ROSETTA LANDER SOFTWARE SIMULATOR

Gábor Tróznai *(troznaig@freemail.hu)*, Attila Baksa (*baksa.attila@syncnet.hu*), Sándor Szalai (*szalai@sgf.hu*), Bálint Sódor (*simplesimon@freemail.hu*)
SGF Ltd. (Space and Ground Facilities, Budapest, Hungary)

Keywords: Rosetta, lander, space research, software, simulator, XML, C++, transputer

Abstract
The software simulator (LSS) was created for Ground simulation of the Rosetta Lander, Philae. The system consists of five personal computers and several Real-Time Message Handler cards. The simulation of the behavior of the on-board equipments is realized using XML syntax based simulation script language. During the design time of LSS the most important aspect was the high level of flexibility. Using the realized solutions it is able to implement simulation of other complex systems.

The aim of the Rosetta mission is to approach and observe the comet Churyumov-Gerasimenko. The role of the lander unit is to make studies on the surface of the comet. For the thoroughgoing measurements there are eight scientific instruments and seven subsystems integrated into the small lander unit. All of the equipments on the lander controlled by a unique developed, embedded on-board computer. The scientific operation of the lander is managed by eight application tasks of the multi-tasking operating system of this computer. The mission of the lander has two phases. The primary mission takes few days after landing onto the comet's surface. During this phase the main goal is to accomplish detailed measurements until the main batteries become flat. During the secondary phase the aim is to analyze the behavior of the comet approaching the Sun. These measurements will be performed with a lower intensity reclined upon the energy gain of the solar cells. The secondary phase is planned to take several month.

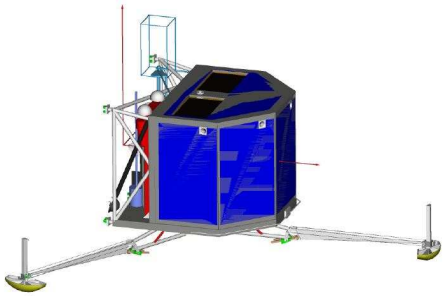

Figure 1. The lander unit of the Rosetta space-probe called Philae.

TASKS

The high level of complexity, and the long lifetime of the Rosetta space-probe necessitate to have a system, which provides an easier way to perform the following tasks all along the 10 years while Rosetta is performing the mission:

- Training of the operator staff
- Testing operational schedules
- Performing long term tests
- Performing endurance tests
- Performing data transfer tests
- Running and testing telecommand sequences
- Testing of the software of the on-board computer, especially in cases of non-nominal situations, which are dangerous to execute on the real lander unit nor the ground reference model.
- Reproduction of the events recorded from the probe.

The best solution to issue the tasks above is a software based simulation of the probe. So the SGF Ltd. in cooperation with the German Aerospace Center (DLR) has developed the Rosetta Lander Software Simulator (LSS) .

THE ENVIRONMENT OF THE LSS

All of the equipments on the Philae are connected to the central computer of the lander unit called CDMS (Command and Data Management System). The CDMS is in connection with the on-board computer of the Rosetta probe (On-board Data

Handling System - OBDH) via the Electrical Separation System (ESS) of the lander. The communication has two possible ways. During the space flight the lander and the orbiter are connected together via cable, but after separation they will use a radio (RF) connection. The orbiter has a high power radio system to keep contact with the Ground Segment which is communicating through big radio telescopes on Earth. The received and transmitted signals go through the Spacecraft Interface Simulator (SIS), and arrive to the Lander Control Centre System (LCCS). All the system elements use the Rosetta Common Packetized Protocol (RPRO) to transfer data. The LCCS receives and handles the scientific data coming from the Philae, and initiates sending of the telecommands to the lander. The structure of LSS should mirror this communication chain, and should provide authentic interface on proper points of the system. This interface has the following elements:

1. Software simulation of the on-board equipments on Philae
2. On-board computer of Philae (CDMS)
3. Software simulation of Rosetta ESS
4. Simulation of Rosetta OBDH communication interface

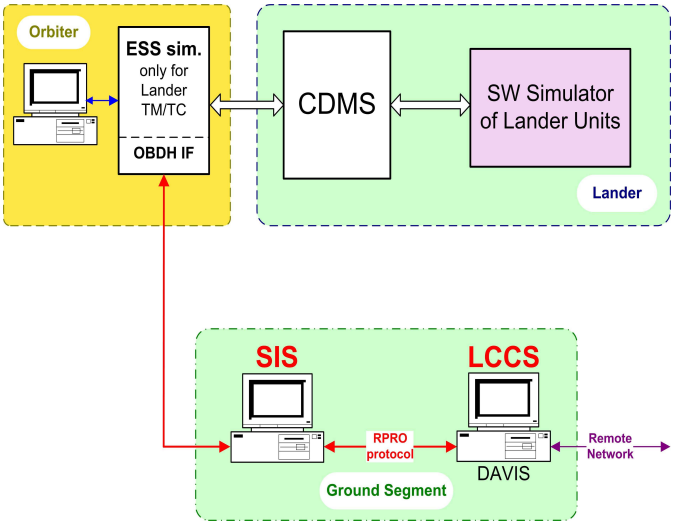

Figure 2. The environment of the software simulator

THE STRUCTURE OF LSS

The software simulator is a shared computer network, which consist of several computers. In case of the Rosetta Lander the several equipments are simulated by four portable computers. The system has a fifth dedicated computer to clamp, and synchronize the simulations, and to store the produced data stream. The low level and high speed simulations of the different equipments are accomplished by unique developed hardware elements. These are the Real-Time Message Handlers, which are connected to the computers via RS-232 serial lines. The simulator of the on-board computer of the lander unit (CDMS) is realized by a one-to-one copy of the real CDMS, because it would be very difficult to realize the simulation of the full functionality, and the real reaction time of the computer by the reason of the complexity of CDMS and it's real-time multitasking operating system. The simulator computers are connected to each other via Ethernet (TCP/IP) network, and it is possible to connect them together through the internet.

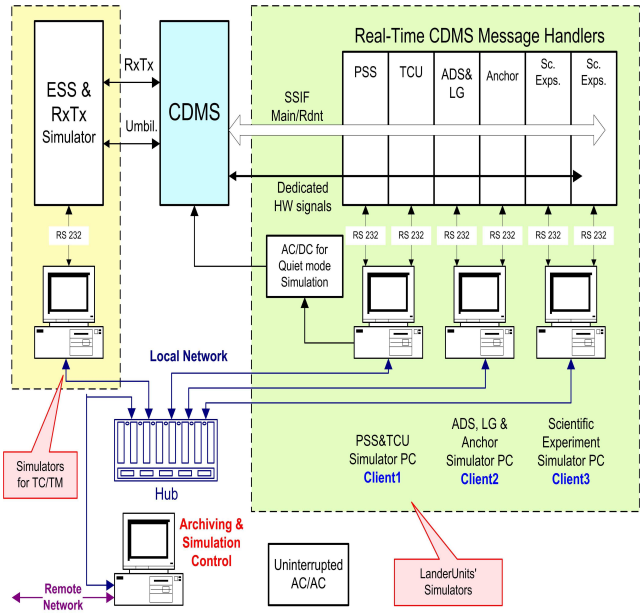

Figure 3. The structure of LSS network

HARDWARE ELEMENTS

The Real-Time Message Handler (RIU) cards are developed by the SGF Ltd. in the middle nineties. They are IBM PC card sized multifunctional signal level simulators with embedded processors based on transputers. The name "transputer" is a combination of words **trans**istor and com**puter**. These processors produced by Inmos (England) in the late eighties. Both of the 16 bit and 32 bit versions has four serial data-transfer channels to connect more processors together, and the internal architecture makes them very suitable for parallel processing. As a matter of fact these RISC processors were the first real products for parallel processing. They can be programmed using OCCAM or C languages, which support the parallel processing on advanced level. In spite of many beneficial properties, the worldwide

spread of the Intel processor family has unfortunately meant the end of the career of these type of processors. The simulator cards with embedded processors are connected to computers via RS-232 serial lines. The built-in memories on the RIU cards buffer the data for both direction of data streaming, and makes it possible to initially upload simulated data stream for real-time reactions.

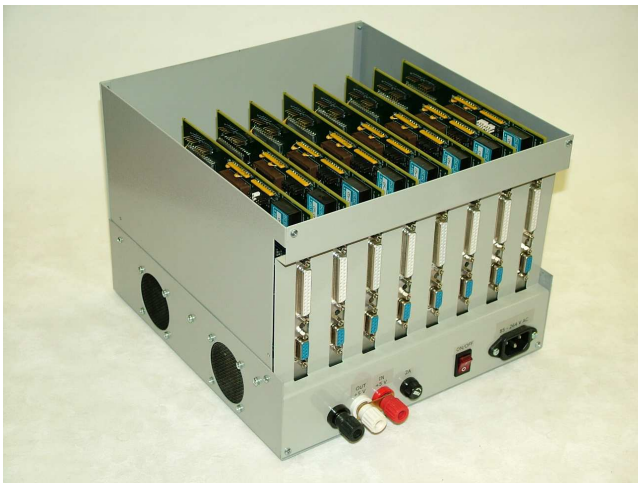

Figure 4. The Real-Time Message Handler cards

SOFTWARE ELEMENTS

The software elements of the simulator system can be classified as follows:

1. Simulation of the equipments on the Lander
2. Software elements for special tasks

The simulation of the equipments on the PhilaeLander

The simulations of the lander equipments are executed by General Unit Models (GUM) in cooperation to the Real-Time Message Handler cards. Each equipment has a GUM, but the different GUMs can be grouped, so the simulation software executed by a PC can manage more then one equipment's simulation simultaneously. The grouping can be modified freely, but in general the simulated system determines the formed groups. It is not practical to run simulations on the same PC which are requiring big amount of computing capacity. In Rosetta LSS the following groups have been formed:

1. PC:
   - Power Sub System (PSS)
   - Thermal Control Unit (TCU)
2. PC:
   - Active Descent System (ADS)
   - Landing Gear (LG)
   - Anchor
   - Sampling and Drilling System (SD2)
3. PC:
   - Scientific equipments (APX, CIVA/ROLIS, CONSERT, COSAC, MUPUS, PTOLEMY, ROMAP, SESAME)

An XML based script language was defined for describing the behavior of the on-board equipments. It is possible to make unique simulation XML file for each scientific unit and servicing subsystem. These files are validated, interpreted and executed by the General Unit Model. All of the equipment models have its own schedule which are executed in different threads. So they run the commands defined in the simulation files independently from each other. The simulation file makes it easy to describe the real operation modes of the onboard equipments and the transitions from one mode to other. The grouping of the simulation models, and their parameters can also be described in an XML based configuration file. With these files it is possible to change dynamically the composition of simulations, including the PC - equipment simulation assignments without to change the program source code. Beyond the XML syntax, the configuration and simulation descriptor files follow a script language syntax as well. The files are validated and checked by the simulator module before executing the file. Accordingly when a new element is needed to be integrated into the system, then it is enough to define the behavior in a simulation script file. The creation and usage of this script file is easy to learn so it is not requires advanced developer skills. For the deeper developing there is programming interface (API), which makes it possible to realize extremely special equipments which would be very difficult to simulate using the script language only. In these cases the developer can implement the unit in C++ language and could integrate it easily to the system by using the API library functions. Of course this way of simulation requires software developer skills. In the LSS system there is only one module called ESS-Bridge (ESS and SIS simulator) which is not using the general approach of the XML script language. The charge of this model is to simulate the behavior of ESS, which ensures the communication between the lander's central computer (CDMS) and ground simulator of the orbiter's onboard computer (OBDH and SIS) on both cable and RF connection using RTS protocol.

Software elements for special tasks

These software elements manage special tasks in the LSS system.

LSS Server

The central software element of the simulation system's TCP/IP segment is the LSS Server. All software module in the system is connected with each other through the server.
The main tasks of the server are:

- Ensure communication channels between the software modules
- Store the data generated the software modules (Server Data Pool)

The elements communicate with each other on the TCP/IP network according to a special protocol called LSS Data Interchange Protocol (LSDIP) developed especially for this system. The protocol uses data packets with a dynamically changing size called Protocol Control and Data Packet (PCDP). These packets consist of a fix sized header and a changing sized data field. Several parameters defined in the header, such as the ID of the sender and receiver modules, information about the acknowledge request, the type and subtype of the packet, special parameters according to the type of the packet, the size of the data field, and a checksum to detect data transfer errors. The modules use such packets (PCDP) to send data to other modules. One of the server's tasks is to handle and log the communication and data transfer on the channels opened by the modules.

The Server Data Pool is a central database, which stores all the data that the modules need for running their simulation tasks. Each module can read or write into this database with the right PCDPs. The server traces the changes of the data content, and can send notice about the change for the modules, which are registered in the server's "Notify on Change" service.

The structure of the database can be changed even on the fly during the execution of the simulations. There is a software module in the system called Simulation Data Pool Presentation/Editor (SDPPE) that handles the database structure.

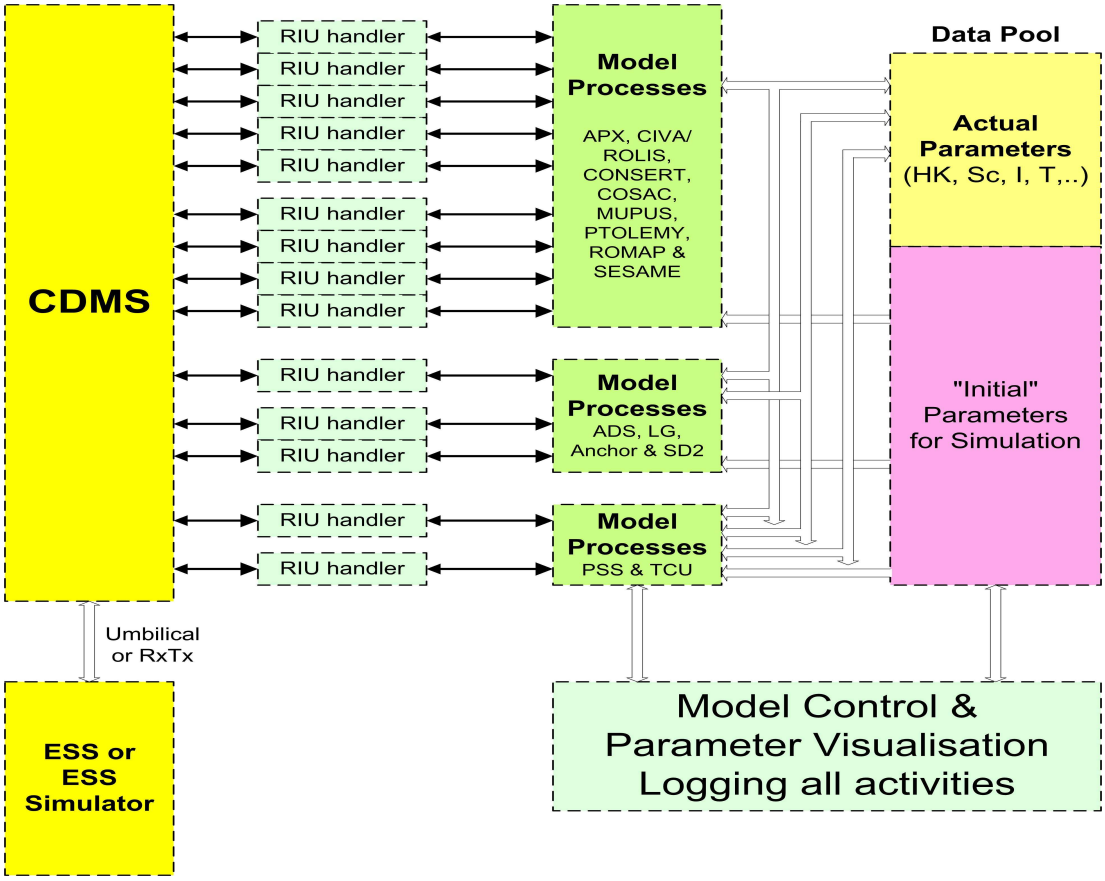

Figure 5. The internal connections between software elements

Simulation Data Pool Presentation/Editor (SDPPE)

This module supports the construction of the database structure, handles the initial values for the fields defined in the data pool. The initial values can be read from files as well. Thanks to this feature it is possible to save a snapshot about the database, and save the current state of the simulation. After reload this files the simulation can be continued at a later time. This software module provides an additional feature to view all parts of the data pool, and all data field in the data pool can be edited too. The server also allows the user to view the data content of the data pool, but the editing of the fields can be performed by this module.

Another role of SDPPE is to control (stop / suspend / start / lock data / etc.) the simulations. The unit of the stored data in the data pool is „word" (2 bytes). These words are raw data. In general case one word can store different data. For example the different bits in a word may have different meanings. In the case of spacecrafts the last eight bits of a word can store a temperature value, the next two bits can store the value of a four state signal, and the other bits can store the values of two state signals. This strategy is used to maximize the utilization of the onboard memories. In the mentioned example to get the right temperature value a mask has to be applied for the raw data. The masking provides the raw temperature data. Finally a mathematical formula has to be applied to calculate the real temperature data. This formula is a linear expression (calibration formula) in most cases. The formula and the mask are assigned to the signal, which need to be decoded. The coding and decoding raw data are managed by more modules in the system, where it is needed to see or handle calibrated data.

CDMS Memory Tool IF (CMTIF)

The role of the CMTIF execute read and write operations in the CDMS memory according to the incoming requests. This module has direct connection to the internal memory handler module of the CDMS via an RIU card. The requests can arrive from any of the modules of LSS connected to the network. The CMTIF sends back the result to the requester module on the network. It is possible to initiate similar requests from the graphical user interface of CMTIF as well.

CDMS Memory Decoder (LDEME)

There is a more sophisticated tool for displaying the memory content of the CDMS called CDMS Memory Decoder (LDEME), which cooperates to CMTIF. This module is connected to CMTIF only on TCP/IP network, receives the data sent back by CMTIF, decodes and displays the data according to the data content. With this tool the memory content of the CDMS can be easily surveyed and interpreted. The communication between these modules is managed by the server. In the present system this is the only communication which requires the flexible timeout handling implemented in the server. The reaction time of the CDMS may be relative slow, because the main task of the CDMS is not to serve the requests of the CMTIF, and LDEME. In the server can be set a parameter for each module that determines the timeout for the module's requests but the amount of the data in the reply for an LDEME request has a great variety.

Obviously the requests with bigger data size need bigger timeout, so it was necessary to implement a data size dependent dynamic timeout handling besides the fix timeout handling. With this feature it became possible to set data size dependent timeout value for different modules in place of fix timeout. In this case the server checks the amount of the requested data and sets the timeout value unique for the packet. This is the situation in the case of LDEME and CMTIF, because the CMTIF waits for the CDMS complete the request, and after receiving the whole data it sends back to LDME.

Graphical data tracer (GraphIT)

This software module visualizes the actual data stored in the data pool in a user friendly form. It is able to trace the changes of data pool sections and draw graphs to show the changes in real time. The graphs can be grouped to draw different graphs into one chart. It is freely adjustable which parts of the data pool should be displayed, and how to decode the raw data. There are modules in the system that store floating point values in the data pool with a size of at least two words. To draw one point of such a graph two words must be queried from the data pool, and must be decoded (with the calibration formula if it exists). A data pool field can also store string values. The changes of these values in time can be traced by GraphIT as well. The constructed graph groups are displayed in different windows, and the all configuration can be saved into file, and reload from file. Two types of refresh modes can be

assigned to the graphs. In the first mode the software draws a new point to the graph only when the referenced data pool area changes. In the second mode GraphIT queries the current value from the server periodically. The query period is adjustable for each graph form. The displayed data can be recorded into files for additional processing in other external programs like Excel, Matlab, etc. Instead of selecting a part of data pool it is possible to select predefined Parameter Objects (PO) from a list. For example the PO which describes a temperature value in data pool contains the accurate position of the temperature's raw data in the data pool, the mask for decoding, and the formula for the calibration.

Summary

At the design of the LSS system the flexibility was the main aspect. Besides the current application this system can be adapted to simulate other complex systems. The modular structure of the system provides the possibility for developers to work on modules simultaneously and mainly independently from each other. During a long development phase in an international cooperation like project Rosetta, it is an important advantage. The XML based script language makes it easy to define simulations without changing the source code of any programs. The creation of the simulation script files dose not require advanced programming skills, neither from the operators who are took into the project during the long term of the mission. The software elements for the special tasks are mainly independent from the simulated system. In the case of the development of difficult modules there is an opportunity to use a C++ API which supports the developers to integrate a new complex module easily into the system.