8-5, 9-1, 10-1, 10-2, 10-3, 11-1, 11-2, 11-3, 11-4, 11-7, 11-10, 11-11, 11-13, 12-1, 12-3, 12-6, 12-7, 12-8, 12-9, 12-16, 13-1

Power Level, 3-1, 4-1, 5-4, 5-37, 5-40, 5-74, 11-12, 12-16, 13-5

PRN, **9-1**, 11-6, 11-12, 12-9, 12-16

Pseudorange, 3-1, 5-5, 5-9, 5-53, 5-54, 5-61, 5-62, 5-83, 5-84, 5-85, 6-1, 6-2, 6-3, 6-4, 6-5, 8-11, 8-13, 13-3

PZ, 13-1

R

RS-232, 4-4, 5-1

S

SA, 13-4

SBAS, 5-10, 5-61, 5-86, 13-5, 13-6, 13-7

Scenario, 3-1, 3-2, 3-3, 4-1, 4-2, 4-3, 4-6, 4-7, 4-9, 5-1, 5-3, 5-4, 5-14, 5-15, 5-16, 5-17, 5-18, 5-19, 5-20, 5-29, 5-30, 5-31, 5-34, 5-36, 5-40, 5-54, 5-63, 7-1, 7-2, 7-3, 7-7, 7-8, 7-12, 8-1, 8-2, 8-4, 8-10, 8-18, 8-21, 8-23, 8-42, 8-43, 10-4, 11-2, 11-3, 11-4, 11-5, 11-6, 11-7, 11-8, 11-11, 11-13, 12-9, 12-10, 12-11, 12-12, 12-14, 12-15, 12-16, 13-2

SCRAMNet, 1-1, 2-1, 2-2, 3-1, 4-4, 4-5, 5-1, 6-1, 7-3, 12-1, 12-2, 12-3, 12-4, 12-6, 12-8, 12-15

SCRAMNet card, 4-4, 12-1, 12-2, 12-3, 12-4

Signal Level, 5-9, 5-41, 5-53, 5-73, 5-83, 5-84, 7-11

Signal Power, 5-4, 5-37, 5-39, 5-40, 5-41

SimREPLAY and SimREPLAYplus, 2-1

Simulation iteration rate, **3-1**

Sockets (and see TCP/IP), 2-2, 4-1

STANAG, 5-40

SVID, 5-4, 5-5, 5-9, 5-10, 5-37, 5-38, 5-39, 5-40, 5-43, 5-53, 5-55, 5-56, 5-61, 5-62, 5-82, 5-83, 5-84, 5-86, 5-88, 5-89, 5-90, 5-91, 8-12, 8-27, 8-28, 13-3, 13-4

Synchronisation, 2-1, 3-2, 4-1, 5-5, 5-54, 5-63, 5-65, 7-1, 7-3, 7-4, 7-8, 7-10, 7-12, 7-13, 10-1, 12-9

T

TCP/IP, 2-1, 2-2, 3-1, 4-4, 5-1, 6-1, 7-3, **9-1**, 11-10

Technical Support, 12-3

time of validity, 6-2, 6-3, 8-24, 8-30

Time Step, **3-1**

Timer card, 1-1, 2-1, 2-2, 5-4, 5-29, 5-34, 7-4, 7-5, 10-1, 10-2, 11-7, 12-9

TOW, 5-4, 5-29, 5-32

Troposphere, 5-9, 5-84

U

UDP/IP, 8-1, 8-2, 8-3, 8-5, 8-6, **9-1**

Unicast, 8-3

USB, 7-4, **9-1**, 10-3

UTC, 5-4, 5-29, 5-30, 5-31, 5-32, 8-2, 8-18

V

VMS, 4-1, 8-1, **9-1**, 13-1, 13-2, 13-4

W

WAAS, 5-26, 5-44, 5-61, 8-12

Warning, 5-12

Warranty, 13

Week Number, 5-4, 5-29, 5-32

WGS84, 5-6, 5-67, 6-1, 8-8, 8-10, 8-15, 8-17

X

XML, 4-3, 4-4, 5-13, 8-1, **9-1**, 12-9, 12-16

This page is intentionally blank

List of figures

Figure 2-1: SimREMOTE synchronisation and interface options.....	2-2
Figure 2-2: SimREMOTE - interconnection diagram.....	2-3
Figure 3-1: Typical closed-loop system configuration (such as a flight simulator).....	3-2
Figure 4-1: Remote Command Settings dialog	4-2
Figure 4-2: Socket string send utility dialog.....	4-6
Figure 4-3: TCP/IP address.....	4-6
Figure 4-4 Socket string send utility	4-7
Figure 4-5: Using Socket string send utility – Run	4-8
Figure 4-6: Using Socket string send utility – Null	4-8
Figure 4-7: Using Socket string send utility - End	4-9
Figure 5-1: External arm source message sequence.....	5-24
Figure 5-2 Vehicle Body Axes	5-71
Figure 5-3: Attitude orientation in relation to body axes	5-72
Figure 7-1: Alignment of the 1PPS signal, trigger disabled and delayed mode.....	7-2
Figure 7-2: Alignment of the 1PPS signal, immediate trigger mode	7-2
Figure 7-3: System interconnection - trigger Disabled	7-4
Figure 7-4: Software start-up sequence	7-5
Figure 7-5: System interconnection – Immediate trigger mode.....	7-6
Figure 7-6: System interconnections - Delayed trigger mode	7-7
Figure 7-7: Interconnections for two signal generators	7-9
Figure 7-8: Interconnections using a GSS4767 distribution unit.....	7-10
Figure 7-9: Timing requirements for 1PPS in and resultant start timing	7-14
Figure 7-10: Timing requirements for TRIG IN (immediate mode) and resulting start timing	7-14
Figure 7-11: Configuring a GSS6560 for remote control operation.....	7-15
Figure 7-12 Old style PCI Timer cable assembly	7-16
Figure 7-13 New style PCI Timer cable assembly	7-16
Figure 8-1: Data streaming definition file selection	8-1
Figure 8-2: Data streaming definition dialog.....	8-2
Figure 8-3: Data Streaming Definition - vehicle motion data dialog	8-4
Figure 8-4 Scenario Options to use Car motion sensor data	8-4
Figure 8-5: Data Streaming Definition - Antenna_n settings dialog	8-5
Figure 8-6: Ethernet client setup	8-6
Figure 8-7 Using Truth_Aiding with SimEGI.....	8-16
Figure 8-8: SimGEN Data Streaming Client.....	8-42
Figure 8-9: Data Streaming Message Logger Utility dialog.....	8-43
Figure 10-1: Set IEEE usage.....	10-4
Figure 10-2: Remote Command Settings dialog	10-4
Figure 11-1: Remote PC Utilities installation dialog	11-1
Figure 11-2: IEEE_488 test configuration.....	11-3
Figure 11-3: IEEE test program menu.....	11-3
Figure 11-4: Running a simulation with the IEEE test program	11-4
Figure 11-5: Changing the scenario	11-5
Figure 11-6: Changing the trigger mode.....	11-6
Figure 11-7: Changing the motion message rate	11-7
Figure 11-8: Testing the timer card	11-8
Figure 11-9: Remote motion demo.....	11-9
Figure 11-10: TCP/IP test configuration	11-10
Figure 11-11: Menu of remote motion software TCPIP.....	11-11
Figure 11-12: Changing the RF power level.....	11-12
Figure 11-13: Running the remote logging example program.....	11-14
Figure 12-1 SCRAMNet test configuration	12-2
Figure 12-2: SCRAMNet Windows installation dialog	12-3
Figure 12-3: SCRAMNet registry values	12-3
Figure 12-4: Node status window	12-4
Figure 12-5: WinDiags dialog	12-4
Figure 12-6: WinDiags Run Tests window	12-5
Figure 12-7: WinDiags Run Tests - tests completed window.....	12-5

Figure 12-8: SCRAMNet WinMon dialog.....	12-6
Figure 12-9: WinDiags communication test.....	12-7
Figure 12-10: Enable the SCRAMNet interface for SimGEN	12-8
Figure 12-11: Scramnet_std_remote utility.....	12-8
Figure 12-12: Get SimGEN scenario status	12-9
Figure 12-13: Entering a new scenario name	12-10
Figure 12-14: Sending a run command to SimGEN.....	12-11
Figure 12-15: Stopping SimGEN	12-12
Figure 12-16: Rewinding the SimGEN scenario.....	12-13
Figure 12-17: Running with motion.....	12-14
Figure 12-18: Enable data streaming	12-15
Figure 12-19: Copy to SCRAMNet	12-15
Figure 12-20: Logging truth data	12-16

List of tables

Table 1-1: Document format conventions.....	1-2
Table 4-1: Simulation status bit description.....	4-3
Table 4-2: Memory map for default SCRAMNet protocol.....	4-5
Table 5-1 Index of remote commands.....	5-1
Table 5-2 Remote command syntax.....	5-2
Table 5-3 Scenario commands.....	5-3
Table 5-4 Time commands.....	5-4
Table 5-5 Antenna Pattern commands.....	5-4
Table 5-6 Signal Power commands.....	5-4
Table 5-7 Signal Control commands	5-5
Table 5-8 Pseudorange commands	5-5
Table 5-9 Data Streaming commands.....	5-5
Table 5-10 Motion commands	5-6
Table 5-11 Hardware and Calibration commands.....	5-6
Table 5-12 Vehicle Data Request commands.....	5-7
Table 5-13 Antenna Data Request commands	5-8
Table 5-14 Signal Data Request commands.....	5-9
Table 5-15 Transmitter Data Request commands	5-10
Table 5-16 Navigation commands.....	5-10
Table 5-17 Navigation Data related commands	5-10
Table 5-18 Interferer commands	5-11
Table 5-19: 6560/5060 timer command and 47xx/7xxx equivalents	5-35
Table 5-20 Choosing between Channel and SVID.....	5-37
Table 5-21 Combining <absolute> and <align> when <all_flag>=1	5-40
Table 5-22 Choosing between Channel and SVID.....	5-43
Table 5-23 Definition of aircraft body axes.....	5-71
Table 5-24 Definition of body axis angles.....	5-72
Table 5-25 Signal Data Request - <type> and <freq> parameters	5-82
Table 5-26 Limits for <centre freq> of FM interfering signal.....	5-96
Table 5-27: UDP Command structure	5-102
Table 5-28: UDP commands for MOT	5-102
Table 5-29 C++ structure for MOT command	5-103
Table 5-30: UDP commands for MOTB.....	5-104
Table 5-31 C++ structure for MOTB command	5-104
Table 5-32: UDP commands for MOD.....	5-107
Table 5-33 C++ structure for MOTB command	5-107
Table 7-1 1PPS signal reference frequencies.....	7-1
Table 7-2 1PPS modes.....	7-1
Table 7-3 Supported trigger modes.....	7-2
Table 7-4: Signal generator I/O port characteristics.....	7-11
Table 7-5: Available timer settings	7-12
Table 7-6: Timer cable assembly - connector details.....	7-15

Table 8-1 UDP data packet for signal type.....	8-30
Table 11-1: SimGEN status responses	11-6
Table 12-1: Scenario status responses	12-9
Table 13-1: Comparison of remote commands used in SimGEN for Windows and SimGEN for VMS	13-1

This page is intentionally blank

Spirent Communications Plc
Aspen Way
Paignton
Devon TQ4 7QR

So far as Spirent Communications Plc is aware the contents of this document are correct. However, such contents have been obtained from a variety of sources and can give Spirent Communications Plc no warranty or undertaking and make no representation as to their accuracy. In particular, Spirent Communications Plc hereby expressly excludes liability for any form of consequential, indirect or special loss, and for loss of data, loss of profits or loss of business opportunity, howsoever arising and whether sustained by the user of the information herein or any third party arising out of the contents of this document.

SimREMOTE User Manual and ICD

Software for the Spirent Range of Satellite Navigation Simulation Products

Copyright © 2002 - 2010 Spirent Communications Plc

The copyright of this publication is the property of Spirent Communications Plc. Without the written consent of Spirent Communications Plc, given by contract or otherwise, this document must not be copied, reprinted or reproduced in any material form, either wholly or in part, and the contents of this document, or any methods or techniques available therefrom, must not be disclosed to any other person whatsoever.

SPIRENT COMMUNICATIONS PLC CONFIDENTIAL:

The information contained in this document is the property of Spirent Communications Plc. Except as specifically authorised in writing by Spirent Communications Plc, the holder of this document shall keep the information contained herein confidential and shall protect same in whole or in part from disclosure and dissemination to third parties and use same for evaluation, operation, and maintenance purposes only.

Document number: DGP00792AAA

Document Issue: 2-16

Date: March 2010

Printed in England