Space Flight Technology, German Space Operations Center (GSOC)

Deutsches Zentrum für Luft- und Raumfahrt (DLR) e.V.

GSOC Multi-Satellite Simulator

J.-S. Ardaens, S. D'Amico, G. Gaias

Doc. No. : TN 09-01

Version : 1.3

Date : 09-Jan-2012

Document Change Record

Issue	Date	Pages	Description of Change
0.1	Feb. 18, 2009	all	Document structure
0.2	Feb. 26, 2009	all	Finalized system description and overview
0.3	Sep. 2010	all	Rewritten the section System Overview
0.4	Sep. 2010	all	Finalized the section System Overview
0.5	Feb 2011	all	Added the section Getting Started

Table of Contents

Document Change Record ..... **ii**

Table of Contents ..... **iii**

Scope ..... **ix**

Acronyms and Abbreviations ..... **x**

1. Motivations **1**

2. System Overview **2**

2.1 Description 2

2.2 Rapid Prototyping 2

2.3 Software Management 3

2.4 Hardware-in-the-Loop Capabilities 3

2.5 Validation and Testing 4

2.6 Application Examples 4

2.6.1 Closed-Loop GPS-Based GNC System for Spacecraft Formation Flying 4

2.6.2 Hardware-in-the-Loop, Closed-Loop GNC System for Rendezvous and Docking 5

2.6.3 Hardware-in-the-loop, Closed-loop GPS-based GNC System for Dual Spacecraft Formation Flying 6

2.6.4 Operational Support of Flight Software 7

3. Getting Started **9**

3.1 Software Prerequisite 9

3.2 Installation Using the MSS Installer 9

3.3 Advanced Manual Installation using Subversion 10

3.3.1 Software download 10

3.3.2 Registration of the libraries and dependencies 10

3.3.3 Registration within the MATLAB® environment 10

3.3.4 Compilation of the Simulink® S-Functions and tools 11

3.4 Overview of the Multi-Satellite Simulator 12

4. Description of the MATLAB® Toolbox **13**

4.1 Overview 13

4.2 Available MATLAB® scripts and functions 13

4.3 Available GHOST C++ objects and methods 13

5. Description of the Simulink® Libraries **15**

5.1 Conversion 15

5.1.1 Cartesian to Spherical 15

5.1.1.1 Description 15

5.1.1.2 Inputs and Outputs 15

5.1.2 Coordinate Converter 16
5.1.2.1 Description 16
5.1.2.2 Parameters and Dialog Box 16
5.1.2.3 Inputs and Outputs 16
5.1.3 Euler Angles for SimGEN 17
5.1.3.1 Description 17
5.1.3.2 Inputs and Outputs 17
5.1.4 GPStime Add 18
5.1.4.1 Description 18
5.1.4.2 Parameters and Dialog Box 18
5.1.4.3 Inputs and Outputs 18
5.1.5 GPStime Interface 19
5.1.5.1 Description 19
5.1.5.2 Parameters and Dialog Box 19
5.1.5.3 Inputs and Outputs 19
5.1.6 Mean to Osculating 21
5.1.6.1 Description 21
5.1.6.2 Parameters and Dialog Box 21
5.1.6.3 Inputs and Outputs 21
5.1.7 Relative Elements 22
5.1.7.1 Description 22
5.1.7.2 Inputs and Outputs 22
5.1.8 State Converter 24
5.1.8.1 Description 24
5.1.8.2 Inputs and Outputs 24
5.2 Environment 25
5.2.1 Atmospheric Density 25
5.2.1.1 Description 25
5.2.1.2 Parameters and Dialog Box 25
5.2.1.3 Inputs and Outputs 25
5.2.2 Reference Systems 26
5.2.2.1 Description 26
5.2.2.2 Parameters and Dialog Box 26
5.2.2.3 Inputs and Outputs 26
5.2.3 Satellite Acceleration 28
5.2.3.1 Description 28
5.2.3.2 Parameters and Dialog Box 28
5.2.3.3 Inputs and Outputs 29
5.2.4 Satellite Torque 30
5.2.4.1 Description 30

5.2.4.2 Parameters and Dialog Box 30
5.2.4.3 Inputs and Outputs 30
5.3 Mathematics 32
5.3.1 Cross 32
5.3.1.1 Description 32
5.3.1.2 Inputs and Outputs 32
5.3.2 Cubic Interpolation 33
5.3.2.1 Description 33
5.3.2.2 Inputs and Outputs 33
5.3.3 Matrix to Quaternion 34
5.3.3.1 Description 34
5.3.3.2 Inputs and Outputs 34
5.3.4 Multiply Quaternion 35
5.3.4.1 Description 35
5.3.4.2 Inputs and Outputs 35
5.3.5 Norm 36
5.3.5.1 Description 36
5.3.5.2 Inputs and Outputs 36
5.3.6 Normalize Quaternion 37
5.3.6.1 Description 37
5.3.6.2 Inputs and Outputs 37
5.3.7 RTN Matrix 38
5.3.7.1 Description 38
5.3.7.2 Inputs and Outputs 38
5.3.8 Rotation Matrix 39
5.3.8.1 Description 39
5.3.8.2 Inputs and Outputs 39
5.3.9 Slerp 40
5.3.9.1 Description 40
5.3.9.2 Parameters and Dialog Box 40
5.3.9.3 Inputs and Outputs 40
5.3.10 Transpose Quaternion 41
5.3.10.1 Description 41
5.3.10.2 Inputs and Outputs 41
5.4 System & Interfaces 42
5.4.1 Add Command 42
5.4.1.1 Description 42
5.4.1.2 Parameters and Dialog Box 42
5.4.1.3 Inputs and Outputs 42
5.4.2 Modular Interface: Client 43

5.4.2.1 Description 43
5.4.2.2 Parameters and Dialog Box 43
5.4.3 Navigation Reader 44
5.4.3.1 Description 44
5.4.3.2 Parameters and Dialog Box 44
5.4.3.3 Inputs and Outputs 44
5.4.4 POD Product Reader 45
5.4.4.1 Description 45
5.4.4.2 Parameters and Dialog Box 45
5.4.4.3 Inputs and Outputs 45
5.4.5 Phoenix Command Generator 47
5.4.5.1 Description 47
5.4.5.2 Parameters and Dialog Box 47
5.4.5.3 Inputs and Outputs 47
5.4.6 Phoenix Data Retriever 48
5.4.6.1 Description 48
5.4.6.2 Parameters and Dialog Box 48
5.4.6.3 Inputs and Outputs 48
5.4.7 Phoenix Message Logging 50
5.4.7.1 Description 50
5.4.7.2 Parameters and Dialog Box 50
5.4.7.3 Inputs and Outputs 50
5.4.8 SimGEN Interface 51
5.4.8.1 Description 51
5.4.8.2 Parameters and Dialog Box 51
5.4.8.3 Inputs and Outputs 52
5.4.9 Time Synchronization 53
5.4.9.1 Description 53
5.4.9.2 Parameters and Dialog Box 53
5.4.9.3 Inputs and Outputs 53
5.4.10 Winmon Message Reader 54
5.4.10.1 Description 54
5.4.10.2 Parameters and Dialog Box 54
5.4.10.3 Inputs and Outputs 54
5.4.11 toFile 55
5.4.11.1 Description 55
5.4.11.2 Parameters and Dialog Box 55
5.4.11.3 Inputs and Outputs 55
5.4.12 toFileTCP Client 56
5.4.12.1 Description 56

5.4.12.2 Parameters and Dialog Box 56
5.4.12.3 Inputs and Outputs 56
5.5 Sensors & Actuators 57
5.5.1 Attitude Errors 57
5.5.1.1 Description 57
5.5.1.2 Inputs and Outputs 57
5.5.2 Phoenix Receiver Emulator 58
5.5.2.1 Description 58
5.5.2.2 Parameters and Dialog Box 58
5.5.2.3 Inputs and Outputs 58
5.5.3 Thruster Model 60
5.5.3.1 Description 60
5.5.3.2 Parameters and Dialog Box 60
5.5.3.3 Inputs and Outputs 60
5.6 Advanced Functions 61
5.6.1 Attitude Dynamics 61
5.6.1.1 Description 61
5.6.1.2 Parameters and Dialog Box 61
5.6.1.3 Inputs and Outputs 61
5.6.2 Frames Setting 64
5.6.2.1 Description 64
5.6.2.2 Parameters and Dialog Box 64
5.6.2.3 Inputs and Outputs 64
5.6.3 Orbit Dynamics 66
5.6.3.1 Description 66
5.6.3.2 Parameters and Dialog Box 66
5.6.3.3 Inputs and Outputs 67
5.6.4 SGP4 Propagator 68
5.6.4.1 Description 68
5.6.4.2 Parameters and Dialog Box 68
5.6.4.3 Inputs and Outputs 69
5.7 Vehicles 70
5.7.1 Minimalistic Spacecraft 70
5.7.1.1 Description 70
5.7.1.2 Parameters and Dialog Box 70
5.7.1.3 Inputs and Outputs 71
5.7.2 Spacecraft 73
5.7.2.1 Description 73
5.7.2.2 Parameters and Dialog Box 73
5.7.2.3 Inputs and Outputs 74

Activities Conducted with the Multi-Satellite Simulator 77

References 78

Scope

The research and development activities on autonomous onboard navigation and control performed in the framework of the Prisma and TanDEM-X missions kicked off in 2005 the development of highly realistic simulation models for formation flying. It has been subsequently recognized that the growing role of multi-satellite missions in future space applications raises the need of reliable, powerful and flexible multi-spacecraft simulation environments for the development of complex flight software. The Multi-Satellite Simulator described in this document stems from the desire of capitalizing on the achievements and experience collected during these pilot missions to create a harmonized simulation framework able to support a large variety of multi-satellite projects.

Acronyms and Abbreviations

EPOS 2.0	European Proximity Operations Simulator
GCC	GNU Compiler Collection
GHOST	GPS High Precision Orbit Determination Software Tools
GNC	Guidance, Navigation and Control
GPS	Global Positioning System
GSOC	German Space Operations Center
RTEMS	Real-Time Executive for Multiprocessor Systems
SPARC	Scalable Processor ARChitecture
SVN	Apache Subversion
SW	Software
TCP	Transmission Control Protocol

1. Motivations

Precise and robust guidance, navigation and control (GNC) of spacecraft are fundamental for successful multi-satellite systems. When designing a GNC system, key aspects like performance, robustness, integration within the space segment or operational utilization need to be analyzed already during the preliminary design and consolidated throughout all the project phases. These analyses require dedicated simulation and validation tools. The scope of functionalities provided by these tools can be as broad as the quick insight into the behavior of the algorithms, the validation of interfaces, the realistic assessment of performance, the inclusion of specific hardware components in the loop, or the analysis of usage of resources.

The Multi-Satellite Simulator described in this document aims at answering to this need by providing a complete framework for the design, implementation, rapid prototyping, testing and validation of complex GNC systems. The objective is to provide a modular and flexible development environment used for every kind of formation flying projects and research activities, no matter how mature they are and which objective they follow. Thanks to this philosophy, all projects benefit from the development and heritage coming from the other projects based on the same framework and, as a direct consequence, inherit a validated and powerful simulation environment as well as a collection of precious analysis tools.

The main challenge is to provide a collection of generic and validated models, functionalities and tools which can interoperate together and can be easily adapted to any mission and, at the same time, to always grant the possibility of mission-specific additional development. This challenge has been solved by implementing the Multi-Satellite Simulator under the Simulink® [1] environment, widely spread among the GNC community, which offers the advantage of modularity, flexibility, and easy porting to other platforms.

The Multi-Satellite Simulator benefits from the heritage of the Prisma [2] and TanDEM-X [3] missions, both pioneer missions in the field of autonomous formation flying. It takes advantages of numerous models and tools developed during the different phases of the project and makes intensive use of the existing models provided by the DLR/GSOC's C++ GHOST library [4]. Special effort has been made to collect the experience gained during these projects and to harmonize the tools and models, strengthening the overall reliability and user-friendliness of the testbed.

2. System Overview

2.1 Description

The Multi-Satellite Simulator is a mission-independent framework. As such it is designed to offer a high flexibility in order to be utilizable for a large variety of mission profiles and research topics. Focus has been given to the simplicity and transparency of utilization. To that end the complete project is implemented using Simulink®, which offers a user-friendly interface, a high modularity and is usually well-known among the aerospace community. Furthermore it makes the complete system easily upgradeable and extendable.

The Multi-Satellite Simulator is composed of Simulink® libraries offering ready-to-use functionalities and models, which allows the quick and easy creation of highly sophisticated simulations. The Simulink® [1] blocks composing the libraries are implemented using either C++ based S-functions or MATLAB® [5] Embedded functions. This choice enables the possibility for automatic generation of C/C++ code, which is of great interest when porting the models to other platforms.

Overall the Simulink® blocks may be divided into seven categories:

- **Environment:** models describing the environment acting on a spacecraft.
- **Conversion**: functions for time and frame transformation.
- **Mathematics**: implementation of specific operations which are not provided in the standard Simulink® libraries, like the quaternion algebra.
- **System and Interfaces**: components which ease the handling of data and the interoperability with other systems or simulations environments.
- **Sensors and Actuators**: models of spacecraft sensors and actuators.
- **Onboard Functionalities**: useful existing pieces of flight software coming from previous projects or newly developed.
- **Advanced Functions**: high-level functions using the aforementioned libraries to create complex, ready-to-use functionalities.
- **Vehicles**: high-level models of spacecraft, built from the low-level models comprised in the libraries described above.

In addition, the Multi-Satellite Simulator offers a collection of tools, MATLAB® scripts or programs written in C++, whose purpose is to facilitate the preprocessing and treatment of data.

2.2 Rapid Prototyping

The main driver underlying the design of the Multi-Satellite Simulator is the ability to conduct in short time scales realistic and meaningful analyses of a given problem. This is made possible by the strategy of having a collection of ready-to-use models and functionalities which are already validated. Another important aspect is the ability to port rapidly and to run the algorithms on dedicated target systems. The goal of rapid prototyping is not to provide flight versions of GNC algorithms, but to get quickly relevant information concerning the ability of the prototype to fulfill mission requirements and to evaluate the resources needed, in order to take the proper design decisions at the early stages of the development. To that end, special attention has been paid to ensure the compatibility of the libraries with Real-Time Workshop® [6], the automatic code generation system of MATLAB®, which allows the instantane-

ous porting of existing Simulink® models to other platforms. The operating systems for true real-time applications VxWorks® [7] and RTEMS [8] are currently supported.

2.3 Software Management

The high complexity and the continuous improvements of the Multi-Satellite Simulator require a rigorous software management. For this reason, the project is handled by a version control system. In addition to the obvious advantages brought by a version control system, like the easy tracking of individual contributions or the access to the complete history of changes for every file, this strategy offers the capability of keeping different versions of the Multi-Satellite Simulator at the same time. This is particularly of interest for Simulink® blocks, whose interfaces are inclined to change very often. The version control system allows the user to freeze the status of the libraries at a given time and to refer to one particular version when creating a new simulation based on these libraries. This ensures the continued availability und functioning of short term studies or past projects over years, even if their development has been stopped a long time ago. A graphical example is shown on Fig. 1. The past project is a Simulink® model whose development is definitively stopped. In-between the libraries might have evolved. In order to avoid any compatibility breaking, the status of the libraries at the epoch of the model development has been frozen and the project is simply pointing to these versions. The current project instead uses the most recent (also most advanced) version of the libraries, while the recycled project reuses an old project and improves only part of it by using the newest version of one library.

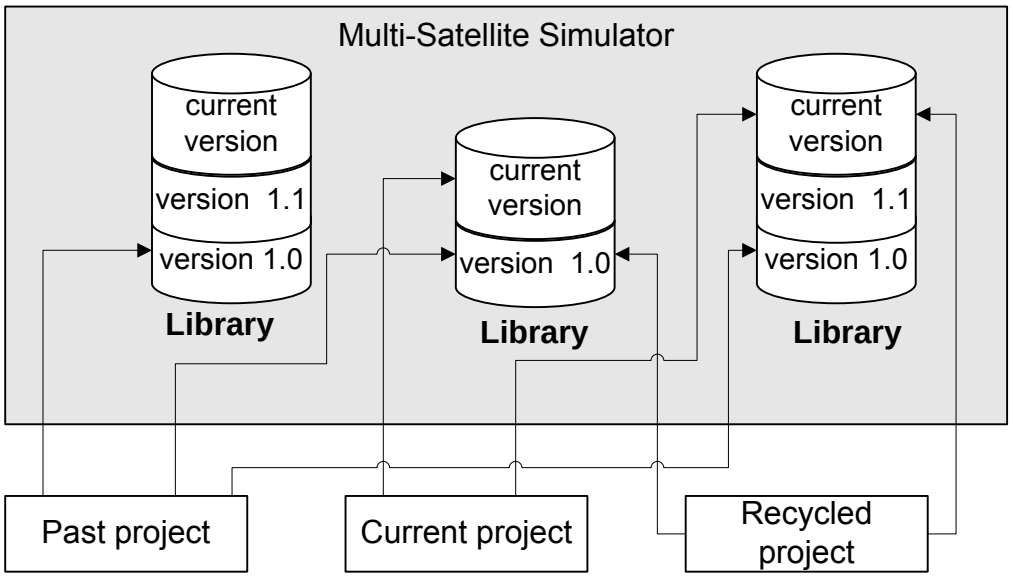


Fig. 1: Example of software management

2.4 Hardware-in-the-Loop Capabilities

A real-time hardware-in-the-loop, closed-loop testbed is of relevance for advanced functional and performance tests. The Multi-Satellite Simulator comprises special system components to enable the utilization of some hardware components in the loop. Currently the following hardware is supported:

- **Phoenix GPS receivers** [9]. The use of up to two single frequency GPS receivers in the loop is done by steering in real-time a GPS signal simulator (GSS) and connect-

ing directly the GPS receivers to the outputs of the simulator. This ensures that the flight software is able to interface properly the sensors and able to handle the failures that may occur. In addition, ultimate assessment of navigation, guidance and control performances is achieved by avoiding any software model of the sensor.

- **LEON 3 board with RTEMS operating system.** The introduction of a target computer representative of the onboard computer on which the flight software will be running is of great help for the analysis of software resource usage. This is achieved by making use of the MATLAB ® automatic code generation capabilities applied to the algorithms developed under Simulink® to generate C++ code and compile it for the target computer.
- **EPOS 2.0 facility.** The Multi-Satellite Simulator has the ability to interoperate with the DLR's European Docking and Rendezvous Simulation Facility (EPOS 2.0) [12] to enable the demonstration of algorithms for real-time rendezvous and docking.

The inclusion of hardware components in the loop introduces special limitations and issues, such as the mandatory time synchronization between all the systems and the communication between components. Dedicated system components have been developed to enable the synchronicity and interoperability between all the subsystems.

2.5 Validation and Testing

The constantly growing complexity of the Multi-Satellite Simulator makes a rigorous validation and testing strategy mandatory. This is achieved on different levels. Unit testing applied to the elementary blocks composing the libraries guarantees that the single blocks execute properly their tasks and prevents them from any future software regression. System testing is instead intended to verify the Multi-Satellite Simulator in able to fulfill system requirements related to the functionalities, performance, interfaces and resource usage. A collection of unit tests is associated to the libraries to ensure at any time the proper functioning of the testbed. In order to reduce the testing efforts, the unit and system testing is fully automated. The ultimate validation of the testbed is achieved when comparing simulation results and flight data. Flight data coming from the Prisma [2] and from the TanDEM-X [3] mission, both launched in June 2010, have been used to verify the overall performance of the testbed [10].

2.6 Application Examples

Thanks to its modularity and flexibility, the Multi-Satellite Simulator is able to support a large variety of applications. In order to help the reader getting an idea of the potential offered by the testbed, typical utilizations are summarized in the sequel.

2.6.1 Closed-Loop GPS-Based GNC System for Spacecraft Formation Flying

The design, implementation and test of GNC algorithms in a pure Simulink® environment represent most of the applications of the Multi-Satellite Simulator. Fig. 2 depicts an example of such a project. In the selected demonstration case, the user aims at developing a GNC algorithm to control actively the relative position of two spacecraft flying on a high elliptical orbit. More precisely, the goal is to control the relative position when GPS data are available, i.e. at the perigee passage. The goal is to set up a realistic simulation environment in which the control functionality (blocks painted in cyan on Fig. 2) is implemented and tested. For this purpose, it is necessary to simulate the behavior of GPS receivers flying on two spacecraft, to derive the navigation solution based on the GPS observations, to compute the control action based on this navigation solution, and to include this control action in the propagation of

the satellites. In order to speed up the development, the user makes intensively use of the models and functionalities available in the Multi-Satellite Simulator.

- The orbit and attitude of the both spacecraft are propagated using a generic, fully parameterizable model of the spacecraft (yellow blocks) coming from the **Vehicles** library. This block is set to control automatically the attitude in any desired attitude mode (in order to ensure an optimal tracking of the GPS signal). The block takes as input a control action which is then used when propagating the spacecraft.
- The GPS receivers are emulated using already available software models of the GPS receivers (blue block). The models for GPS receivers and orbit and attitude propagations belong both to the library for **Sensors and Actuators**.
- In the current show case, the user is more interested in the control part of the problem. As a consequence, he prefers to reuse existing software for precise relative navigation based on GPS measurements, and makes use of the navigation filter developed in the framework of the Prisma mission (orange block). This block belongs to the library for **Onboard Functionalities**.
- Small functionalities (magenta blocks) are in addition needed to ease the interfaces and extend the capabilities of Simulink®. The small blocks implement the handling of the time using the GPS time format, while the big magenta block on the right is an improved version of the standard Simulink® block used to save data in a file. These magenta blocks belong to the library for **System and Interface**.
- Finally, the green block is used to implement the conversion of direction cosine matrix to the quaternion representation. This block belongs to the library for **Mathematics**.

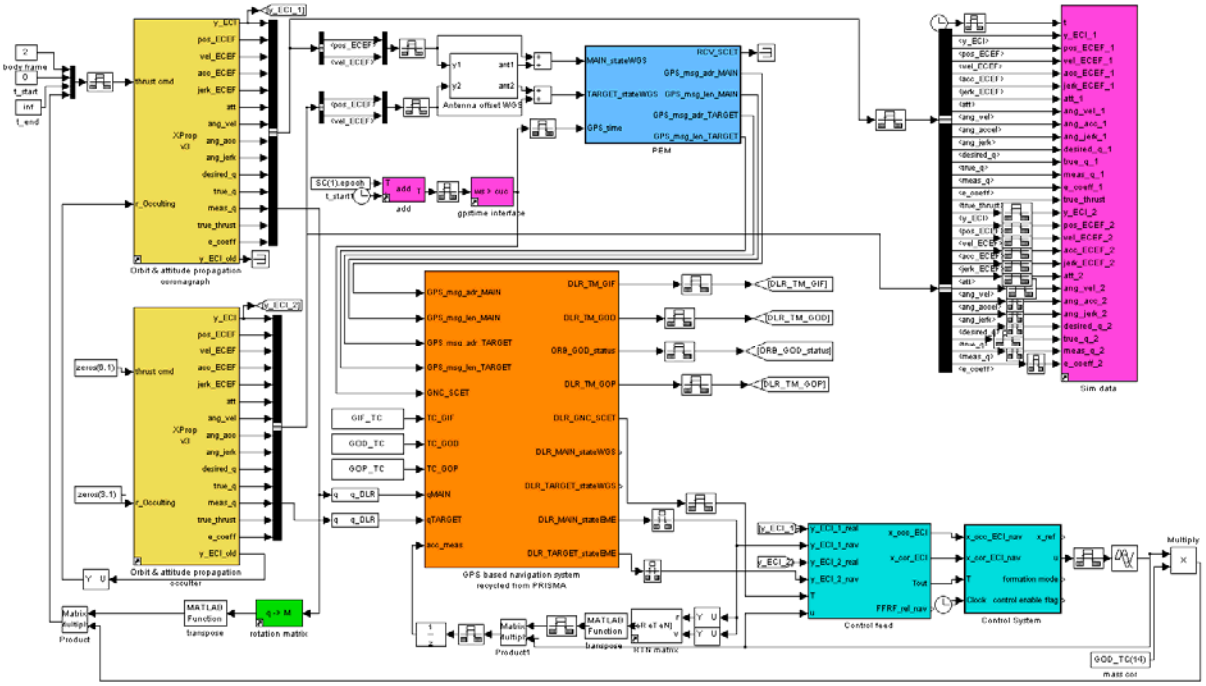


Fig. 2: Example of simulation based on the Multi-Satellite Simulator

2.6.2 Hardware-in-the-Loop, Closed-Loop GNC System for Rendezvous and Docking

The fact that the Formation Flying Testbed is implemented in Simulink® offers the possibility to port easily the models to other platforms. In the following example, a GNC system for Rendezvous and Docking [11] is tested using the EPOS 2.0 [12] facility (Fig. 3). For this pur-

pose, a complete simulation environment comprising the propagation of the dynamics, the GNC algorithms and the models of the actuator is first developed with Simulink®, as described in the previous section. Once the development is achieved, a block for interfacing the EPOS facility is added to the model and the complete model is translated into C/C++ and compiled for VxWorks® using the MATLAB ® Real-time Workshop and a GNU cross compiler. The model can then be run in real-time and steer the displacement of robots on which true hardware sensors are mounted. The measurements of these sensors are finally fed back to the GNC system running in the VxWorks® environment.

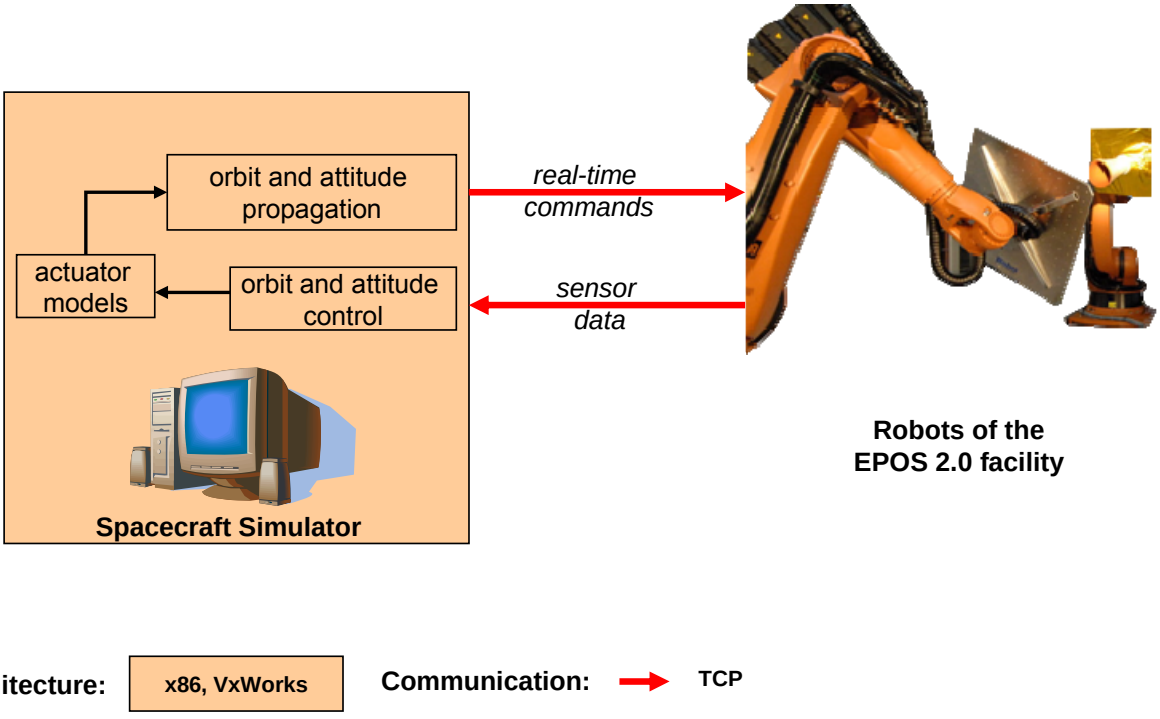


Fig. 3: Hardware-in-the-loop, closed-loop GPS-based simulation environment

2.6.3 Hardware-in-the-loop, Closed-loop GPS-based GNC System for Dual Spacecraft Formation Flying

At a certain phase of development, the user might be interested in using true GPS receivers in the loop instead of software emulations. The Multi-Satellite Simulator comprises several functions for the creation of GPS-based hardware-in-loop simulations. Among them, the block for steering a Spirent® GPS signal simulator from a Simulink® model s a key element which allows interfacing complex simulation environments with GPS receivers. In this case the Multi-Satellite Simulator is not anymore comprised in a single software model but becomes a facility composed of several units, which need to work together. Other functionalities are implemented to facilitate the exchange of information through a network or to synchronize the different units. Fig 3. depicts a possible configuration, in which not only the GPS receivers are used in the loop , but also target computers representative the onboard computer. In this example, the spacecraft simulator is implemented using a Simulink® model. Motion commands are sent in real-time to the computer steering the GPS Signal Simulator. This functionality as well as the mutual synchronization is ensured by a block for steering the GPS signal simulator. The GPS receivers process the radio-frequency signals coming from the GPS signal simulator and send the GPS observations to the onboard computers, on which the flight software is running. The control actions sent by the flight software are finally sent back to the spacecraft emulator, which uses advanced models of the dynamics and the actuator to propagate the transational and rotational motion. It is as well possible to emulate

the sensors which are not physically in the loop in the spacecraft simulator and send the resulting simulated measurements to the flight software.

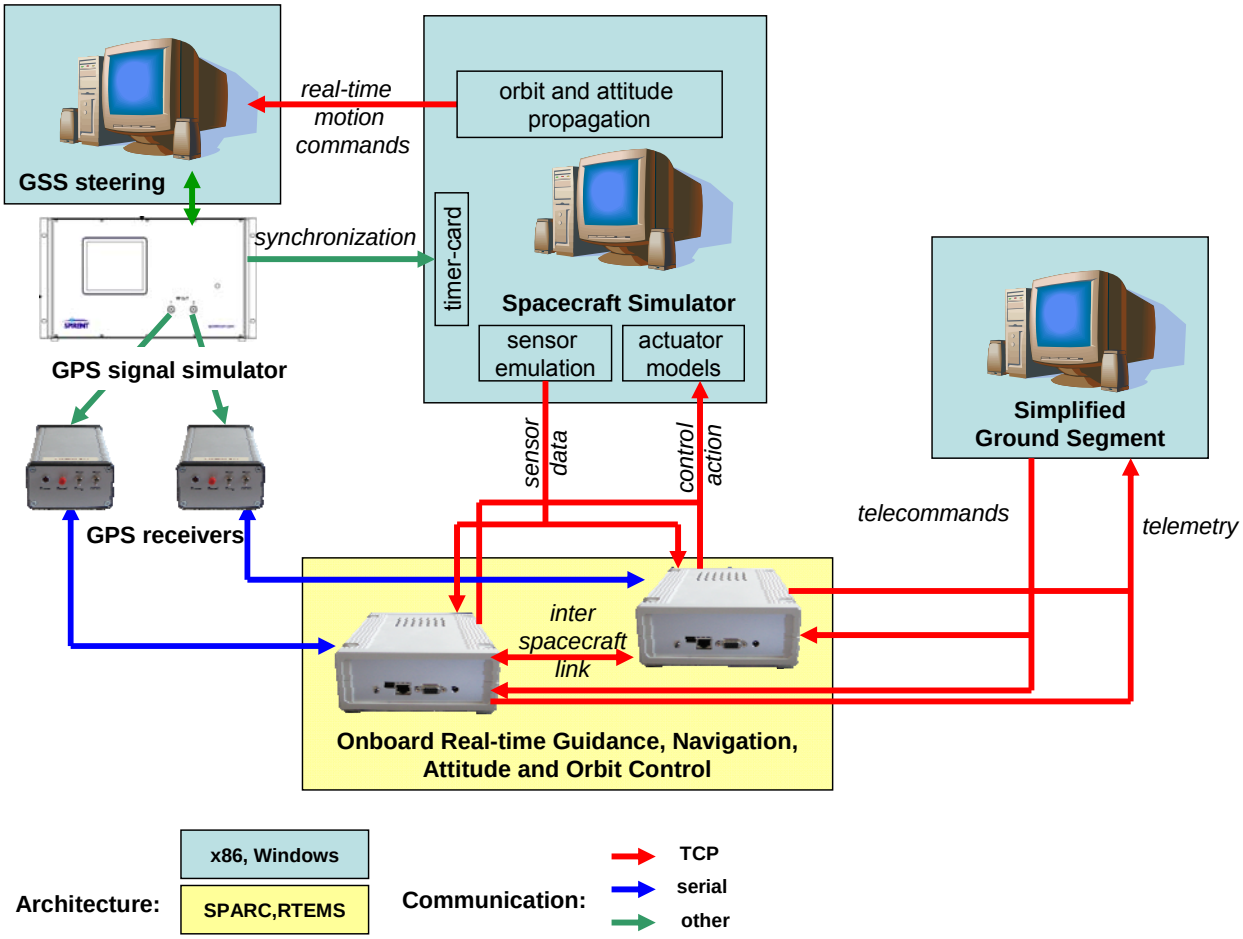


Fig. 4: Hardware-in-the-loop, closed-loop GPS-based simulation environment

The choice of TCP as communication protocol between the system components allows the possibility to create easily additional applications to monitor and control the system. The Simplified Ground Segment depicted on the Fig. 4 is for example intended to offer to the user a simple way to interact in real-time with the system, i.e. to send telecommands and receive telemetries, and thus to address some operational aspects when testing the flight software.

2.6.4 Operational Support of Flight Software

The efforts done to harmonize the components and tools as well as the flexibility offered by the testbed make very easy the creation of applications focused on dedicated purposes but sharing the same technical background. The Replay Tools developed for the post facto analysis of the DLR's contribution to the onboard software of the Prisma and TanDEM-X missions is a good illustration for this concept. Based on the simulation environment used to develop and validate the flight software, the Replay Tool replaces simply the simulation part (orbit and attitude propagation associated with models for the sensors and actuators) with a Simulink® block for reading real flight data. As depicted on Fig. 5, the other blocks remain identical, which allows the rapid replay of the flight software for offline comparison. In this case, the flight software is fed with the same inputs as it was during the flight and it is checked that the off-line behavior is identical with the in-flight one. Such a tool allows as well

the tuning of the software, whose goal is to analyze how would have behaved the flight software in the same conditions but with different settings.

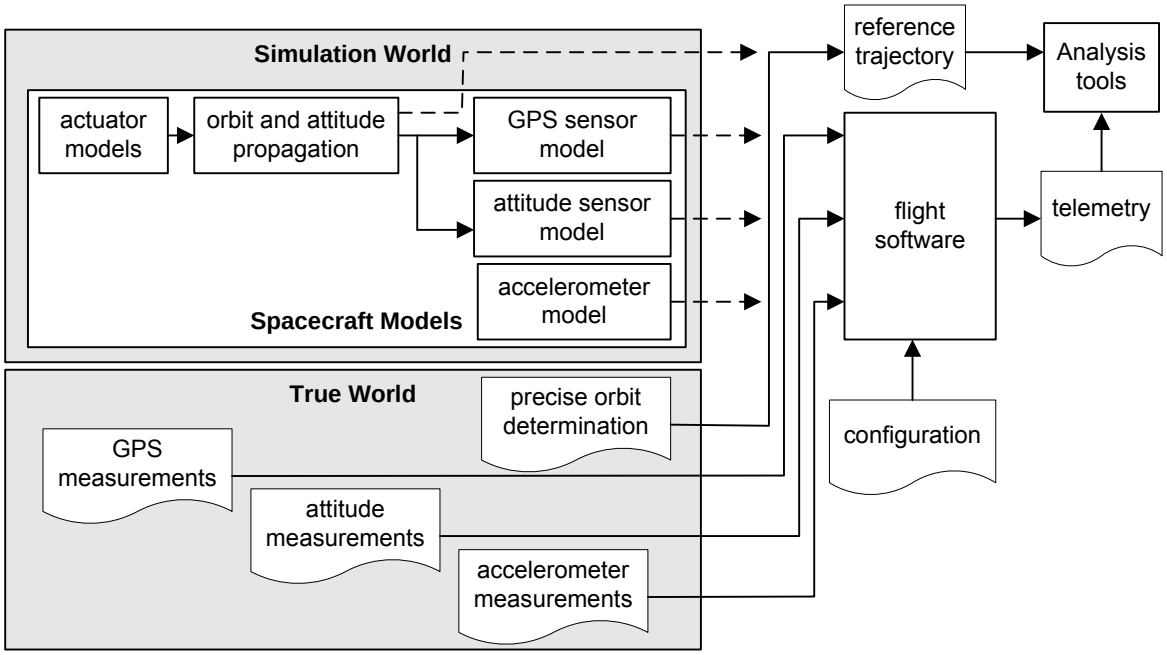


Fig. 5: Schematic view of the Replay Tool

3. Getting Started

3.1 Software Prerequisite

The Multi-Satellite Simulator requires MATLAB /Simulink® (version 2007b or above) installed on the workstation. MSS can be installed in two different ways. The simplest approach is to use the MSS installer, which offers a ready to use solution by installing automatically a complete release on the local drive (Section 3.2). In case the user wants to use instead the latest available developments, MSS is available as a subversion (SVN) repository (Section 3.3). In this case, in addition to the SVN client required to download and update the sources, a C++ compiler must be available and registered in the MATLAB/Simulink® environment to compile the S-Functions. The following compilers are currently supported:

- Microsoft Visual C++ .NET 2003
- GCC v. 4.3

3.2 Installation Using the MSS Installer

An installation package is available to install automatically MSS on a workstation. The MSS installer copies all the required files and performs automatically all the necessary actions to get a system immediately running. Only the relevant files will be installed, composing the ***light*** version of the MSS (cf. Fig 10). This approach is recommended for most of the users who do not intend to contribute actively as developer/tester.

Fig. 9: Installation using the MSS installer

3.3 Advanced Manual Installation using Subversion

This section is intended for advanced users or developers aiming at always using up-to-date versions of the Multi-Spacecraft Simulator and at contributing to the project. In this case, installing MSS requires the following steps:

3.3.1 Software download

The complete software can be downloaded using SVN at the following address:

`http://10.61.9.62/svn/MSS`.

A user account is required to access the repository (standard or developer) and can be obtained from the administrator of the repository.

3.3.2 Registration of the libraries and dependencies

The path to the MSS directory must be registered in the local system. This must be done by defining the string value `MSS_path` located in the registry key:

```
HKEY_CURRENT_USER\Software\GSOC
```

The variable contains the path to the MSS directory installed on the local workstation, eg:

```
MSS_path = 'C:\Program Files\MSS'
```

Writing values in the registry can be done using the Microsoft Registry Editor (regedit) or using a dedicated graphical:

```
[MSS_path]\trunk\gui\SetupPath.m
```

which lists all the values already written in the aforementioned registry key and allows the introduction of additional values. In addition, the GHOST Component Object Models need to be registered using the command:

```
regsvr32 [MSS_path]\trunk\lib\bin\GhostCOM.dll.
```

3.3.3 Registration within the MATLAB® environment

The MSS directory and its subdirectories must be finally registered in the MATLAB® path. Since the MSS directory comprises several versions (the current version being located in the `trunk` directory and the older ones in the `tags` directory), it is recommended to do this operation ***at every start*** of MATLAB®, so that the user can choose which version must be loaded for its MATLAB® session. Registering all the sub-directories of one MSS version can be done by calling simply the script `RegisterMatlabPath.m` located at the root of every version. For example, if it is desired to work with the old version 1.1, it is sufficient to call manually the script:

```
[MSS_path]\tags\1.1\RegisterMatlabPath.m.
```

On the contrary, if one intends to work always only with the newest version of MSS, it is possible to save the MATLAB® path after the registration, so that this operation will never be required afterwards.

3.3.4 Compilation of the Simulink® S-Functions and tools

All the necessary binary files are provided as default in the MSS package. The possibility is offered to the user to recompile the tools. This requires first the installation of the GHOST C++ library, which can be downloaded at

```
http://10.61.9.62/svn/GHOST
```

and the registration of the path to the installation GHOST library by defining the value GHOST_path in the aforementioned registry key. Once this step is done, the S-Functions can be compiled using the script:

```
C:\Work\MSS\trunk\lib\mex_all.m
```

3.4 Overview of the Multi-Satellite Simulator

After the installation, the new libraries are ready for use in the Simulink® environment. A new collection of models is now available in the Simulink® Library Browser.

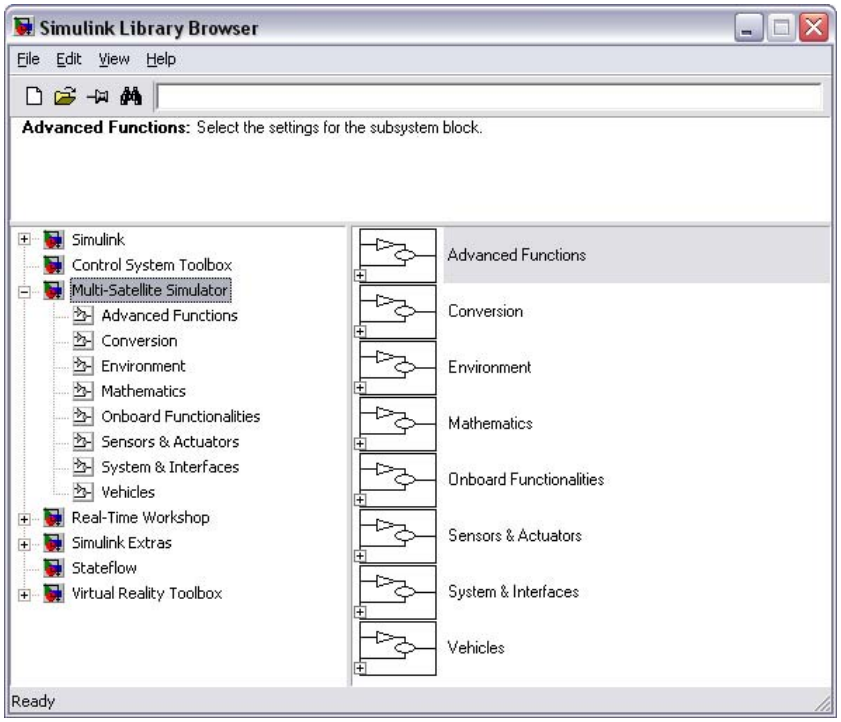


Fig. 10: Overview of the available libraries using the Simulink Library Browser

If the S-Functions have been properly compiled, the libraries can now be used. MSS comprises numerous files which are installed on the local drive of the workstation according to the following structure (the directory in blue are not part of the ***light*** version provided by the installer):

- trunk – *current version of the Multi-Satellite Simulator*
  - doc – *scripts and sources for automatic documentation*
  - gui – *graphical interface to ease the testing and documentation*
  - lib
    - bin – *compiled Simulink S-Functions (mex files)*
    - RTW – *support for automatic code generation using Real-Time Workshop*
    - scripts – *miscellaneous matlab scripts*
    - simulink – *libraries composing the Multi-Satellite Simulator*
    - src – *C++ source files of the S-Functions*
    - tlc – *Target Language Compiler files*
  - MinGS – *Minimalistic Ground Segment for real-time interaction with flight software*
  - orbit_model_data – *Auxiliary data for orbit modeling (flux, gravity, magnetic field, Earth parameters,…)*
  - setup – *sources for the installation program*
  - tests – *collection of unit and system tests & unit testing framework*
  - tools – *additional tools (written in C++) for data processing and analysis*

Fig. 11: File structure of the Multi-Satellite Simulator

4. Description of the MATLAB® Toolbox

4.1 Overview

MSS provides a collection of functions which can be called directly from the MATLAB® prompt. The functions are of different nature, depending on their complexity and dependencies:

- **The MATLAB® scripts and functions** are coded using the MATLAB® language and located in the directories

```
[MSS_path]\lib\scripts
[MSS_path]\tools
```

- **The GHOST Component Object Models** offers an easy interface under MATLAB® to the complex C++ classes of the GHOST [4] library. They are implemented as ActiveX server and can "live" in the MATLAB® workspace. They need to be declared using the syntax `actxserver`, for example:

```
SP3Reader = actxserver('GhostCOM.SP3');
SP3Reader.readFile('test.SP3');
r = SP3Reader.position('L01',[1382 0])
```

4.2 Available MATLAB® scripts and functions

The table below summarizes the available functions:

Name	Location	Purpose
EccAnom.m	\lib\scripts	Computes the eccentric anomaly for elliptic orbits
Elements.m	\lib\scripts	Computes the osculating Keplerian elements from the satellite state vector for elliptic orbits
Mean2osc.m	\lib\scripts	AIAA conversion from mean to osculating orbital elements
Modulo.m	\lib\scripts	Modulo function returning between 0 and 2?
Osc2mean.m	\lib\scripts	AIAA conversion from osculating to mean orbital elements
State.m	\lib\scripts	Computes the satellite state vector from osculating Keplerian elements for elliptic orbits

4.3 Available GHOST C++ objects and methods

Only a subset of the GHOST library is implemented. Currently, the supported classes and methods are:

GhostCOM.att	
readFile	void readFile(handle, string)
appendFile	void appendFile(handle, string)
quaternion	Variant(Pointer) quaternion(handle, string, SafeArray(double), int16)

GhostCOM.SP3	
readFile	void readFile(handle, string)
appendFile	void appendFile(handle, string)
position	Variant(Pointer) position(handle, string, SafeArray(double))
velocity	Variant(Pointer) velocity(handle, string, SafeArray(double))
visibOrb	Variant(Pointer) visibOrb(handle, string, SafeArray(double), SafeArray(double), SafeArray(double))
ImportAlmanach	void ImportAlmanach(handle, string, SafeArray(double), SafeArray(double), SafeArray(double), int16)
GhostCOM.GPStime	
Set	Variant(Pointer) Set(handle, SafeArray(double))
SetCUC	Variant(Pointer) SetCUC(handle, double, double)
plus	Variant(Pointer) plus(handle, double)
minus	Variant(Pointer) minus(handle, double)
GetDate	Variant(Pointer) GetDate(handle)
GetMjd	double GetMjd(handle)
GetCUC	Variant(Pointer) GetCUC(handle)
GetFloatCUC	double GetFloatCUC(handle)
MatlabTime	double MatlabTime(handle)
GhostCOM.LSQ	
resize	void resize(handle, int16)
Init	void Init(handle, SafeArray(double), SafeArray(double))
Accumulate	void Accumulate(handle, SafeArray(double), double, double)
Solve	Variant(Pointer) Solve(handle)
Cov	Variant(Pointer) Cov(handle)
StdDev	Variant(Pointer) StdDev(handle)
GhostCOM.RefSys	
PrecessionMatrix	Variant(Pointer) PrecessionMatrix(handle, double, double)
NutationMatrix	Variant(Pointer) NutationMatrix(handle, double)
GreenwichHourAngleMatrix	Variant(Pointer) GreenwichHourAngleMatrix(handle, double)
PoleMatrix	Variant(Pointer) PoleMatrix(handle, double, double)
ITRF_Matrix	Variant(Pointer) ITRF_Matrix(handle, SafeArray(double), int16, double, double, double)
ITRFdot_Matrix	Variant(Pointer) ITRFdot_Matrix(handle, SafeArray(double), int16, double, double, double)

5. Description of the Simulink® Libraries

5.1 Conversion

5.1.1 Cartesian to Spherical

Transformation from Cartesian to spherical coordinates.

5.1.1.1 Description

The block computes the spherical coordinates of a Cartesian vector. The angles provided as output in radians are computed using the following convention:

- r ≥0
- 0≤ longitude < 2*π positive east-wards
- 0≤ co-latitude ≤ π positive south-wards

5.1.1.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
x	1x1 double	x-coordinate
y	1x1 double	y-coordinate
z	1x1 double	z-coordinate

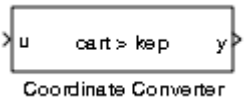
Overview of the block outputs

Name	Format	Description
r	1x1 double	radius
longitude	1x1 double	longitude
co-latitude	1x1 double	co-latitude
latitude	1x1 double	latitude

5.1.2 Coordinate Converter

Conversion between Cartesian state vectors and Keplerian orbit elements (a,e,i,Ω,ω,M).

5.1.2.1 Description



Depending on the conversion type selected in the mask, the block computes either the satellite state vector from osculating Keplerian elements or the osculating Keplerian elements from the satellite state vector. The conversion algorithms are valid for elliptic orbits. The vector of Keplerian elements is defined by:

- a: semimajor axis [m]
- e: eccentricity [-]
- i : inclination [rad]
- Ω: longitude of the ascending node [rad]
- ω: argument of pericenter [rad]
- M: Mean anomaly [rad]

Reference:

- Montenbruck O., Gill E.; Satellite Orbits - Models, Methods and Applications ; Springer Verlag, Heidelberg; (2000).

5.1.2.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Conversion type	selection of the desired conversion: from Cartesian state vector to Keplerian elements

5.1.2.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
u	1x6 double	input vector (Cartesian state vector or Keplerian elements)

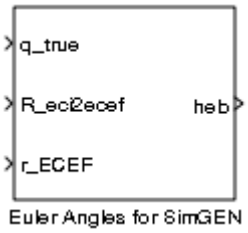
Overview of the block outputs

Name	Format	Description
y	1x6 double	output vector

5.1.3 Euler Angles for SimGEN

Provides a set of Euler angles as required by the SimGEN remote interface.

5.1.3.1 Description



The spacecraft attitude is described within SimGEN by a set of three Euler angles defined with respect to a local North-East-Down frame. The block provides these angles as output using the attitude described by the mean of a quaternion (Wertz convention) as input.

Reference:

- Spirent Communication; SimREMOTE User Manual; Issue 2-02; DGP00792AAA; (2007).

5.1.3.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
q_true	1x4 double	quaternion describing the attitude (inertial to body frame)
R_eci2ecef	1x9 double	rotation from ECI (inertial) to ECEF (Earth-centered Earth-fixed) frame
r_ECEF	1x3 double	spacecraft position vector (m) in ECEF

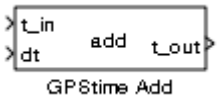
Overview of the block outputs

Name	Format	Description
heb	1x3 double	heading, elevation and bank angles as seen from SimGEN axis (NED frame)

5.1.4 GPStime Add

Add a time interval to a date in GPStime format.

5.1.4.1 Description



5.1.4.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block

5.1.4.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t_in	1x2 double	date in GPS time [GPS weeks, seconds of week]
dt	1x1 double	time interval [s]

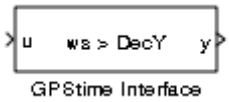
Overview of the block outputs

Name	Format	Description
t_out	1x2 double	date in GPS time [GPS weeks, seconds of week]

5.1.5 GPStime Interface

Handling of GPS time format.

5.1.5.1 Description



An accurate GPS time representation can not be achieved using a single variable because of its limited amount of significant digits. A precise time representation requires at least the use of two variables. The block is able to handle following time representations:

Description	Size of the format	Abbreviation
year, month, day, hour, minutes, seconds	6	full date
GPS week, seconds of the week	2	ws
number of entire seconds elapsed since 1980, fractional part the current second in microseconds	2	CUC

In addition the block supports date types which can not be used to describe the time accurately but are sometime useful (Modified Julian Day, short date representation, etc.). The table below summarizes the possible conversions offered by the block:

Conversion type	Description
date > ws	[YYYY MM DD] -> [Week Secs]
date > cuc	[YYYY MM DD] -> [CUC]
full date > ws	[YYYY MM DD HH MM SS.S] -> [Week Secs]
full date > cuc	[YYYY MM DD HH MM SS.S] -> [CUC]
ws > full date	[Week Sesc] -> [YYYY MM DD HH MM SS.S]
cuc > full date	[CUC] -> [YYYY MM DD HH MM SS.S]
ws > cuc	[Week Secs] -> [CUC]
cuc > ws	[CUC] -> [Week Secs]
ws > MJD	[Week Secs] -> [Modified Julian Day]
cuc >MJD	[CUC] -> [Modified Julian Day]
date > MJD	[YYYY MM DD] -> [Modified Julian Day]
ws > DecY	[Week Secs] -> [Decimal year]

5.1.5.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Conversion type	desired conversion

5.1.5.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
u	depends on the type of conversion	date in any time representation

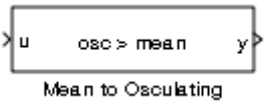
Overview of the block outputs

Name	Format	Description
y	depends on the type of conversion	date converted to the desired time representation

5.1.6 Mean to Osculating

Conversion between mean and osculating orbit elements (a,e,i,Ω,ω,M).

5.1.6.1 Description



The block performs the conversion from osculating orbital elements into mean orbital elements or vice versa. A first-order mapping algorithm is applied developed by Brouwer and Lyddane. The vector of Keplerian elements is defined by:

- a: semimajor axis [m]
- e: eccentricity [-]
- i : inclination [rad]
- Ω: longitude of the ascending node [rad]
- ω: argument of pericenter [rad]
- M: Mean anomaly [rad]

References:

- Brouwer D., Solution of the Problem of Artificial Satellite Theory Without Drag,Astronautical Journal,Vol. 64,No. 1274, pp. 378-397 (1959).
- Lyddane R.H., Small Eccentricities or Inclinations in the Brouwer Theory of the Artificial Satellite, Astronautical Journal, Vol. 68, No. 8, pp. 555-558 (1963).

5.1.6.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Conversion type	selection of the desired conversion: from osculating to mean orbit elements or vice versa

5.1.6.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
u	1x6 double	input orbit elements

Overview of the block outputs

Name	Format	Description
y	1x6 double	output orbit elements

5.1.7 Relative Elements

Compute a set of relative orbit elements between two spacecraft.

5.1.7.1 Description

Considering two spacecraft described by a set of mean orbit elements

- a: semimajor axis [m]
- e: eccentricity [-]
- i : inclination [rad]
- Ω: longitude of the ascending node [rad]
- ω: argument of pericenter [rad]
- M: Mean anomaly [rad]

the blocks provides a set of relative orbit elements defined as follows:

$$\Delta a = \begin{pmatrix} \Delta a \\ a_1 \Delta e_x \\ a_1 \Delta e_y \\ a_1 \Delta i_x \\ a_1 \Delta i_y \\ a_1 \Delta u \end{pmatrix} = \begin{pmatrix} a_2 - a_1 \\ a_1(e_2 \cos(\omega_2) - e_1 \cos(\omega_1)) \\ a_1(e_2 \sin(\omega_2) - e_1 \sin(\omega_1)) \\ a_1(i_2 - i_1) \\ a_1(\Omega_2 - \Omega_1)\sin(i_1) \\ a_1(u_2 - u_1) \end{pmatrix}$$

where the subscript 1 and 2 represent respectively the first and second inputs of the block and where u=M+ω.

Reference:

- D'Amico S., Relative Orbital Elements as Integration Constants of the Hill's Equations; TN 05-08; Deutsches Zentrum für Luft- und Raumfahrt, Oberpfaffenhofen (2005).

5.1.7.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
elem1	1x6 double	orbit elements of spacecraft 1
elem2	1x6 double	orbit elements of spacecraft 2

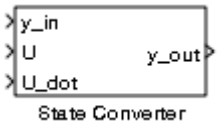
Overview of the block outputs

Name	Format	Description
relElem	1x6 double	relative orbit elements (2)-(1)

5.1.8 State Converter

Convert a state vector into a rotating frame.

5.1.8.1 Description



The state vector **y_in=[r ;v]**, written in the frame F1, is converted in the frame F2 using the transformation matrix **U** and its time derivative **U_dot** as follows: **y_out=[U.r ; U_dot.r + U.v]**.

5.1.8.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
y_in	1x6 double	input state vector written in the frame F1
U	3x3 double	transformation matrix from F1 to F2
U_dot	3x3 double	time derivative of U

Overview of the block outputs

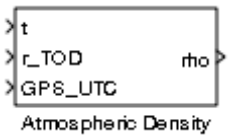
Name	Format	Description
y_out	1x6 double	output state vector written in the frame F2

5.2 Environment

5.2.1 Atmospheric Density

Provide the atmospheric density using the Jacchia-71/Gill model.

5.2.1.1 Description



The block reads an external file comprising the F10.7, F10.7, and Kp values and computes the atmospheric density using the Jacchia-71/Gill model.

NB: The altitude range supported by the model is [90,2500] km.

Reference:

- Gill E.; Smooth Bi-Polynomial Interpolation of Jacchia 1971 Atmospheric Densities For Efficient Satellite Drag Computation ; DLR-GSOC IB 96-1 (1996).

5.2.1.2 Parameters and Dialog Box

Name	Description
Flux file	path to the file containing the F10.7, F10.7, and Kp values

5.2.1.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t	1x2 double	epoch of the state in GPS time [Weeks, Seconds of Weeks]
r_TOD	1x3 double	true-of-date position vector [m]
GPS_UTC	1x1 double	difference between GPS and UTC time [s]

Overview of the block outputs

Name	Format	Description
rho	1x1 double	atmospheric density [kg/m^3]

5.2.2 Reference Systems

Provides various tranformation matrices between reference systems.

5.2.2.1 Description

The block reads the Earth's orientation parameters and the list of leap seconds from external files specified as parameters when the model starts. The transformation matrix **ITRF** between ICRF and ITRF and its time derivative are provided as output. The block provides in addition the tranformation matrices **TOD** between mean equator and equinox of J2000 and true-of-date frames as well as precession, nutation, Greenwich Hour Angle, and pole matrices. The difference between GPS and UTC times is also provided.

Reference:

- Montenbruck O., Gill E.; Satellite Orbits - Models, Methods and Applications ; Springer Verlag, Heidelberg; (2000).

5.2.2.2 Parameters and Dialog Box

Name	Description
Leap seconds (UTC-TAI [s])	difference between UTC and TAI times [s]
Earth Orientation Parameters [xPole yPole (1e-6 '') UT1-UTC (0.1 mus)]	Pole rotation [→ Arcsecond] and difference between UT1 and UTC times [0.1 → s]
Use external files instead	option to load external Earth Orientation Parameter files instead of using the values provided above
Leap second file	path to the file containing the list of leap seconds
EOP file	path to the Earth Orientation Parameter file

5.2.2.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t	1x2 double	epoch of state vector in GPS time [GPS weeks, seconds of week]

Overview of the block outputs

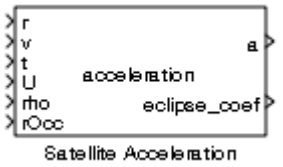
Name	Format	Description		
RS	Simulink Bus	Comprises the following signals:		
		Precession	3x3 double	precession transformation of equatorial coordinates
		Nutation	3x3 double	transformation from mean to true equator and equinox
		GHA	3x3 double	transformation from true equator and equinox to Earth equator and Greenwich meridian system
		Pole	3x3 double	Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date

TOD	3x3 double	transformation matrix from ICRF to true of date
ITRF	3x3 double	transformation matrix from ICRF to ITRF
ITRF_dot	3x3 double	time derivative of **ITRF**
GPS_UTC	1x1 double	difference between GPS and UTC time [s]

5.2.3 Satellite Acceleration

Provide the acceleration acting on a satellite.

5.2.3.1 Description



Given the position **r** and velocity **v** of the satellite at time **t** (GPS time), the blocks computes the acceleration acting on the satellite using up-to-date force models provided in the GHOST library. The following forces are included

- gravitational force using an external gravity model file
- luni-solar perturbations using precise ephemerides for the Sun and Moon
- drag modeling
- tidal effects
- relativistic effects
- solar radiation pressure using a simple cannon ball model

The block offers in addition the possibility to take an occulting body into account, which would shadow the satellite, thus reducing the solar radiation pressure. The shadow is computed using a conical shadow model.

Reference:

- Montenbruck O., Gill E.; Satellite Orbits - Models, Methods and Applications ; Springer Verlag, Heidelberg; (2000).

5.2.3.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block.
Spacecraft area	satellite property
Spacecraft mass	satellite property
Spacecraft drag coefficient	satellite property
Spacecraft radiation pressure coefficient	satellite property
Radius of occulting object	radius in [m] of an optional occulting body.
m gravity field order	desired m-order of the gravity field
n gravity field order	desired n-order of the gravity field
Sun perturbation	selection of the force model (1: selected, 0: not applied)
Moon perturbation	selection of the force model (1: selected, 0: not applied)
Tides perturbation	selection of the force model (1: selected, 0: not applied)
Relativity effect	selection of the force model (1: selected, 0: not applied)
Drag	selection of the force model (1: selected, 0: not applied)
Solar radiation pressure	selection of the force model (1: selected, 0: not applied)
Gravity file	path to the file containing the model of the gravity field

5.2.3.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
r	1x3 double	satellite position in ECI frame [m]
v	1x3 double	satellite velocity in ECI frame [m/s]
t	1x2 double	epoch of the state in GPS time [Weeks, Seconds of Weeks]
U	1x9 double	flattened transformation matrix from inertial to Earth-fixed frames (1st column - 2nd column - 3rd column)
rho	1x1 double	atmospheric density [kg/m^3]
rOcc	1x3 double	position in ECI frame [m] of an optional occulting body. Use a zero-vector if this input is not used.

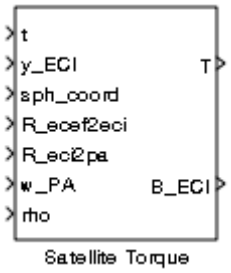
Overview of the block outputs

Name	Format	Description
a	1x3 double	acceleration [m/s²] acting on the satellite
eclipse_coef	1x1 double	eclipse coefficient [-] created by the occulting body (between 0 and 1)

5.2.4 Satellite Torque

Computes the torque sensed by an Earth orbiting satellite.

5.2.4.1 Description



The block computes the torque sensed by an Earth orbiting satellite due to:

- the Earth's spherical gravity field
- the Earth's magnetical field
- the solar radiation pressure
- the atmospheric drag

Reference:

- Montenbruck O., Gill E.; Satellite Orbits - Models, Methods and Applications ; Springer Verlag, Heidelberg; (2000).

5.2.4.2 Parameters and Dialog Box

Name	Description
Earth Magnetic Model: g_data	path to the file containing the G data from the IGRF
Earth Magnetic Model: h_data	path to the file containing the H data from the IGRF
Earth Magnetic Model: level of accuracy	desired level of accuracy
Spacecraft effective magnetic moment [A m^2]	spacecraft property
Apply magnetic field external torque	user-selectable inclusion of physical effects
Apply gravity gradient external torque	user-selectable inclusion of physical effects
Apply solar radiation pressure external torque	user-selectable inclusion of physical effects
Apply aerodynamical external torque	user-selectable inclusion of physical effects
Spacecraft inertia tensor in principal-axis-frame (PA)	spacecraft property
Spacecraft geometry properties in PA	spacecraft property
Drag coefficient	spacecraft property

5.2.4.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t	1x2 double	epoch of the state (GPStime format [Week, Seconds of Week])
y_ECI	1x6 double	position [m] and velocity [m/s] at time t in the ICRF(EME2000)
sph_coord	1x3 double	spherical ECEF coordinates: radial longitude and co-latitude [m, rad, rad]

R_ecef2eci	1x9 double	rotation matrix from ECEF to ECI
R_eci2pa	1x9 double	rotation matrix from ECI to Principal Axis (PA) frame
w_PA	1x3 double	spacecraft angular velocity vector (w.r.t. ECI) written in PA [rad/s]
rho	1x1 double	atmospheric density [kg/m^3]

Overview of the block outputs

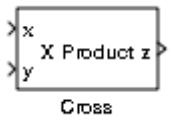
Name	Format	Description
T	1x3 double	total torque [Nm]
B_ECI	1x3 double	value of the Earth magnetic field in ECI [Tesla]

5.3 Mathematics

5.3.1 Cross

Cross product.

5.3.1.1 Description



The block computes the cross product **z=xxy**.

5.3.1.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
x	1x3 double	first variable
y	1x3 double	second variable

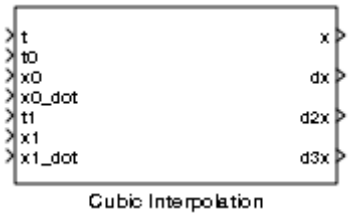
Overview of the block outputs

Name	Format	Description
z	1x3 double	result

5.3.2 Cubic Interpolation

Performs a cubic interpolation between two points.

5.3.2.1 Description



Given the values **f**(t0)=**x0** and **f**(t1)= **x1** and their time derivatives at time t0 and t1, the block interpolates the value of the function **f** at time t.

5.3.2.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t	1x1 double	desired time of interpolation [s]
t0	1x1 double	epoch of the first point
x0	1x3 double	value of the function at t0
x0_dot	1x3 double	value of the time derivative at t0
t1	1x1 double	epoch of the second point
x1	1x3 double	value of the function at t1
x1_dot	1x3 double	value of the time derivative at t1

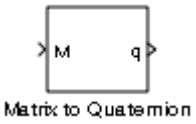
Overview of the block outputs

Name	Format	Description
x	1x3 double	interpolated value at time t
dx	1x3 double	first derivative
d2x	1x3 double	second derivative
d3x	1x3 double	third derivative

5.3.3 Matrix to Quaternion

Compute the quaternion associated to a give direction cosine matrix.

5.3.3.1 Description



Using the Wertz convention, the quaternion is computed minimizing the numerical inaccuracy.

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978).

5.3.3.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
M	3x3 double	direction cosine matrix

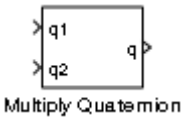
Overview of the block outputs

Name	Format	Description
q	1x4 double	quaternion

5.3.4 Multiply Quaternion

Calculate the product of two quaternions.

5.3.4.1 Description



The quaternions are expressed using the Wertz convention. The product is computed as **q=q1*q2.**

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978)

5.3.4.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
q1	1x4 double	first quaternion
q2	1x4 double	second quaternion

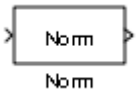
Overview of the block outputs

Name	Format	Description
q	1x4 double	resulting product

5.3.5 Norm

Norm.

5.3.5.1 Description



Provide the Euclidean norm of a vector.

5.3.5.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
u	1x3 double	input vector

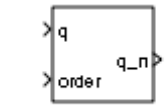
Overview of the block outputs

Name	Format	Description
v	1x1 double	norm of the vector

5.3.6 Normalize Quaternion

Normalize a quaternion.

5.3.6.1 Description



Due to numerical errors, the norm of a quaternion can be slightly different from 1. The block normalizes the input quaternion using an iterative process. The quaternions are expressed using the Wertz convention.

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978)

5.3.6.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
q	1x4 double	quaternion to be normalized
order	1x1 double	order of the iteration process

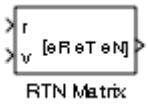
Overview of the block outputs

Name	Format	Description
q_n	1x4 double	normalized quaternion

5.3.7 RTN Matrix

Provide the transformation matrix from the inertial to the local co-moving RTN frame.

5.3.7.1 Description



Given an inertial state vector y=[r v], the unit vectors defining the Radial-Tangential-Normal frame are computed as follows:

$$e_{\mathrm{R}} = \frac{r}{\| r \|}, e_{\mathrm{N}} = \frac{r \times v}{\| r \times v \|}, e_{\mathrm{T}} = e_{\mathrm{N}} \times e_{\mathrm{R}}$$

5.3.7.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
r	1x3 double	inertial position vector [m]
v	1x3 double	inertial velocity vector [m/s]

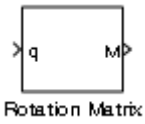
Overview of the block outputs

Name	Format	Description
[eR eT eN]	3x3 double	transformation matrix

5.3.8 Rotation Matrix

Compute the direction cosine matrix associated to a quaternion.

5.3.8.1 Description



The quaternions are expressed using the Wertz convention.

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978).

5.3.8.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
q	1x4 double	quaternion

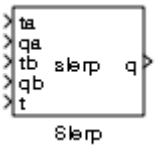
Overview of the block outputs

Name	Format	Description
M	3x3 double	direction cosine matrix

5.3.9 Slerp

Spherical Linear Interpolation (SLERP) of quaternion.

5.3.9.1 Description



Given two quaternions **qa** and **qb** at time ta and tb, the block interpolates the value of the quaternion at time t.

The given time t shall lie within the input times ta and tb.The quaternions are expressed using the Wertz convention.

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978).

5.3.9.2 Parameters and Dialog Box

Name	Description
Sample time	Sample time of the block

5.3.9.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
ta	1x2 double	epoch of the quaternion **qa** in GPS time [Weeks, Seconds of Weeks]
qa	1x4 double	input quaternion
tb	1x2 double	epoch of the quaternion **qb** in GPS time [Weeks, Seconds of Weeks]
qb	1x4 double	input quaternion
t	1x2 double	desired time for the interpolation in GPS time [Weeks, Seconds of Weeks]

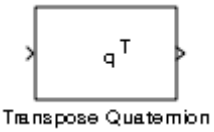
Overview of the block outputs

Name	Format	Description
q	1x4 double	interpolated quaternion at time t

5.3.10 Transpose Quaternion

Transpose a quaternion.

5.3.10.1 Description



The quaternions are expressed using the Wertz convention.

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978)

5.3.10.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
q_in	1x4 double	input quaternion

Overview of the block outputs

Name	Format	Description
q_out	1x4 double	transpose of the quaternion

5.4 System & Interfaces

5.4.1 Add Command

Add a TE command to a string buffer.

5.4.1.1 Description

This block is intended to ease the interfacing between the Phoenix GPS receiver and a GNC model under Simulink. Since the receiver communicates with ASCII messages, a "string" stream has been introduced within the Simulink environment. The "string" data type is in fact composed of a pointer to an existing string buffer (allocated in the memory by another Simulink block) and of the length of the string to be read when the block is called. The block adds the Transmit Ephemeris (TE) command received as input to this buffer. It outputs the same address to the modified buffer and updates its length accordingly.

Reference:

- Montenbruck O., Markgraf M.; User's Manual for the Phoenix GPS Receiver; v 1.9 (2008).

5.4.1.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Verbose mode	displays additional information when running

5.4.1.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
TE_cmd	1x8 uint8	Phoenix Transmit Ephemeris command: [SXT TE XX CK EXT] with XX the PRN of the GPS satellite and CK the checksum
address_in	1x1 uint32	address in the memory to an existing string buffer
length_in	1x1 uint32	length of the buffer to be read by the block

Overview of the block outputs

Name	Format	Description
address_out	1x1 uint32	same address as provide as input
length_out	1x1 uint32	updated length (=input length+8)

5.4.2 Modular Interface: Client

Interface for the exchange of data between two Simulink models through a TCP network.

5.4.2.1 Description

Modular Interface: Client

The block implements an advanced interface for the exchange of data using the TCP protocole. Being designed to serve a large variety of purpose, the block is fully configurable through the external definition of interfaces.

The interface definition is done using a structure which is provided as parameter to the block. The following fields must be supplied:

- : name of the port
- :

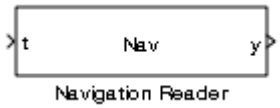
5.4.2.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Server address	IP address of the server
Port	TCP port (must be identical on both client and server blocks)
Structure defining the interfaces	see description above
Array of TCP tags	see description above
Verbose	displays additional information when running

5.4.3 Navigation Reader

Read the content of a navigation file (SP3-like format) at a specific time.

5.4.3.1 Description



The blocks aims at reading navigation data coming from an absolute positioning sensor, typically a GPS receiver. Since the purpose is to replay exactly what the sensor has measured, no interpolation is done. If the time provided at input matches a navigation record within the file, then the block returns the value read in the file. If not, the blocks outputs a zero vector. The navigation data file must be in SP3-like format, i.e. recorded like a SP3 format but without header and without the need of having equidistant measurement epochs.

Reference:

- Spofford P.R., Remondi B.W.; The National Geodetic Survey Standard GPS Format SP3; NOAA/NGS.

5.4.3.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Product file(s)	path to the navigation file to be read in SP3-like format
Spacecraft descriptor	string describing the spacecraft identifier
Verbose mode	display additional information when running

5.4.3.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t	1x2 double	desired time in GPS time format [GPS weeks, seconds of week]

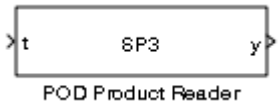
Overview of the block outputs

Name	Format	Description
y	1x6 double	state vector [m,m/s] read from the file at time t

5.4.4 POD Product Reader

Read the content of a SP3 or attitude file at a specific time.

5.4.4.1 Description



The block loads the SP3 or attitude files specified in the dialog box when the model starts. It provides as output the state or attitude at the epoch given as input. If the input time does not belong to the epochs recorded in the file, the output is interpolated.

The following files formats are accepted:

- SP3 file : ASCII file following the National Geodetic Survey Standard SP3 GPS Format
- attitude file: ASCII file following the convention of the GHOST C++ library.

Warning: Due to historical reasons, the ".att" file read by the block lists columnwise the time-tagged quaternions describing the rotation from body-fixed to inertial frames using the DLR/GSOC conventions, i.e. [q0 -q1 -q2 -q3] where q0 is the scalar part. The block outputs the transpose of the quaternions as described in the file in order to describe the rotation from inertial to body-fixed frames. The block can use as convention for the output quaternion either the DLR/GSOC convention or the Wertz convention

Reference:

- Spofford P.R., Remondi B.W.; The National Geodetic Survey Standard GPS Format SP3; NOAA/NGS.
- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978)
- Montenbruck O.; Quaternion Representation of BIRB Orientation and Reference System Transformations; DLR-GSOC TN 00-03 (2000)

5.4.4.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Product file(s)	list of files to be loaded by the block. The list is specified using a Matlab cell arrays of strings, i.e. a list of n files is written by: {'file1','file2',...,'filen'}.
Spacecraft descriptor	string describing the spacecraft identifier
Product type	type of file read by the block (either SP3 or attitude file)
Quaternion convention	convention used to output the quaternions (either DLR/GSOC or Wertz)
Verbose mode	display additional information when running

5.4.4.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description

t	1x2 double	epoch of the data to be retrieved in GPS time [GPS weeks,seconds of week]

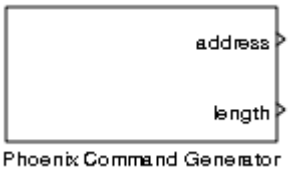
Overview of the block outputs

Name	Format	Description
y (if product type=**SP3**)	6 x double	State [m, m/s] at the epoch specified as input
y (if product type=**att**)	4 x double	Quaternion [-] at the epoch specified as input

5.4.5 Phoenix Command Generator

Output time-tagged commands for the Phoenix GPS receiver.

5.4.5.1 Description



The block takes as parameter a structure comprising a list of time-tagged commands and outputs them at the proper time. Since the receiver communicates with ASCII messages, a "string" stream has been introduced within the Simulink environment. The "string" data type is composed of a pointer to an existing string buffer (allocated in the memory by the block) and of the length of the string to be read when the block is called.

No checksum nor message delimiters (SXT and EXT) are required in the definition of telecommands. The block will add this additional information automatically. The list of time-tagged command is defined using a structure with the fields **t** and **cmd**. The time **t** used to tag the telecommands refers to the elapsed time since the begining of the simulation. For example, if the following structure is provided as input:

```
CommandList(1).t=10;
CommandList(1).cmd={'DR4010','SE20080101123000GPS'};
CommandList(2).t=20;
CommandList(2).cmd={'AM1'};
...
```

the block will write at simulation time t=10 and t=20 the desired messages in the string buffer and set the length to be read to non zero. The size of the buffer is limited to 5000 characters.

Reference:

- Montenbruck O., Markgraf M.; User's Manual for the Phoenix GPS Receiver; v 1.9 (2008).

5.4.5.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
List of time-tagged commands	structure comprising the list of parameters (cf. description above)
Verbose	display additional information when running

5.4.5.3 Inputs and Outputs

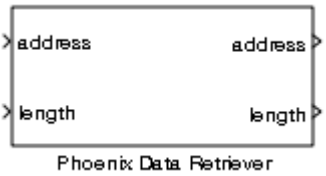
Overview of the block outputs

Name	Format	Description
address	1x1 uint32	address in the memory to the string buffer allocated within the block
length	1x1 uint32	length of the buffer to be read by other blocks

5.4.6 Phoenix Data Retriever

Serial interface to the GPS Phoenix receiver.

5.4.6.1 Description



The block enables the real-time serial communication with the GPS Phoenix receiver. Since the receiver communicates with ASCII messages, a "string" stream has been introduced within the Simulink environment. The "string" data type is composed of a pointer to an existing string buffer (allocated in the memory by the block) and of the length of the string to be read when the block is called. The block allocates internally a string buffer and writes the messages received from the Phoenix GPS receiver.

Thanks to its multi-threading architecture, the behavior of the block does not depend on the arrival of data. When the block is called, it writes in the string buffer all the messages arrived since the last call to the block. The block gets also as input the address to an existing string buffers and sends its content to the Phoenix GPS receiver.

Finally, the block offers the possibility to transmit and receive data in real-time from an external application using the TCP protocole. This allows operating manually in real-time the receiver when the simulink model is running.

Reference:

- Montenbruck O., Markgraf M.; User's Manual for the Phoenix GPS Receiver; v 1.9 (2008).

5.4.6.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Serial port number	port number connected to the Phoenix GPS receiver
Verbose mode	display additional information when running
Enable TCP support	enable the transmission of data in real-time to an external TCP server
Remote IP address	IP address of the server
Remote port	TCP port on which the server is waiting for incoming connection

5.4.6.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
address	1x1 uint32	address in the memory to an existing string buffer
length	1x1 uint32	length of the buffer to be read by the block

Overview of the block outputs

Name	Format	Description
address	1x1 uint32	address in the memory to the string buffer allocated within the block
length	1x1 uint32	length of the buffer to be read by other blocks

5.4.7 Phoenix Message Logging

Write the content of a string buffer into a text file.

5.4.7.1 Description

Since the Phoenix GPS receiver communicates with ASCII messages, a "string" stream has been introduced within the Simulink environment. The "string" data type is composed of a pointer to an existing string buffer (allocated in the memory by another Simulink block) and of the length of the string to be read when the block is called. Everytime the block is called, it reads the number of bytes specified by the input length from the string buffer and writes it to a text file.

Reference:

- Montenbruck O., Markgraf M.; User's Manual for the Phoenix GPS Receiver; v 1.9 (2008).

5.4.7.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Log file name	path to the output text file

5.4.7.3 Inputs and Outputs

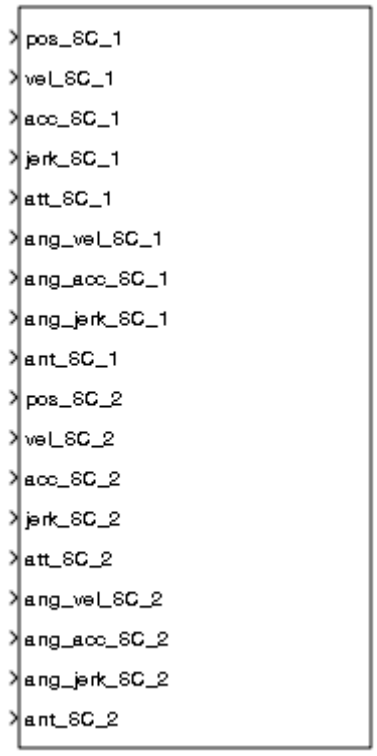
Overview of the block inputs

Name	Format	Description
address	1x1 uint32	address in the memory to an existing string buffer
length	1x1 uint32	length of the buffer to be read by the block

5.4.8 SimGEN Interface

Send motion commands through a TCP network to SimGEN.

5.4.8.1 Description



SimGEN Interface

The block transmits motion commands to SimGEN using TCP. Up to two spacecraft are supported. SimGEN works only at 1Hz or 10Hz. As a consequence, only 1s or 100ms sample time are allowed for the block. The block is intended to work in real-time. To that end, it slows down the Simulink model by artificially delaying the execution time of the block in order to synchronize the Simulink time with the really elapsed time. This can be achieved using the internal clock of the computer or with an external synchronization signal sent by the GPS Signal Simulator and retrieved by a dedicated timer card. The motion command messages are provided shortly before the SimGEN simulation tick (every 100 ms or 1s). Currently, the block supports only one model of timer card: the Amplicon PCI215 digital I/O 48 channels and 6 counter timers PCI board.

Reference:

- Spirent Communication; SimREMOTE User Manual; Issue 2-02; DGP00792AAA; (2007).

5.4.8.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block (1 or 0.1)
SimGEN server IP address	IP address of the computer on which SimGEN is running
Number of spacecraft	number of spacecraft (1 or 2)
Clock Type	select the internal clock or an external reference clock through a timer card

Time interval to provide the messages in advance [ms]	time interval to send the motion commands before the SimGEN simulation tick
Simulation start [YYYY MM DD HH MM SS]	Epoch of the simulation, will be sent to SimGEN during the initialization
Simulation duration [s]	Duration of the simulation, will be sent to SimGEN during the initialization
Verbose mode	display additional information when running

5.4.8.3 Inputs and Outputs

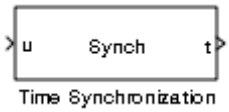
Overview of the block inputs

Name	Format	Description
pos_SC_1	1x3 double	ECEF position [m] of spacecraft 1
vel_SC_1	1x3 double	ECEF velocity [m/s] of spacecraft 1
acc_SC_1	1x3 double	ECEF acceleration [m/s^2] of spacecraft 1
jerk_SC_1	1x3 double	ECEF jerk [m/s^3] of spacecraft 1
att_SC_1	1x3 double	Euler angle describing the spacecraft 1 using the SimGEN convention [rad]
ang_vel_SC_1	1x3 double	angular velocity [rad/s] of spacecraft 1
ang_acc_SC_1	1x3 double	angular acceleration [rad/s^2] of spacecraft 1
ang_jerk_SC_1	1x3 double	angular jerk [rad/s^3] of spacecraft 1
ant_SC_1	1x6 double	antenna offset[3xm] and orientation [3x-] of spacecraft 1
pos_SC_2	1x3 double	ECEF position [m] of spacecraft 2
vel_SC_2	1x3 double	ECEF velocity [m/s] of spacecraft 2
acc_SC_2	1x3 double	ECEF acceleration [m/s^2] of spacecraft 2
jerk_SC_2	1x3 double	ECEF jerk [m/s^3] of spacecraft 2
att_SC_2	1x3 double	Euler angle describing the spacecraft 2 using the SimGEN convention [rad]
ang_vel_SC_2	1x3 double	angular velocity [rad/s] of spacecraft 2
ang_acc_SC_2	1x3 double	angular acceleration [rad/s^2] of spacecraft 2
ang_jerk_SC_2	1x3 double	angular jerk [rad/s^3] of spacecraft 2
ant_SC_2	1x6 double	antenna offset[3xm] and orientation [3x-] of spacecraft 2

5.4.9 Time Synchronization

Synchronize the Simulink time with the real time.

5.4.9.1 Description



The block slows down a Simulink model by increasing artificially its computational time. This allows synchronizing the Simulink time with the real time measured with the computer internal clock. In order to better control when the block is exactly called, the block is augmented with one useless input and one output which simply gives the time of the simulation. This allows including the block in a chain of blocks.

5.4.9.2 Parameters and Dialog Box

Name	Description
Synchronization interval in ms	Time interval between two Simulink ticks

5.4.9.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
u	1x1 double	useless input

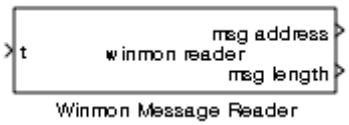
Overview of the block outputs

Name	Format	Description
t	1x1 double	time of the simulation [s]

5.4.10 Winmon Message Reader

Replay the time-tagged Mitel messages output by the Phoenix receiver and stored in a log file.

5.4.10.1 Description



Winmon Message Reader

Since the Phoenix GPS receiver communicates with ASCII messages, a "string" stream has been introduced within the Simulink environment. The "string" data type is composed of a pointer to an existing string buffer (allocated in the memory by the block) and of the length of the string to be read when the block is called. The blocks writes all the messages found at GPS time t, specified as input, into the string buffers. If the time of the message does not match exactly the input time, the blocks writes all the messages arrived since the last call of the block.

Reference:

- Montenbruck O., Markgraf M.; User's Manual for the Phoenix GPS Receiver; v 1.9 (2008).

5.4.10.2 Parameters and Dialog Box

Name	Description
Winmon filename	path to the name of the file comprising the Mitel messages
Sample time	sample time of the block
Verbose	display additional information when running

5.4.10.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t	1x2 double	desired time in GPS time format [GPS weeks, seconds of week]

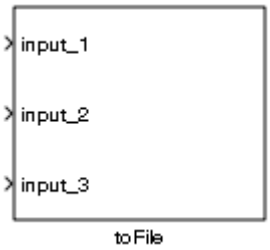
Overview of the block outputs

Name	Format	Description
msg address	1x1 uint32	address in the memory to the string buffer allocated within the block
msg length	1x1 uint32	length of the buffer to be read by other blocks

5.4.11 toFile

Write multiple variable in one single matlab output file.

5.4.11.1 Description



The block takes as many inputs as desired. The user needs simply to specify the total number and the names of the inputs in the mask, the block will recognize the size and data types of the inputs.

The block has in addition the ability to split the output file into several smaller files in order to avoid the generation of too voluminous files

Warning: In order to be compatible with RTW, the block requires the parameter describing the list of inputs to be an array of chars instead of a structure. This can be achieved using the matlab command: char({'input_1','input_2','input_3'})

5.4.11.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Number of inputs	number of inputs
Name of the inputs	char array describing the inputs
Output file	path to the matlab ouput file
Maximum file size (Mb)	maximum size for the output file before being split in smaller files.
Verbose	display additional information when running

5.4.11.3 Inputs and Outputs

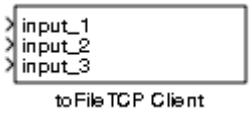
Overview of the block inputs

Name	Format	Description
user defined inputs	user defined	user defined

5.4.12 toFileTCP Client

Send multiple variable through a TCP network and write them in one single matlab output file.

5.4.12.1 Description



The block behaves like the toFile block, except that the data are transmitted first through a TCP network. This allows for the logging of data using target computers which do not have any local storage device. The server block runs on a machine equipped whith a storage device and received remotely data which are sent in real-time by the client block, runnign on the target computer.

5.4.12.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Server address	IP address of the server
Port	TCP port used for the communication
Number of inputs	number of inputs
Name of the inputs	char array describing the inputs
Verbose	display additional information when running

5.4.12.3 Inputs and Outputs

Overview of the block inputs

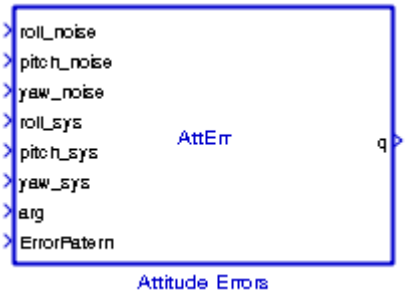
Name	Format	Description
user defined inputs	user defined	user defined

5.5 Sensors & Actuators

5.5.1 Attitude Errors

Generate a quaternion describing attitude errors, which can be applied for attitude estimation or control.

5.5.1.1 Description



The block takes random error and systematic error pattern into account and returns the quaternion associated with the rotation matrix AttErr = R_x(roll_error) * R_y(pitch_error) * R_z(yaw_error). The roll, pitch and yaw errors are computed by summing the noise values and an error pattern if this option is selected. Currently, only the sinusoidal pattern is supported, so that the systematic error is computed as error=error_systematic*sin(arg).

5.5.1.2 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
roll_noise	1x1 double	roll random error [rad]
pitch_noise	1x1 double	pitch random error [rad]
yaw_noise	1x1 double	yaw random error [rad]
roll_sys	1x1 double	roll systematic error [rad]
pitch_sys	1x1 double	pitch systematic error [rad]
yaw_sys	1x1 double	yaw systematic error [rad]
arg	1x1 double	pulsation of the sinusoidal error pattern
ErrorPatern	1x1 double	1 if the sinusoidal pattern is selected, 0 if no systematic error pattern is applied

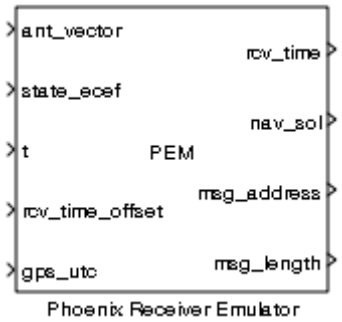
Overview of the block outputs

Name	Format	Description
q	1x4 double	quaternion describing the attitude errors

5.5.2 Phoenix Receiver Emulator

Model of the Phoenix GPS receiver.

5.5.2.1 Description



The block provides ASCII navigation messages (only F14, F4 and F62) as output by the real receiver. To that end, a "string" stream has been introduced within the Simulink environment. "string" data type is composed of a pointer to an existing string buffer (allocated in the memory by the block) and of the length of the string to be read when the block is called.

The almanach of the GPS constellation is currently hardcoded in the model.

Reference:

- Montenbruck O., Markgraf M.; User's Manual for the Phoenix GPS Receiver; v 1.9 (2008).

5.5.2.2 Parameters and Dialog Box

Name	Description
Sample time	Sample time of the simulation [s]
B_DLL	DLL bandwidth [Hz]
B_PLL	PLL bandwidth [Hz]
elevMask	elevation mask [deg]
smoothing	flag
start_time	Scenario start time [YY MM DD HH MM SS]
GPS_UTC	difference GPS-UTC [s]
ephErr	broadcast ephemeris error [m]
VTEC	vertical TEC (in TECU)
day_night	flag for TEC variation
iSeed	Seed for internal random number generator
GPS Almanac	YUMA Almanac used to simulate the constellation
Antenna pointing mode (for channel allocation)	AP command: Zenith or ECEF pointing

5.5.2.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
ant_vector	1x3 double	boresight for chan. allocation

state_ecef	1x6 double	Earth-fixed state of the spacecraft [m,m/s]
t	1x2 double	epoch of the state vector in GPS time [GPS weeks,seconds of week]
rcv_time_offset	1x2 double	receiver clock and frequency errors [s, Hz]
gps_utc	1x1 double	difference between GPS and UTC times [s]

Overview of the block outputs

Name	Format	Description
rcv_time	1x2 double	GPS time of the navigation solution [GPS weeks,seconds of week]
nav_sol	1x6 double	Earth-fixed navigation solution
msg_address	1x1 uint32	address in the memory to the string buffer allocated within the block
msg_length	1x1 uint32	length of the buffer to be read by other blocks

5.5.3 Thruster Model

Simple model of a thruster affected by execution errors.

5.5.3.1 Description

The block applies a random Gaussian error on the thrust vector. The burn duration is as well modified according to the thruster parameters.

5.5.3.2 Parameters and Dialog Box

Name	Description
Fundamental simulation step	Sample time of the block
Seed	Seed number for the random generator
Size error mean [percent]	Mean value of the thrust error
Size error std [percent]	Standard deviation of the thrust error
Minimum burn time [s]	Minimum burn time of the thruster system
Quantization of burn time [s]	Quantization step of the thruster system

5.5.3.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
start_time	1x1 double	start time of the thrust
end_time	1x1 double	end time of the thrust
thrust	1x3 double	inertial thrust vector

Overview of the block outputs

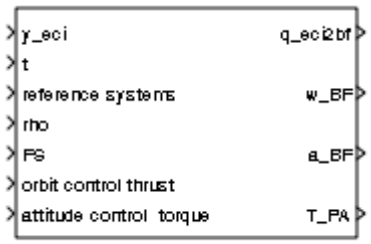
Name	Format	Description
to propagator	1x5 double	vector [start_time; end_time; thrust]

5.6 Advanced Functions

5.6.1 Attitude Dynamics

Attitude propagation using the force models of the GHOST library.

5.6.1.1 Description



Attitude Dynamics

Attitude propagation using the force models of the GHOST library.

5.6.1.2 Parameters and Dialog Box

Name	Description
Spacecraft inertia tensor in principal-axis-frame (PA)	spacecraft inertia tensor in principal-axis-frame
Spacecraft geometrical center position in PA	spacecraft geometrical center position in principal-axis-frame
Spacecraft geometry properties in PA	spacecraft geometry properties in principal-axis-frame
Rotation matrix BF>PA	rotation matrix describing the rotation matrix from body frame to principal axes
Spacecraft initial angular velocity error in BF	spacecraft initial angular velocity error in body frame [rad/s]
Spacecraft initial attitude (CDM) error wrt BF	cosine direction matrix describing the spacecraft initial attitude error with respect to the body frame
Order for the normalization of the quaternion	order for the normalization of the quaternion
Gravity gradient external torque	selection of the force model (1: selected, 0: not applied)
Magnetic field external torque	selection of the force model (1: selected, 0: not applied)
Solar radiation pressure external torque	selection of the force model (1: selected, 0: not applied)
Aerodynamic external torque	selection of the force model (1: selected, 0: not applied)
Engine firings internal torque	selection of the force model (1: selected, 0: not applied)
Earth Magnetic Model IGRF 2010: g_data	path to the file containing the G data of the Earth Magnetic Model IGRF 2010
Earth Magnetic Model IGRF 2010: h_data	path to the file containing the H data of the Earth Magnetic Model IGRF 2010
Earth Magnetic Model: level of accuracy. 1 --> Dipole; 2 --> Complete	level of accuracy of the Earth Magnetic Model
Spacecraft effective magnetic moment [A m^2]	spacecraft effective magnetic moment [A m^2]

5.6.1.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
y_eci	1x6	inertial state vector [m m/s]

	double	
t	1x2 double	epoch of the state in GPS time [Weeks, Seconds of Week]
reference systems	Simulink Bus	Simulink bus comprising the following signals:

Precession	3x3 double	precession transformation of equatorial coordinates
Nutation	3x3 double	transformation from mean to true equator and equinox
GHA	3x3 double	transformation from true equator and equinox to Earth equator and Greenwich meridian system
Pole	3x3 double	Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date
TOD	3x3 double	transformation matrix from ICRF to true of date
ITRF	3x3 double	transformation matrix from ICRF to ITRF
ITRF_dot	3x3 double	time derivative of **ITRF**
GPS_UTC	1x1 double	difference between GPS and UTC time [s]

rho	1x1 double	atmospheric density [kg/m^3]
FS	Simulink Bus	Comprises the following signals:

q_ref2bf	1x4 double	rotation from reference frame to body frame
q_eci2bf_des	1x4 double	rotation from inertial frame to body frame
q_eci2ref	1x4 double	rotation from inertial frame to reference frame
wBFdes	1x3 double	angular velocity profile to be tracked (imposed reference or selected pointing mode)
sun_uv_eci	1x3 double	unit vector towards the Sun (in inertial frame)

orbit control thrust	1x5 double	thrust command: vector of dimension 5 describing the control action to be applied. The first two vector elements are the start and end time of the thrust interval (in Simulink time, starting from zero at the begining of the Simulation). The remaining three elements describe the thrust vector written in the inertial frame.
attitude control torque	1x3 double	torque command [Nm]

Overview of the block outputs

Name	Format	Description
q_eci2bf	1x4 double	quaternion describing the spacecraft attitude from inertial to body frame
w_BF	1x3 double	spacecraft angular velocity in body frame [m/s]
a_BF	1x3 double	spacecraft angular acceleration in body frame [m/s²]
T_PA	1x3 double	torque sensed by the spacecraft in principle axes

5.6.2 Frames Setting

Provide the rotation associated to a user-defined attitude profile.

5.6.2.1 Description

The block provides the quaternion describing the rotation from the inertial frame to the body frame following a specific attitude profile. The definition of the attitude is done by selecting the reference frame which can be:

- arbitrary inertial frame described by a rotation matrix given as parameter
- co-moving Radial-Tangential-Normal (RTN) frame
- Sun-Zenith pointing frame
- Profile imposed by an external attitude guidance

and then by aligning the body axes of the spacecraft with the selected reference frame. For example, the user can choose that the spacecraft x axis has to be aligned with the Sun and the y axis should point towards the Zenith.

Reference:

- Wertz J. R.; Spacecraft Attitude Dermination and Control; Kluwer (1978)

5.6.2.2 Parameters and Dialog Box

Name	Description
Reference frame (REF) assumed for defining the desired attitude	inertially fixed, RTN,Sun-Zenit pointing or external guidance
If "Inertial fixed", constant cosine-matrix (CDM) written in inertial-Earth-centered (ECI)	rotation matrix describing the inertially fixed frame (only used if the chosen reference frame is inertially fixed.
Desired attitude: x_BF direction w.r.t. REF	axis of the reference frame to be aligned with the x axis of the body. Not used if external guidance mode is selected.
Desired attitude: y_BF direction w.r.t. REF	axis of the reference frame to be aligned with the y axis of the body. Not used if external guidance mode is selected.

5.6.2.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description		
y_ECI	1x6 double	inertial position and velocity at time t [m m/s]		
t	1x2 double	epoch of the state in GPS time [Weeks, Seconds of Week]		
ExtAtt	Simulink Bus	Simulink bus for external guidance mode comprising the following signals:		
		q_eci2bfT	1x4 double	rotation from inertial to body frame to be tracked
		w_BFT	1x4 double	angular velocity profile to be tracked

Overview of the block outputs

Name	Format	Description
FS	Simulink Bus	Comprises the following signals:

q_ref2bf	1x4 double	rotation from reference frame to body frame
q_eci2bf_des	1x4 double	rotation from inertial frame to body frame
q_eci2ref	1x4 double	rotation from inertial frame to reference frame
wBFdes	1x3 double	angular velocity profile to be tracked (imposed reference or selected pointing mode)
sun_uv_eci	1x3 double	unit vector towards the Sun (in inertial frame)

5.6.3 Orbit Dynamics

Orbit propagation using the force models of the GHOST library.

5.6.3.1 Description

Given the spacecraft parameters and initial state vector, the block integrates two times the values provided by the block Satellite Acceleration to provide the inertial state of the spacecraft at time t. The block includes in the force model the optional control action provided as input. This control is described by a thrust vector, written in the inertial frame, over a time interval. Depending on the size of the control action, the block chooses wheter to include the control as extended maneuver or as impulsive maneuver.

The block offers in addition the possibility to take an occulting body into account, which would shadow the satellite, thus reducing the solar radiation pressure. The shadow is computed using a conical shadow model.

5.6.3.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block. The value -1 is forbidden, because the sample time is used to determine if the maneuvers needs to be considered as impulsive or extended.
initial epoch (GPS weeks, GPS secs)	initial epoch of the state vector in GPS time [Weeks, Seconds of Week]
initial state in the ICRF frame	initial inertial state vector [m m/s]
spacecraft area	spacecraft area [m]
spacecraft mass	spacecraft mass [kg]
spacecraft drag coefficient	spacecraft drag coefficient [-]
spacecraft solar radiation pressure coefficient	spacecraft solar radiation pressure coefficient[-]
Threshold to consider a maneuver as impulsive [s]	if the maneuver duration is smaller than this threshold, the maneuver will be considered as impulsive.
Radius of occulting body	radius in [m] of an optional occulting body.
m gravity field order	desired m-order of the gravity field
n gravity field order	desired n-order of the gravity field
Apply drag	selection of the force model (1: selected, 0: not applied)
Apply solar radiation pressure	selection of the force model (1: selected, 0: not applied)
Apply Sun perturbation	selection of the force model (1: selected, 0: not applied)
Apply Moon perturbation	selection of the force model (1: selected, 0: not applied)
Apply Tidal perturbation	selection of the force model (1: selected, 0: not applied)
Apply relativity effect	selection of the force model (1: selected, 0: not applied)
Gravity file	path to the file containing the model of the gravity field
Flux file	path to the file containing the F10.7, F10.7, and Kp values

5.6.3.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
thrust command	1x5 double	thrust command: vector of dimension 5 describing the control action to be applied. The first two vector elements are the start and end time of the thrust interval (in Simulink time, starting from zero at the begining of the Simulation). The remaining three elements describe the thrust vector written in the inertial frame.
reference systems	Simulink Bus	Simulink bus comprising the following signals:

Precession	3x3 double	precession transformation of equatorial coordinates
Nutation	3x3 double	transformation from mean to true equator and equinox
GHA	3x3 double	transformation from true equator and equinox to Earth equator and Greenwich meridian system
Pole	3x3 double	Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date
TOD	3x3 double	transformation matrix from ICRF to true of date
ITRF	3x3 double	transformation matrix from ICRF to ITRF
ITRF_dot	3x3 double	time derivative of **ITRF**
GPS_UTC	1x1 double	difference between GPS and UTC time [s]

r_occulting	2x3 double	position in ECI frame [m] of an optional occulting body. Use a zero-vector if this input is not used.

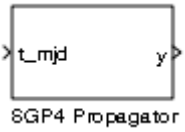
Overview of the block outputs

Name	Format	Description
y_eci	1x6 double	inertial state vector [m m/s]
t	1x2 double	epoch of the state in GPS time [Weeks, Seconds of Week]
rho	1x1 double	atmospheric density [kg/m^3]
eclipse coefficient	1x1 double	eclipse coefficient [-] created by the occulting body (between 0 and 1)
maneuver type	1x1 double	0: no maneuver applied; 1: extended maneuver applied; 2: impulsive maneuver applied

5.6.4 SGP4 Propagator

The SGP4 Orbit Propagator generates position and velocity of a Low Earth Orbit (LEO) spacecraft at a given time. Position and velocity are computed based on the NORAD SGP4 orbit prediction model.

5.6.4.1 Description



Given a set of mean orbital elements at time t0:

- a: Mean semimajor axis [m]
- e: Mean eccentricity [-]
- i : Mean inclination [rad]
- Ω: Mean right ascention of of the ascending node [rad]
- ω: Mean argument of perigee [rad]
- M: Mean anomaly [rad]

the block propagates the state at time t given at input using the Norad orbit propagator.

Reference:

- Brouwer D.; Solution of the Problem of Artificial Satellite Theory without Drag; Astronomical Journal vol. 64, pp. 378-397 (1959).
- van Flandern T.C., Pulkinnen K.F.; Low-Precision Formulae for Planetary Positions; Astrophys. Journ. Suppl. vol. 41, pp. 391 (1979).
- Hoots F. R., Roehrich R. L.; Models for propagation of NORAD element sets; Spacecraft Report No 3; Aerospace Defense Command, United States Air Force; Dec. (1980). 2nd ed. by Kelso T. S., December (1988).
- Lane M.H., Fitzpatrick P.M., Murphy J.J.; On the Representation of Air Density in Satellite Deceleration Equations by Power Functions with Integral Exponents; Project Space Track Technical Report No. APGC-TDR-62-15, March 1962, Air Force Systems Command, Eglin AFB, FL (1962).
- Lane M.H., Cranford K.H.; An Improved Analytical Drag Theory for the Artificial Satellite Problem; AIAA Paper No. 69-925 (1969).
- Lane M.H., Hoots, F.R.; General Perturbations Theories Derived from the 1965 Lane Drag Theory; Project Space Track Report No. 2, December 1979, Aerospace Defense Command, Peterson AFB, CO. (1979).

5.6.4.2 Parameters and Dialog Box

Name	Description
Sample time	sample time of the block
Orbital elements epoch	epoch [mjd]of the orbital elements provided as parameters
Mean orbital elements	vector of mean Keplerian elements [a,e,i,Ω,ω,M]

Ballistic coefficient	ballistic coefficient (B=0.5*C_D*A/m [m^2/kg])

5.6.4.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
t_mjd	1x1 double	Modified Julian date

Overview of the block outputs

Name	Format	Description
y	1x6 double	True-of-date inertial state vector [m,m/s]

5.7 Vehicles

5.7.1 Minimalistic Spacecraft

Spacecraft without sensor and actuator model, following an ideal attitude profile.

5.7.1.1 Description

Minimalistic Spacecraft

The block combines the blocks **Orbit Dynamics** and **Frames Setting** to model the translational motion using advanced force models from the GHOST library (cf. description of **Orbit Dynamics** for further details). This spacecraft model assumes that the attitude is controlled onboard to follow an attitude profile specified by the user (cf. description of **Frames Setting** for further details).

5.7.1.2 Parameters and Dialog Box

Name	Description
Fundamental simulation step	sample time of the block. The value -1 is forbidden, because the sample time is used to determine if the maneuvers needs to be considered as impulsive or extended.
Epoch of the state	initial epoch of the state vector in GPS time [Weeks, Seconds of Week]
Initial state (inertial)	initial inertial state vector [m m/s]
Area	spacecraft area [m]
Mass	spacecraft mass [kg]
Drag coefficient	spacecraft drag coefficient [-]
Solar radiation pressure coefficient	spacecraft solar radiation pressure coefficient[-]
Gravity file	path to the file containing the model of the gravity field
Flux file	path to the file containing the F10.7, F10.7, and Kp values
m gravity field order	desired m-order of the gravity field
n gravity field order	desired n-order of the gravity field
Reference frame (REF) assumed for defining the desired attitude	inertially fixed, RTN,Sun-Zenit pointing or external guidance
If "Inertial fixed", constant cosine-matrix (CDM) written in inertial-Earth-centered (ECI)	rotation matrix describing the inertially fixed frame (only used if the chosen reference frame is inertially fixed
Desired attitude: x_BF direction w.r.t. REF	axis of the reference frame to be aligned with the x axis of the body
Desired attitude: y_BF direction w.r.t. REF	axis of the reference frame to be aligned with the y axis of the body
Apply drag	selection of the force model (1: selected, 0: not applied)
Apply solar radiation pressure	selection of the force model (1: selected, 0: not applied)
Apply Sun perturbation	selection of the force model (1: selected, 0: not applied)
Apply Moon perturbation	selection of the force model (1: selected, 0: not applied)
Apply Tidal perturbation	selection of the force model (1: selected, 0: not applied)
Apply relativity effect	selection of the force model (1: selected, 0: not applied)

5.7.1.3 Inputs and Outputs

Overview of the block inputs

Name	Format	Description
thrust_eci	1x5 double	thrust command: vector of dimension 5 describing the control action to be applied. The first two vector elements are the start and end time of the thrust interval (in Simulink time, starting from zero at the begining of the Simulation). The remaining three elements describe the thrust vector written in the inertial frame.
RefSys	Simulink Bus	Simulink bus comprising the following signals:

Precession	3x3 double	precession transformation of equatorial coordinates
Nutation	3x3 double	transformation from mean to true equator and equinox
GHA	3x3 double	transformation from true equator and equinox to Earth equator and Greenwich meridian system
Pole	3x3 double	Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date
TOD	3x3 double	transformation matrix from ICRF to true of date
ITRF	3x3 double	transformation matrix from ICRF to ITRF
ITRF_dot	3x3 double	time derivative of **ITRF**
GPS_UTC	1x1 double	difference between GPS and UTC time [s]

ExtAtt	Simulink Bus	Simulink bus for external guidance mode comprising the following signals:

q_eci2bfT	1x4 double	rotation from inertial to body frame to be tracked
w_BFT	1x4 double	angular velocity profile to be tracked

Overview of the block outputs

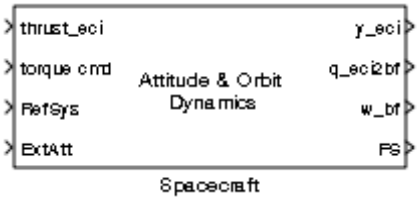
Name	Format	Description
y_eci	1x6 double	inertial state vector
FS	Simulink Bus	Comprises the following signals:

q_ref2bf	1x4 double	rotation from reference frame to body frame
q_eci2bf_des	1x4 double	rotation from inertial frame to body frame
q_eci2ref	1x4 double	rotation from inertial frame to reference frame
wBFdes	1x3 double	angular velocity profile to be tracked (imposed reference or selected pointing mode)
sun_uv_eci	1x3 double	unit vector towards the Sun (in inertial frame)

5.7.2 Spacecraft

Spacecraft with full transational and rotational motion propagation, without sensor and actuator model.

5.7.2.1 Description



The block combines the **Orbit Dynamics** and **Attitude Dynamics** to model the transational and rotational motions using advanced force models. (cf. description of the aforementioned blocks for further details).

5.7.2.2 Parameters and Dialog Box

Name	Description
Fundamental simulation step	sample time of the block. The value -1 is forbidden, because the sample time is used to determine if the maneuvers needs to be considered as impulsive or extended.
Epoch of the state	initial epoch of the state vector in GPS time [Weeks, Seconds of Week]
Initial state (inertial)	initial inertial state vector [m m/s]
Spacecraft initial attitude (CDM) error wrt BF	cosine direction matrix describing the spacecraft initial attitude error with respect to the body frame
Spacecraft initial angular velocity error in BF	spacecraft initial angular velocity error in body frame [rad/s]
Area	spacecraft area [m]
Mass	spacecraft mass [kg]
Drag coefficient	spacecraft drag coefficient [-]
Solar radiation pressure coefficient	spacecraft solar radiation pressure coefficient[-]
Spacecraft inertia tensor in principal-axis-frame (PA)	spacecraft inertia tensor in principal-axis-frame
Spacecraft geometrical center position in PA	spacecraft geometrical center position in principal-axis-frame
Spacecraft geometry properties in PA	spacecraft geometry properties in principal-axis-frame
Rotation matrix BF>PA	rotation matrix describing the rotation matrix from body frame to principal axes
Spacecraft effective magnetic moment [A m^2]	spacecraft effective magnetic moment [A m^2]
Gravity file	path to the file containing the model of the gravity field
Flux file	path to the file containing the F10.7, F10.7, and Kp values
Earth Magnetic Model IGRF 2010: g_data	path to the file containing the G data of the Earth Magnetic Model IGRF 2010
Earth Magnetic Model IGRF 2010: h_data	path to the file containing the H data of the Earth Magnetic Model IGRF 2010
m gravity field order	desired m-order of the gravity field
n gravity field order	desired n-order of the gravity field
Earth Magnetic Model: level of accuracy	level of accuracy of the Earth Magnetic Model
Order for the normalization of the quaternion	Order for the normalization of the quaternion
Reference frame (REF) assumed for defining the desired attitude	inertially fixed, RTN,Sun-Zenit pointing or external guidance

If "Inertial fixed", constant cosine-matrix (CDM) written in inertial-Earth-centered (ECI)	rotation matrix describing the inertially fixed frame (only used if the chosen reference frame is inertially fixed
Desired attitude: x_BF direction w.r.t. REF	axis of the reference frame to be aligned with the x axis of the body
Desired attitude: y_BF direction w.r.t. REF	axis of the reference frame to be aligned with the y axis of the body
Apply drag	selection of the force model (1: selected, 0: not applied)
Apply solar radiation pressure	selection of the force model (1: selected, 0: not applied)
Apply Sun perturbation	selection of the force model (1: selected, 0: not applied)
Apply Moon perturbation	selection of the force model (1: selected, 0: not applied)
Apply Tidal perturbation	selection of the force model (1: selected, 0: not applied)
Apply relativity effect	selection of the force model (1: selected, 0: not applied)
Gravity gradient external torque	selection of the force model (1: selected, 0: not applied)
Magnetic field external torque	selection of the force model (1: selected, 0: not applied)
Solar radiation pressure external torque	selection of the force model (1: selected, 0: not applied)
Aerodynamic external torque	selection of the force model (1: selected, 0: not applied)
Engine firings internal torque	selection of the force model (1: selected, 0: not applied)

5.7.2.3 Inputs and Outputs

Overview of the block inputs

<table>
<tr><th>Name</th><th>Format</th><th>Description</th></tr>
<tr><td>thrust_eci</td><td>1x5 double</td><td colspan="3">thrust command: vector of dimension 5 describing the control action to be applied. The first two vector elements are the start and end time of the thrust interval (in Simulink time, starting from zero at the begining of the Simulation). The remaining three elements describe the thrust vector written in the inertial frame.</td></tr>
<tr><td>torque cmd</td><td>1x3 double</td><td colspan="3">torque command [Nm]</td></tr>
<tr><td>RefSys</td><td>Simulink Bus</td><td colspan="3">Simulink bus comprising the following signals:<table><tr><td>Precession</td><td>3x3 double</td><td>precession transformation of equatorial coordinates</td></tr><tr><td>Nutation</td><td>3x3 double</td><td>transformation from mean to true equator and equinox</td></tr><tr><td>GHA</td><td>3x3 double</td><td>transformation from true equator and equinox to Earth equator and Greenwich meridian system</td></tr><tr><td>Pole</td><td>3x3 double</td><td>Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date</td></tr><tr><td>TOD</td><td>3x3 double</td><td>transformation matrix from ICRF to true of date</td></tr><tr><td>ITRF</td><td>3x3 double</td><td>transformation matrix from ICRF to ITRF</td></tr><tr><td>ITRF_dot</td><td>3x3 double</td><td>time derivative of ITRF</td></tr><tr><td>GPS_UTC</td><td>1x1 double</td><td>difference between GPS and UTC time [s]</td></tr></table></td></tr>
<tr><td>ExtAtt</td><td>Simulink Bus</td><td colspan="3">Simulink bus for external guidance mode comprising the following signals:<table><tr><td>q_eci2bfT</td><td>1x4 double</td><td>rotation from inertial to body frame to be tracked</td></tr><tr><td>w_BFT</td><td>1x4 double</td><td>angular velocity profile to be tracked</td></tr></table></td></tr>
</table>

Overview of the block outputs

Name	Format	Description		
y_eci	1x6 double	inertial state vector		
q_eci2bf	1x4 double	quaternion describing the spacecraft attitude from inertial to body frame		
w_bf	1x3 double	spacecraft angular velocity [rad/s]		
FS	Simulink Bus	Comprises the following signals:		
		q_ref2bf	1x4 double	rotation from reference frame to body frame
		q_eci2bf_des	1x4 double	rotation from inertial frame to body frame
		q_eci2ref	1x4 double	rotation from inertial frame to reference frame
		wBFdes	1x3 double	angular velocity profile to be tracked (imposed reference or selected pointing mode)
		sun_uv_eci	1x3 double	unit vector towards the Sun (in inertial frame)

Activities Conducted with the Multi-Satellite Simulator

This section comprises a non-exhaustive list of published projects or research activities already performed using the Multi-Satellite Simulator:

- **Offline and Hardware-in-the-loop Validation of the GPS-based Real-Time Navigation System for the Prisma Formation Flying Mission** [15]. The paper describes the pre-flight validation of the Prisma GPS-based navigation software using the precursor version of the Multi-Satellite Simulator. Among the different methods used for the validation, the utilization of existing flight data coming from another mission is described as well as the use of a GPS signal simulator in the loop.
- **Spaceborne Autonomous Relative Control System for Dual Satellite Formations** [13]**.** A prototype implementing the algorithms for onboard autonomous formation flying described in the paper is tested. The navigation and control performance is investigated using software models, while the computational effort is analyzed by porting the prototype to a target representative of the onboard computer of the TanDEM-X mission.
- **Relative Control of a Virtual Telescope Using GPS and Optical Metrology** [14]. This study deals with the relative control of a formation flying on a high elliptical orbit. The relative navigation is based on the GPS at the perigee passage. The testing activities have been performed using a GPS signal simulator and Phoenix receivers in the loop.
- **Hardware in-the-Loop Multi-Satellite Simulator For Proximity Operations** [11]. The paper describes the first steps realized to set up a hardware in-the-loop multi-satellite simulator aimed at the support of closed-loop real-time real-scale on-orbit servicing scenarios.

References

[1] Simulink® 7 Reference, The Mathworks, Inc. (2010).

[2] Persson S., Jakobsson B., and Gill E.; PRISMA, Demonstration Mission for Advanced Rendezvous and, Formation Flying Technologies and Sensors; Number 05-B56B07, 56th International Astronautical Congress, Fukuoka, Japan, International Astronautical Congress (2005).

[3] Moreira, A., et al., "TanDEM-X: A TerraSAR-X Add-on Satellite for Single-Pass SAR Interferometry", IGARSS, Achorage, USA; 2004.

[4] Van Helleputte T.; User manual for the GHOST orbit determination software; FDS-SUM-3110; Deutsches Zentrum für Luft-und Raumfahrt, Oberpfaffenhofen (2004).

[5] Matlab® 7 Getting Started Guide, The Mathworks, Inc. (2010).

[6] Real-Time Workshop® 7 Reference , The Mathworks, Inc. (2010).

[7] VxWorks Reference Manual, Edition 1, 5.3.1, WindRiver Systems (1998)

[8] RTEMS C User's Guide, Edition 4.6.5, for RTEMS 4.6.5, On-Line Applications Research Corporation (2003)

[9] Montenbruck, O., Nortier, B. and Mostert, S.; A Miniature GPS Receiver for Precise Orbit Determination of the SUNSAT2004 Micro-Satellite; ION National Technical Meeting, 26-28 Jan. 2004, San Diego, California (2004).

[10] Gaias G., Ardeans J.-S., D'Amico S.; Formation Flying Testbed at DLR's German Space Operations Center; 8th International ESA Conference on Guidance, Navigation & Control Systems; 5-10 June 2010, Carlsbad, Czech Republic (2011).

[11] Gaias, G., D'Amico, S., Ardaens, J.-S., and Boge, T., Hardware in-the-Loop Multi-Satellite Simulator For Proximity Operations, 11th Int. WS on Simulation & EGSE facilities for Space Programs, SESP 2010, 28-30 September, ESTEC, Noordwijk, the Netherlands (2010)

[12] EPOS – Facility Manual, Robo-Technology GmbH, Puchheim, Germany (2009).

[13] Ardaens, J.-S. und D'Amico, S. Spaceborne Autonomous Relative Control System for Dual Satellite Formations. Journal of Guidance, Control, and Dynamics, 32 (6), pp. 1859-1870. DOI: 10.2514/1.42855 (2009).

[14] Perea, L., Ardaens, J.-S., D'Amico, S., Elosegui, P., Relative Control of a Virtual Telescope Using Global Positioning System and Optical Metrology, Journal of Guidance, Control, and Dynamics, 33 (4), Seiten 1859-1870. DOI: 10.2514/1.48287 (2010).

[15] D'Amico, S., De Florio, S., Ardaens, J.-S. and Yamamoto, T., Offline and Hardware-in-the-loop Validation of the GPS-based Real-Time Navigation System for the PRISMA Formation Flying Mission, 3rd International Symposium on Formation Flying, Missions and Technology, Noordwijk, The Netherlands. (2008)