End-to-end Ground Testing Framework for Raman Laser Spectrometer (RLS) on board Exomars 2020

J. Zafra[(1)], J.Saiz[(3)], L. Seoane[(1)], C. Quintana[(2)], S. Ibarmia [(2)], C. Perez[(2)], A.G. Moral[(2)]

[(1)] Ingeniería de Sistemas para la Defensa de España, S.A. (ISDEFE)
Calle de Beatriz de Bobadilla, 3, 28040 Madrid (Spain)
Email: zafraij.pers_externo@inta.es, seoanepl.pers_externo@inta.es

[(2)] Instituto Nacional de Técnica Aeroespacial (INTA)
Carretera de Ajalvir, Km 4, 28850 Torrejón de Ardoz (Spain)
Email: quintanarc@inta.es, ibarmiahsa@inta.es, carlos.perez@cab.inta-csic.es, moralia@inta.es

[(3)] Universidad de Valladolid - Centro de Astrobiología
Calle Francisco Valles 8, 47151, Valladolid (Spain)
Email: jsaiz@cab.inta-csic.es

INTRODUCTION

Functional testing is needed in order to build the system right. Nevertheless, it is a cost- and time-consuming part of the system engineering process. Thus, in the context of a demanding schedule for the developing and delivery of a space product, automated testing acquires special importance so that cost and time can be reduced in the same way as the quality of the space product increases. The use of automated testing in AIT phases contributes in many benefits: testing, efficiency improvement, faster feedback, reusability of automated test, long term testing, fewer human resources, reliability or the possibility of automated documentation. The End-to-end Ground Testing Framework for Raman Laser Spectrometer (RLS) was born in this context. The need of tests automation to increase the effectiveness of software tests together with a demanding schedule, leads to the creation of a system to validate software requirements applicable to the embedded RLS Application Software. Once the framework basis was created, it grew with new applications to perform testing at different levels. The ground-testing framework described in the paper, is also able to support the validation of operational and scientific requirements, with a set of dedicated system tests scripts. In this way, the system allows the operator to execute Raman instrument operations. Additionally, the framework is able to simulate Raman instrument operation on ground, allowing activity plans, built as a sequence of tasks and actions, to be created and executed to handle the instrument operation by translating tasks into the corresponding sequence of actions, and actions into a sequence of telecommands.

THE CONTEXT: THE MISSION AND THE INSTRUMENT

The ESA's 2020 mission of the ExoMars programme will deliver a surface science platform and a rover carrying both European and Russian instruments. The two science stations will operate in parallel, using the communications infrastructure of the Trace Gas Orbiter (ExoMars scientific and communications orbiter module, launched in 2016). The ExoMars rover will travel across the Martian surface to search for signs of life. It will collect samples with a drill and analyse them with next-generation instruments. ExoMars will be the first mission to combine the capability to move across the surface and to study Mars at depth.

The Raman Laser Spectrometer (RLS) [1] is one of the Pasteur Payload instruments belonging to the analytical suite of the ExoMars2020´s Rover Module (placed in the rover Analytical Laboratory Drawer, ALD). It will perform Raman spectroscopy on Mars samples after the Rover´s drill has acquired bulk samples and the Sample Preparation and Distribution System has powdered and crushed their cores. The (RLS) instrument provides a powerful tool for the definitive identification and characterisation of minerals and biomarkers. Raman spectroscopy is sensitive to the composition and structure of any mineral or organic compound.

Fig. 1 shows the RLS qualification model during a functional test campaign using End-to-end Ground Testing Framework for Raman Laser Spectrometer RLS as EGSE.

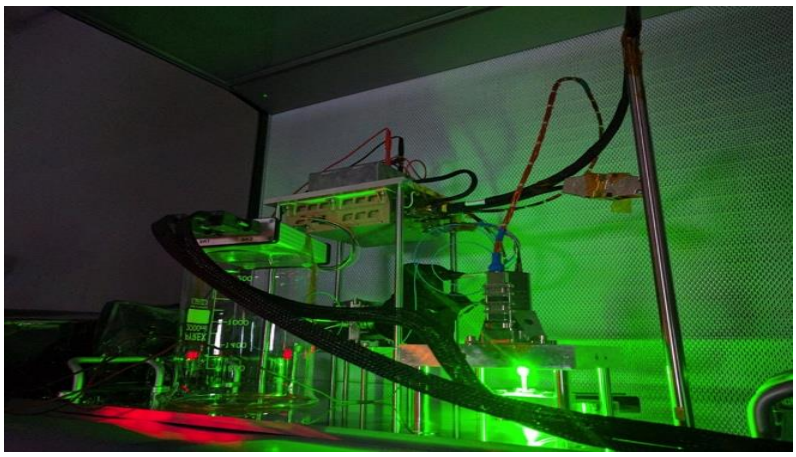

Fig. 1. Raman Laser Spectrometer (RLS)

THE BASIS OF THE SYSTEM: ROVER VEHICLE INTERFACE SIMULATOR

The basis of the system is the RVIS (Rover Vehicle Interface Simulator), a hardware/software equipment developed by Celestia and distributed by ESA to every instrument team. The RVIS supports the development of the Rover Payloads and their integration. The primary function of the RLS RVIS is to simulate the Rover interfaces to the dedicated Pasteur Payloads (PPL) in terms of:

- Power
- Telemetry and Communication (CAN)

It also serves TM packets to RLS IDAT (Instrument Data Analysis Tool), a user-friendly Specialised Check-out Equipment (SCOE).

The RVIS concept and its interfaces are depicted in Fig. 2:

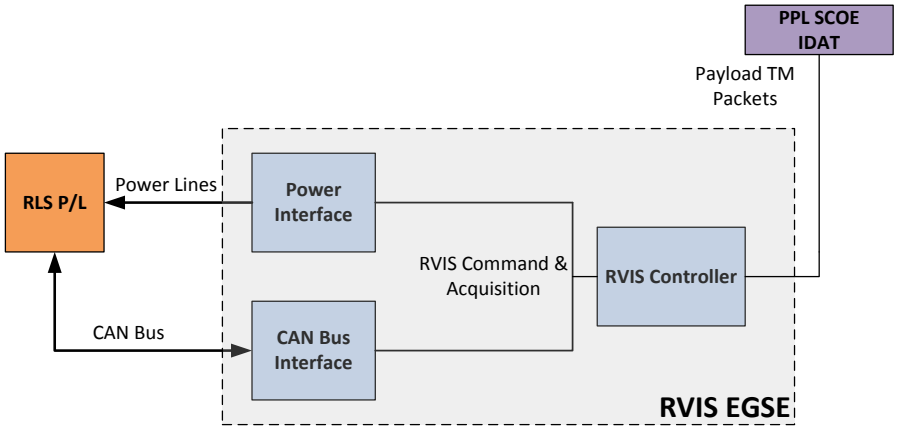

Fig. 2. RLS RVIS Concept

Rover Vehicle Interface Simulator Architecture

A high-level diagram that shows the main data flows between the elements of the RVIS EGSE is depicted in Fig 3.

TC and TM processing in the system is based on Satellite Control and Operation System 2000 (SCOS-2000) [2], thus TC and TM APIDs (packet identification file) are provided from instrument teams in a MIB (Management Information Base) file Database.

For control purposes, a distinction is made between TC packets that are destined for RLS and Command and Control messages that are used to control the operations of the RVIS EGSE. The Remote interface uses the APID of the TC packet header to determine the final destination. The Command Data Block that is encapsulated in the data field of the so-called Spacecraft TC packets will be extracted from the packet and based on the APID the packet data field content is routed to the CAN Front-End (FE) that is responsible for the actual transfer of the Command Data Block to the RLS instrument.

A Telemetry Data Block as received (or acquired) by CAN Front-End from RLS will be encapsulated in the data field of a pre-defined TM packet skeleton. For monitoring purposes of the RVIS itself the EGSE supports cyclic housekeeping updates and event driven TM packet distribution towards IDAT SCOE as well.

The baseline for the RVIS is the Control, Monitoring, Data processing and Visualisation Software (CMDVS). The CMDVS is an integrated software environment with project specific Front-Ends that allows for the implementation and operation of integrated Simulation, test and Monitoring & Control systems for Space applications.

Test Sequence Controller (TSC) is a software application developed by TERMA, allowing for the development, execution and verification of Test Scripts and referencing and acting upon TM/TC data and decoded TM parameters. This includes among other facilities for script editor, TCL (Tool Command Language), microTOPE, TM/TC Service, TM packet and parameter displays.

IDAT provides software tools for the reception, de-codification and calibration of the HK TM received from the RLS instrument as well as for the conversion of science TM into science products. In addition, it provides the basic scientific functionality to analyse these science products obtained from the instrument.

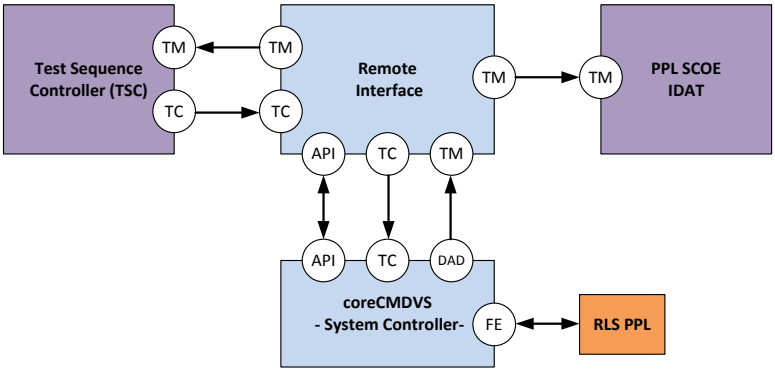

Fig. 3. RVIS EGSE architecture

TESTING FRAMEWORK ARCHITECTURE

The End-to-end ground testing framework for RLS is based on major applications managed by TCL scripts running on the TSC. Fig. 4 illustrates the overall system: The operator configures and selects the tests to be commanded in the graphical interface application "RLS RVIS Scripts App" which creates an input file (either Scripts_Input.txt or ActivityPlan_Input.txt) depending on the selection (RLS test / RLS operation) made by the operator, and executes the corresponding scripts in TSC so that the RLS test or operation starts. The operation is managed by the instructions from TSC scripts allowing CMDVS to power the instrument and exchange TCs and TMs. In the meanwhile, the operator can follow the execution using RLS IDAT to view the housekeeping or the science spectra sent by the instrument in real time.

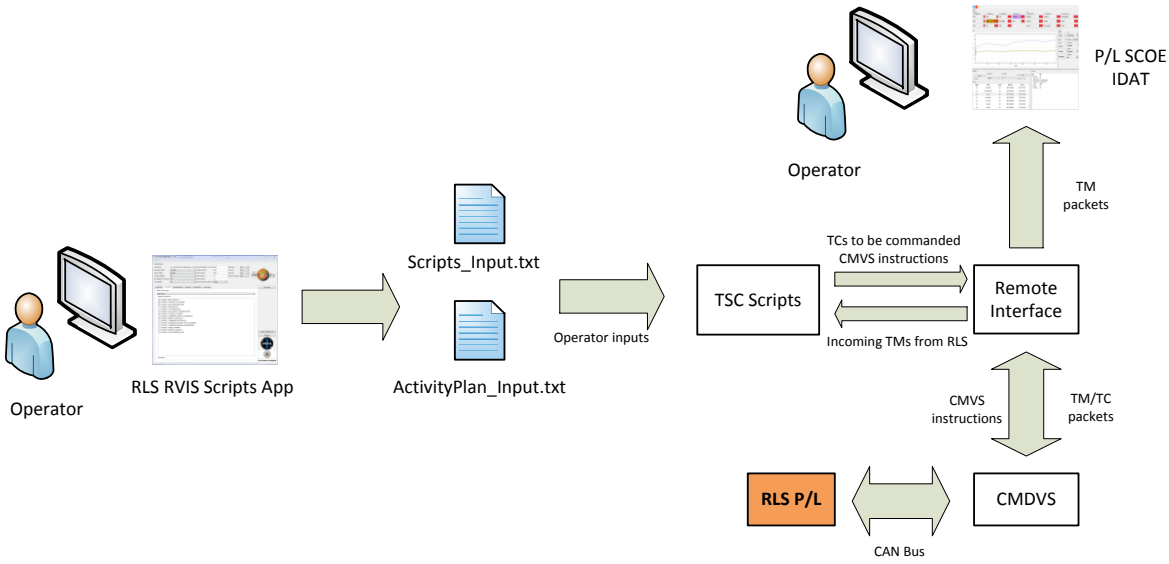

Figure 4. Testing framework diagram

COMMAND OPERATOR INTERFACES

A user-friendly Graphical User Interface lets command easily the different RLS models developed (EQM, EIS, FM, FS), change between different configurations and kind of tests (e.g. different operation parameters limits will apply, depending on testing thermal environment), and save all data generated by each test in an identifiable way.

RLS RVIS Scripts App is a software tool used for the command and control of the testing framework. This software is developed in QT/C++ and runs on any Microsoft Windows platform from Windows 7. The application allows the user to configure to select the RLS system configuration, weather to operate RLS simulating a real RLS operation in Mars via Action and Tasks or performing different type or tests/operations at SW or system level.

Fig. 5 illustrates the main window showed just after opening the executable for executing RLS tests:

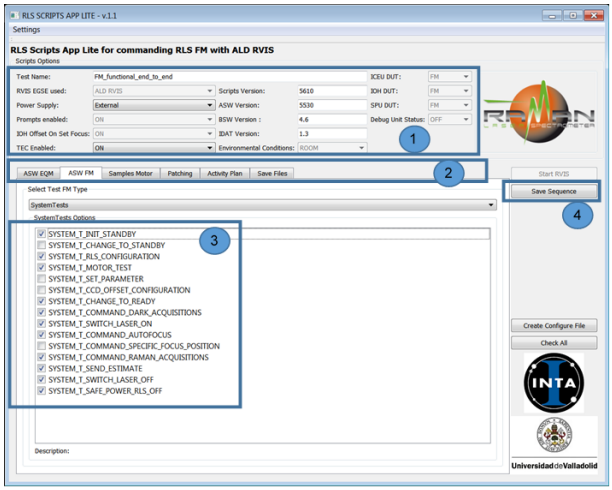

Figure 5. RLS RVIS Scripts App Main window for RLS tests

In Fig. 5, the different sections are identified with numbers:

1. Scripts Options: Used to define the test configuration.
2. Test tabs: Tabs allow the selection of a RLS test or RLS operation simulation
3. SytemTests Options: After selecting SystemTests above in "Select Test FM Type", RLS scripts are listed below so that the operator can choose the script to be executed in order (from top to bottom).
4. Save Sequence button: Operator shall click "save sequence" in order to save RLS script sequence

The testing framework system allows also the simulation of the RLS operation in Mars, creating and executing activity plans, to handle the instrument operation by translating tasks into the corresponding sequence of actions, and actions into a sequence of telecommands.

- An activity Plan (AP) is a logically related group of commandability primitives (action and tasks) set in order to be executed from the Rover OBC to operate RLS instrument.
- Action and tasks contain the sequence of the TCs to be sent, TMs to be expected and the preconditions to be satisfied.

A graphical SW tool provides the possibility of creating activity plans in a user-friendly way by adding action, tasks with their corresponding input parameters (Fig. 6). Then, already created activity plans can be loaded inside RLS RVIS Scripts App and executed by selecting "Activity Plan" tab in the App. It also provides the possibility of creating .bin files for loading RLS configuration by clicking on "create Configure File" button.

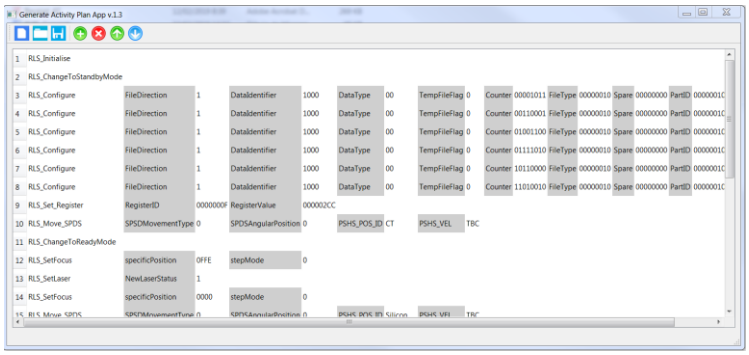

Figure 6. GUI for creating Activity Plans

TESTING FRAMEWORK SCRIPTS

The power of the system commandability is based on TCL/micro Tope scripts executed inside the TSC SW environment.

The execution environment: Test Sequence Controller

Test Sequence Controller (TSC) is a software tool developed by TERMA that allows users to connect, observe and interact with a system under test in real time. Its main goal is to allow automated testing using a test sequence scripting language. Its main application is the validation of a system under test.

The system under test is typically a spacecraft subsystem or instrument.
TSC uses the same database layout (the "MIB") to describe commands and telemetry as the one used in European spacecraft missions for mission operations and system level checkout. This means that a database validated at instrument/subsystem level can be delivered to the system level.

TSC also supports a variant of the "TOPE" test language compatible with that used by SCOS2000-based test systems.

The programming language: Tool Command Language

TCL stands for Tool Command Language. TCL is a multifaceted language package widely used for in-house packages, as an embedded scripting language in commercial products, as a rapid prototyping language, as a framework for regression testing, and for 24/7 mission critical applications

One of the strengths of TCL is the number of special-purpose extensions that have been added to the language. One of them is TOPE (Test and Operations Procedures Executive), an extension to TCL for spacecraft data handling. uTOPE is a "micro" version of TOPE specifically targeted for check-out of spacecraft subsystems. It shares many of the same commands and features as TOPE, but is lighter weight and, being a more recent implementation, offers some performance improvements and additional features. RLS scripts are developed using TCL with its uTOPE extension specially to handle TCs and TMs based on SCOS2000 standard. During this development, [3] has been a very useful guide to gain knowledge in TCL.

TSC Scripts

The high level structure of TSC scripts developed for the RSL testing frame is shown in Fig. 7:

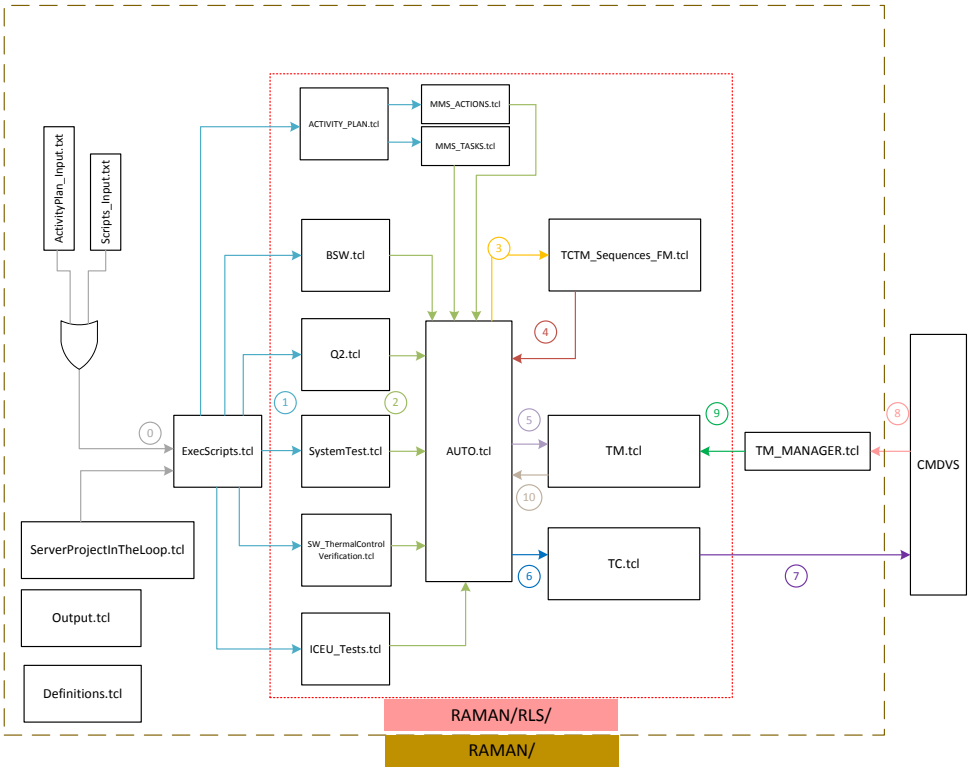

Figure 7. TSC Scripts diagram

TSC scripts are responsible for the following activities:

- Read input files generated by RLS RVIS Scripts App.
- Command CMDVS to power ON/OFF RLS.
- Command CMDVS to send TCs with the corresponding parameters.
- On board last time (OBLT) correlation between EGSE and RLS.
- Verify incoming TM parameters against expected values.
- Generate automated reports based on a sequence of delivered TCs and incoming TMs checks. The test/operation configuration will also be present in the report together with the timings for delivery and reception of each TC/TM.
- Command RLS to go to safe mode in case something goes wrong as well and sending an email reporting the error.

During the scripts execution three threads are running in parallel:

- ServerProjectInTheLoop: is in charge of calling ExecScripts thread at the beginning and also to send a TC for going to safe mode and sending an email to the test responsible if something goes wrong.
- ExecScripts: main thread. It manages the real time operation.
- TM_MANAGER: responsible of detecting every incoming TM.

As seen in Fig. 7, different steps are listed in numbers to explain the complete process from operator configuration in the UI to TCs/TMs exchange with RLS:

0. RLS RVIS Scripts App generates the corresponding .txt input files and calls ServerProjectInTheLoop to call and start the main thread in TSC: "ExecScripts", responsible of reading the input file with the test configuration to be executed and calling the second thread: "TM_MANAGER", which is responsible of detecting every incoming TM.
1. Depending on the test type selected by the operator, "ExecScripts.tcl" calls the corresponding test/operation script:
    a. "ACTIVITY_PLAN.tcl" for a RLS operation simulation.
    b. "BSW.tcl" for BSW (RLS Boot SW) verification tests.
    c. "Q2.tcl" for functional RLS verification tests at SW level.
    d. "SystemTests.tcl" for RLS functional test or operations at system level.
    e. "SW_ThermalControlVerification.tcl" for thermal RLS verification tests at SW level.
    f. "ICEU_Tests.tcl" for functional verification tests at ICEU (Instrument Control and Excitation Unit, the RLS electronic box) level.
2. Each test sequence consists of an entity of TC to be delivered with its corresponding parameters and the expected incoming TMs with their corresponding parameters in the EGSE. Each test sequence with all this information is passed to "Auto.tcl" in order to handle it in real time.
3. "Auto.tcl" calls "TCTM_Sequences_FM.tcl" in order to build a list with the information about the TC to be delivered and the incoming TMs to be expected.
4. The list with the test sequence is passed back to "Auto.tcl" as an argument. The OBC intelligence is coded in the functions of "TCTM_Sequences_FM.tcl". Different TMs are built to be expected depending on the TCs and parameters to be delivered. In the case of a RLS Operation the pseudocode from Action and Tasks is implemented in the scripts: "MMS_ACTIONS.tcl" and "MMS_TASKS.tcl".
5. The way both threads communicates each other is made with shared variables (via setshared in TCL) for the received TM data and shared counters to handle a flag for the reception. Each TM has an incoming counter shared variable with the number of already received TMs in the EGSE. Each time a new TM arrived in CMVS, TM_MANAGER increases its counter variable and saves its content in its TM shared variable, so that ExecScripts can get the incoming data to be verified afterwards. Thus, when "Auto.tcl" calls "TM.tcl", the variables of the reference counters from each expected TM are increased with the number of TMs to be received in the current test sequence.
6. "Auto.tcl" calls "TC.tcl" and pass the parameter values of the TC in the form of a list as an argument.
7. "TC.tcl" uses uTOPE functions for adapting the current TC to the instrument database and commanding CMDVS to send the TC with the corresponding parameter values of the current test sequence.
8. Once the TC has been sent via CAN, as soon as expected TM are received in EGSE, CMDV interrupts the TM_MANAGER thread execution to increase the shared counter of the corresponding incoming TM.
9. In the meanwhile, ExecScripts thread is polling the shared counter till its value is equal as the one from the reference counter set in Step 5. When that happens, the TM content in the shared variable is read and its parameter values are passed back to "Auto.tcl" as a list.
10. "Auto.tcl" verifies the expected TM parameters against the received ones and reports using "Output.tcl" the results of this verification in real time. Afterwards, the execution returns back to step 1 for every test sequence until no more sequences (TCs) are remaining.

“Output.tcl” is in charge of generating a script report in .txt format to document the configuration and the execution of the scripts in real time so that it can be followed by the operator while the scripts are running, otherwise it can be analysed afterwards.

“Definitions.tcl” contains some common functions used by more than one script in order to avoid code duplication.

MONITORING OPERATOR INTERFACES

During the test execution, the operator has different tools to follow the execution in real time:

- Script Report .txt file focuses on the test results
- CAN Bus Monitor (a SW tool provided by Celestia in RVIS) focuses on incoming TM in raw format
- IDAT a friendly UI that focuses on housekeeping and science TMs.

IDAT (Instrument Data Analysis Tool)

The RLS IDAT (Instrument Data Analysis Tool) is a software tool that can be used for the reception, de-codification, calibration and verification of the TMs (science and housekeeping) generated by the RLS instrument. IDAT will interface the RVIS as shown in Fig. 4 as a TM reception SCOE. The software is developed in QT/C++ and runs on any windows platform from Windows 7, it is developed by the “University of Valladolid”.

The aim is to provide the RLS engineering team a user-friendly tool for the analysis of the instrument health status, and the representation of the HK TM received in RLS instrument. IDAT provides the capability to analyse the TM data in “real time” (understanding this as not needing to perform post-processing for the analysis of the data). This allows the monitoring of the instrument status, operation mode, incoming TM events, temperatures and powers both in ADCs or engineering magnitudes during the instrument operation, ensuring the performance of the instrument in every test during AIT phases. IDAT also allows the operator the possibility of adding comments to the session at any time during the test execution. This is depicted in Fig. 8:

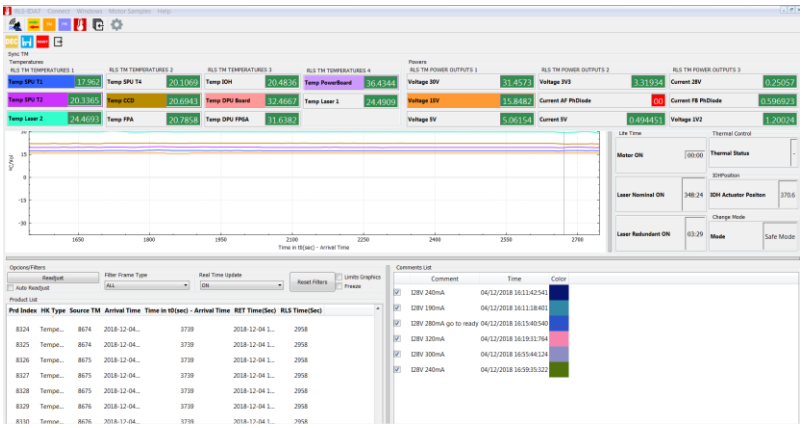

Figure 8. IDAT RLS Housekeeping data window

In addition, IDAT is also a science analysis tool. It allows the user to analyse the science products. This functionality is provided by SpectPro software, which is integrated together with IDAT SW tool. SpectPro allows the representation and processing of RLS spectra to perform a basic analysis of these products, in order to rapidly assess the spectral quality and even to verify the instrument performances against the technical specifications. This is depicted in fig. 9:

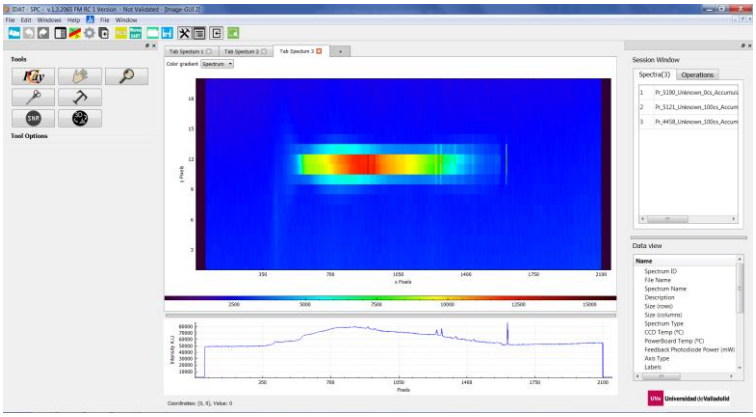

Figure 9. IDAT SpectPro linear spectrum GUI

FRAMEWORK FACILITIES

Fig.10 depicted RVIS and the facilities where the use of the testing framework takes place:

Figure 10. RLS RVIS EGSE setup

The overall RLS RVIS EGSE setup, was placed and operated into an ISO8 cleanroom, for being able to have direct and visual contact with the RLS HW, displayed into a ISO4 laminar flow booth, during the whole functional and E2E tests campaign.

CONCLUSSION

In the context of the flight segment, the end-to-end Functional Testing Framework has been a widely used powerful EGSE during the development and AIT phases of RLS instrument qualification and flight models in every functional test including: SW verification tests, unit tests, thermal tests, EMC tests, system performances tests or science test. As every operation performed by the testing framework is automatically recorded and documented, the automated testing framework brings also many advantages in terms of product quality like traceability or version control.

In addition, the functionality of activity plans execution brings many possibilities in the context of ground segment for supporting the future RLS instrument operation in Mars. Every activity plan set to be executed by the ExoMars Rover in Mars can be simulated using the end-to-end testing framework with representative models of RLS instrument. Thus, the framework is also a powerful tool for helping to prepare the daily activity plans, verify and confirm their suitability in terms of time and data budgets, as well as for the analysis and replication of problems occurred during the operation in Mars.

REFERENCES

[1] Fernando Rull et al., "The Raman Laser Spectrometer for the ExoMars Rover Mission to Mars", *ASTROBIOLOGY*, vol. 17, Num-bers 6 and 7, 2017.

[2] SCOS-2000 Team, *SCOS-2000 Database Import ICD*, Version 6.9, July 2010.

[3] Clif Flynt, *Tcl/Tk A Developer's Guide*, Third Edition, 2012