SINBAD Flight Software, the on board software of NOMAD in ExoMars 2016

M. Carmen Pastor-Morales[a], Julio F. Rodríguez-Gómez[a], Rafael Morales-Muñoz[a], Juan M. Gómez-López[a], Beatriz Aparicio-del-Moral[a], Gian Paolo Candini[a], Jose M. Jerónimo-Zafra[a], Jose J. López-Moreno[a], Nicolás F. Robles-Muñoz[a], Rosario Sanz-Mesa[a], Eddy Neefs[b], Ann Carine Vandaele[b], Rachel Drummond[b], Ian R. Thomas[b], Sophie Berkenbosch[b], Roland Clairquin[b], Sofie Delanoye[b], Bojan Ristic[b], Jeroen Maes[b], Sabrina Bonnewijn[b], Manish R. Patel[c], Mark Leese[c], Jon P. Mason[c], and The NOMAD team.

[a]Instituto de Astrofísica de Andalucía, IAA-CSIC, Glorieta de la Astronomía, 18008 Granada, Spain;
[b]Belgian Institute for Space Aeronomy, BIRA-IASB, Ringlaan 3, 1180 Brussels, Belgium;
[c]The Open University, Milton Keynes, MK7 6AA, UK.

ABSTRACT

The Spacecraft INterface and control Board for NomAD (SINBAD) is an electronic interface designed by the Instituto de Astrofísica de Andalucía (IAA-CSIC). It is part of the Nadir and Occultation for MArs Discovery instrument (NOMAD) on board in the ESA's ExoMars Trace Gas Orbiter mission. This mission was launched in March 2016.

The SINBAD Flight Software (SFS) is the software embedded in SINBAD. It is in charge of managing the interfaces, devices, data, observing sequences, patching and contingencies of NOMAD.

It is presented in this paper the most remarkable aspects of the SFS design, likewise the main problems and lessons learned during the software development process.

Keywords: ExoMars, embedded software, flight software, RTEMS, patching

1. INTRODUCTION

The Nadir and Occultation for MArs Discovery instrument (NOMAD), in the ExoMars Trace Gas Orbiter (TGO) 2016 mission, has as goal looking for trace gases, in particular methane, which could be signatures of active biological or geological process in Mars[1]. This instrument is a spectrometer suite that consists of three separate channels, Solar Occultation (SO)[2,3], Limb Nadir and Occultation (LNO)[2,3], and Ultraviolet and VISible Spectrometer (UVIS)[4].

The Spacecraft INterface and control Board for NomAD (SINBAD) is the module in charge of managing the power and the communication with the TGO spacecraft for the NOMAD instrument.

The SINBAD Flight Software (SFS) is the software implemented in SINBAD, which controls the NOMAD instrument. It main tasks have been defined as follows:

- **Manage the communication with TGO spacecraft.** These interfaces are implemented by a MIL-STD-1553B bus for telecommands and housekeeping transmission, and a Spacewire[5] bus for science data transmission.
- **Control the observing sequence of the NOMAD channels.** When an observation is commanded, the SFS sends the relative set of parameters to the channels to perform the observations and afterwards, it reads the data produced. These data are sending later to the spacecraft. SINBAD stores the observation parameters in the on board memory in Channel Observation Parameters (COP) tables.

pastor@iaa.es, phone: +34958230652, web: www.iaa.es

Software and Cyberinfrastructure for Astronomy III, edited by Gianluca Chiozzi, Juan C. Guzman, Proc. of SPIE Vol. 9913, 99133I · © 2016 SPIE
CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2230760

- **Control of system devices:** SFS controls the LNO flip mirror and the operational heaters, and reads the magnitudes of the temperature, current consumption and voltage from the system sensors.
- **Patch software and data.** SFS is able to patch or update code and data in the system memory during flight.
- **Implement the NOMAD contingency and recovery plan.** This plan is designed in order to detect problems and recover the instrument autonomously under certain circumstances.

1.1 NOMAD description

NOMAD is composed of SINBAD, the three channels (SO, LNO and UVIS), a mechanical structure and two thermal control systems. These last are a survival heater (under the control of the spacecraft) and an operational heater (under the control of SINBAD). See Figure 1.

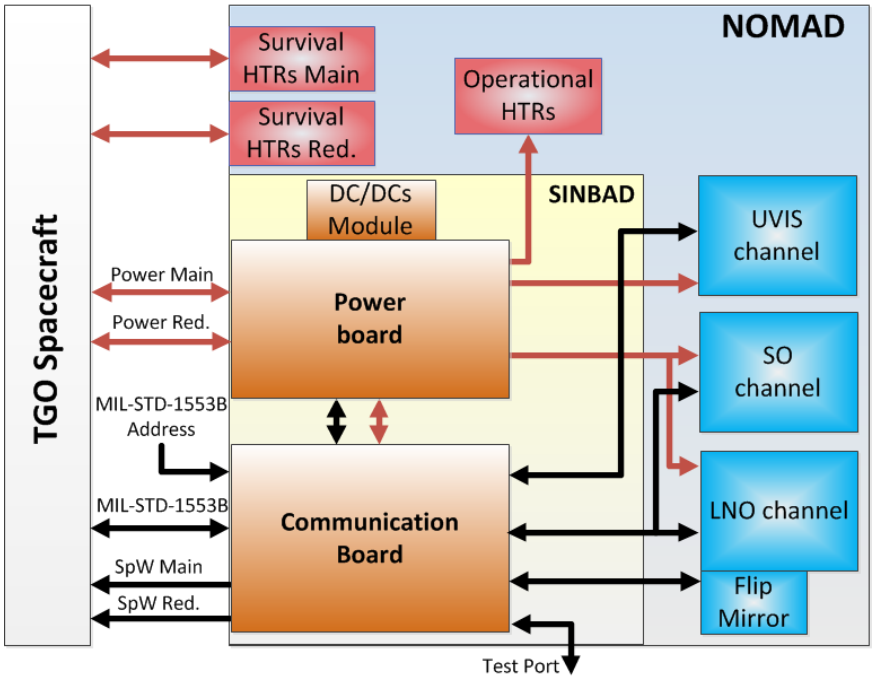

Figure 1. NOMAD block diagram

NOMAD external interfaces are used in the communication with the TGO spacecraft. There are two different communication standards for these interfaces, a MIL-STD-1553B bus and a Spacewire bus. The instrument has also an external connector that configures the MIL-STD-1553B bus address for NOMAD and a test port.

SINBAD consists of two boards, a communication board and a power board. The power board is in charge of receiving, adjusting and distributing power to all the modules of the system and the communication board implements the communications and control of the NOMAD instrument.

The communication board contains two FPGAs, a main one called Shireen and a secondary one called Chimera, see Figure 2. Shireen implements a fault tolerant IP core of the Leon 3 processor (SPARC V8 architecture of 32 bits) running at 25 MHz and different IP cores to manage several functions. The arbiter between these IP cores is a standard bus AMBA. This FPGA has also two RS-422 interfaces: one for communicating with UVIS channel and other for testing and debugging purposes. It also has a RS-232 port for the debug support unit of the Leon 3 processor (DSU). Furthermore it has a GPIO interface with 16 I/O lines used for different purposes: read the MIL-STD-1553B bus address connector, control LNO flip mirror mechanism and handle chimera error registers.

This board also has a memory bus interface with three blocks: an Electrically Erasable Programmable Read-Only Memory (EEPROM), a Magneto-resistive Random-Access Memory (MRAM) and finally the I/O module. Shireen and Chimera are connected through this I/O memory module.

Part of the I/O module is allocated for three FIFO memory areas, which are used to send telecommands and to receive science and housekeeping data from SO and LNO channels. Other part of the IO module is used to manage several system devices: LNO flip mirror (stepper motor and pin puller), sensors (read through ADCs), operational heaters, power control, chimera error and system reset registers.

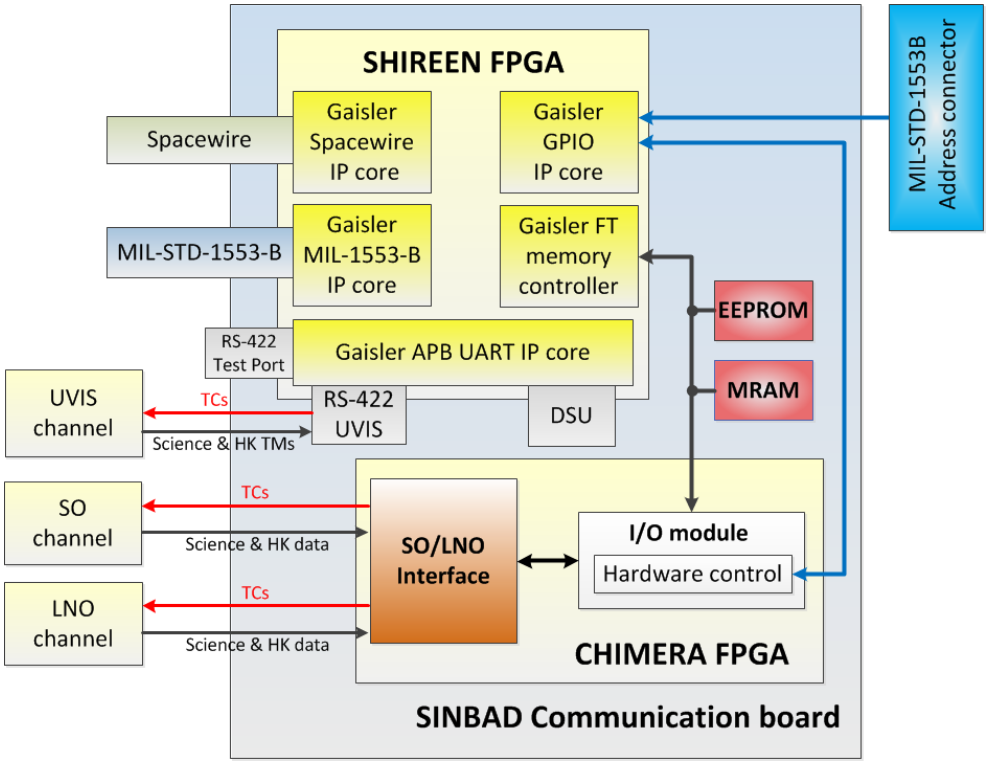

Figure 2. SINBAD Communication board interfaces

1.2 Communication between TGO spacecraft and NOMAD

To operate the instrument, the spacecraft sends telecommands (TCs) to NOMAD through the MIL-STD-1553B bus. The maximum size of a TC is 64 bytes (B) and the limit number of TCs per instrument in an operation cycle (5 days) is 750 TCs. NOMAD sends to the spacecraft data packets called telemetries (TMs). Depending of its type a TM can be sent through MIL-STD-1553B or Spacewire, see Table 1.

TCs and TMs are gathered by activities:

- **Housekeeping activities.** TMs in this activity are used to report instrument status. NOMAD housekeeping (HK) and event TMs contain information about contingencies, packets, patches, etc. Channels HK TMs include values from channels environmental information. The HK data are transmitted to the spacecraft every 30 seconds.
- **Science data activities.** TC and TMs in this activity are used to schedule and deliver science data to the spacecraft.
- **Memory management activities.** These TCs and TMs are used to update, download, patch or check data in memory.

Table 1. NOMAD telecommands and telemetries for communication with the TGO spacecraft

Housekeeping	**Bus**
TM(10) Event	MIL-STD-1553B
TM(11) NOMAD HK 1	MIL-STD-1553B
TM(12) NOMAD HK 2	MIL-STD-1553B
TM(13) NOMAD HK 3	MIL-STD-1553B
TM(23) SO HK	MIL-STD-1553B
TM(26) LNO HK	MIL-STD-1553B
TM(29) UVIS HK	MIL-STD-1553B and Spacewire
Science data activity	**Bus**
TC(20) Start operation	MIL-STD-1553B
TM(22) SO Science	Spacewire
TM(25) LNO Science	Spacewire
TM(27) UVIS Applied parameters	Spacewire
TM(28) UVIS Science	Spacewire
Memory management	**Bus**
TC(30) Patch memory	MIL-STD-1553B
TC(31) Dump memory	MIL-STD-1553B
TM(32) Dump memory report	MIL-STD-1553B
TC(33) Check memory	MIL-STD-1553B
TM(34) Check memory report	MIL-STD-1553B
TC(35) File manager operation	MIL-STD-1553B
TM(36) File manager operation report	MIL-STD-1553B
TM(37) File manager download file report	Spacewire
Mode change	**Bus**
TC(40) Safe Mode	MIL-STD-1553B
Power	**Bus**
TC(50) Ready to power off	MIL-STD-1553B
System log	**Bus**
TM(60) System log	Spacewire
General	**Bus**
TC(70) Custom command	MIL-STD-1553B

- **Mode change activity.** This TC sets the instrument in safe mode.
- **Power TC activities.** This TC is used to inform the instrument that must be ready to be powered off after one second.
- **System log activities.** This TM stores all the relevant system information.
- **General activities.** This TC is used to command special control operations: command LNO flip mirror position, fire LNO pin puller and force/unforce operational heater to power on.

1.3 NOMAD software

The NOMAD software consists of two separate programs, the Boot loader and the SINBAD Flight Software (SFS), see Figure 3.

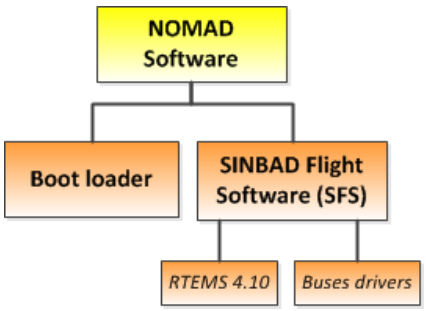

Figure 3. NOMAD software

The company Aeroflex-Gaisler provides the utility program `MKPROM2`. It is used to create boot-images for programs compiled with the C Cross-compiler `BCC` or the RTEMS cross-compiler `RCC` (`www.gaisler.com/index.php/downloads/compilers`). This program encapsulates the application in a loader suitable to be placed in a boot PROM.

The boot loader of the NOMAD software contains the `MKPROM2` code with some modification done at IAA to adapt the program to the project requirements (see Section 4.3: Errors at boot). This program is in charge of loading a SFS version to the execution area (see Figure 6: SINBAD memory map).

The SFS has been developed by the IAA, in C language, over a tailored version of the Real-Time Executive for Multiprocessor Systems RTEMS 4.10 (`www.rtems.org`), provided by Aeroflex-Gaisler. RTEMS is an open source Real Time Operating System (RTOS) that is thoroughly used in space embedded systems.

The SFS also uses some RTEMS drivers developed by Aeroflex-Gaisler: MIL-STD-1553-B, Spacewire and GPIO drivers. Part of these drivers code has been modified and adapted by the IAA to the project requirements.

2. SOFTWARE DESIGN AND DEVELOPMENT

This section describes the tools used during the software development and the most significant aspects of the software design.

2.1 Software development environment

The development environment of the SFS consists of the following tools:

- **Laboratory EGSE:** the IAA Laboratory EGSE (LEGSE) is a spacecraft simulator implemented by the IAA to test the SINBAD subsystem. It is composed of a computer that incorporates the following interfaces: MIL-STD-1553B, Spacewire, SINBAD test port and Leon DSU. Spacecraft buses interfaces were validated against a TGO simulator provided by ESA. The LGSE runs `openSUSE` 12.3, 64-bits and a software to simulate the TGO spacecraft behavior developed by the IAA. The main purposes of this software are the execution of test scripts and the storage of the generated traffic in log and data files to be analyzed. During the validation campaign the LEGSE has been used to automatize the SFS tests.
- **MIL-STD-1553B and Spacewire break-out boxes:** two break out boxes developed at IAA to analyze the buses signals.
- **Spacewire conformance tester:** this tool executes a variety of tests to check that the unit under test, SINBAD in this case, is compliant to the Spacewire standard.

- **Software development tools:** the integrated development environment `Eclipse CDT Helios` (eclipse.org/downloads/packages/heliossr2) and compilers: the LEON C/C++ IDE for Eclipse `LIDE`, the Bare-C Cross-compiler system `BCC` and the RTEMS Cross-compiler system `RCC` (www.gaisler.com/index.php/products/compilers). The debug monitor for LEON processors `GRMON` (www.gaisler.com/index.php/products/debug-tools/grmon) and the boot prom builder `MKPROM2` (http://www.gaisler.com/index.php/downloads/compilers).

- **Software quality and management tools:** the version control system `Git` (git-scm.com), the project management web application `Redmine` (www.redmine.org), the programmable tool for verification of coding standards `Vera ++` (bitbucket.org/verateam/vera/wiki/Home) and the `Source Monitor` tool (www.campwoodsw.com/sourcemonitor.html) to analyze how much code has a program and its relative complexity. To generate the code coverage of the SFS have been used the following tools: the Leon computer simulator `TSIM` (www.gaisler.com/index.php/products/simulators/tsim), the RTEMS Coverage Analysis tool `Covoar` (devel.rtems.org/wiki/TBR/UserManual/RTEMS_Coverage_Analysis) and the program for analyzing the code coverage `Gcov` (gcc.gnu.org/onlinedocs/gcc/Gcov.html).

2.2 Software architecture

This section describes the architecture used to design the SFS and its relation with hardware components, see Figure 4.

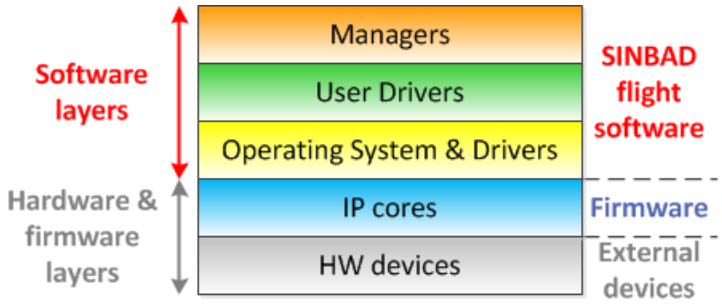

Figure 4. Scheme of system layers

The lowest layer gathers hardware devices, e.g. channels, LNO flip mirror, etc. The second layer contains the IP cores implemented in the firmware of the FPGA. These IP cores have been also considered as hardware devices.

The operating system, drivers and managers are compiled in a single executable file, the SINBAD Flight Software.

The operating system layer contains the operating system (OS) and the low level drivers. These drivers implement the logic to communicate the OS and the IP cores, providing an API to be used by the upper layers.

The user drivers layer is composed of modules that build wrapper functions for OS drivers. These units configure the high level parameters of the hardware devices.

The managers layer contains the software modules in charge of implementing the SFS high level logic. These modules utilize the user drivers from the lower layer to handle system devices. Moreover the managers implement the necessary functions to complete the behavior of the system. They are:

- **Main manager:** creates and initializes all managers in the system, stores log information about the SFS initialization, checks watchdog information and commands the first HK generation.

- **MIL-STD-1553B manager:** configures and implements the access to the MIL-STD-1553B bus.

- **Spacewire manager:** configures and implements the access to the Spacewire bus.

- **TC manager:** receives TCs from MIL-STD-1553B manager and dispatches them to relevant manager.
- **TM manager:** receives all TMs generated in the system and dispatches them to be send to the relevant bus manager.
- **HK manager:** gathers system information and generates HK, events an reports TMs. It updates HK values every 30 seconds.
- **SO/LNO manager:** handles SO and LNO channel activities.
- **UVIS manager:** handles UVIS channel activities.
- **System logger manager:** gathers messages produced by other managers. When these messages reach a defined size (or by a timeout) they are compressed and send in a TM System log.
- **File manager:** configures and implements the access to the file system.
- **Operational mode manager:** implements the relevant actions during mode changes. The operational modes of NOMAD are: Safe mode, Observing mode and Power off. This last one is not actually a mode, it is only a state where NOMAD have not power.
- **Chimera manager:** is in charge of the initialization of Chimera registers.
- **Sensor manager:** reads the NOMAD sensors values. It also detects sensor contingencies in case that some of the sensor is out of the security ranges.
- **Flip mirror manager:** handles the operation of the LNO flip mirror.
- **Contingency and recovery manager:** implements and controls the contingencies and associated recovery actions of the system.
- **Mission clock manager:** updates the mission clock of the system. The mission time is transmitted from the spacecraft through the MIL-STD-1553B bus.
- **Watchdog manager:** refreshes the watchdog register and analyses watchdog error, if any, after booting.
- **Trap manager:** implements a handler for the system traps as part of the Contingency and recovery plan of the instrument.

2.3 Observation procedure

One of the main tasks of the SFS is to manage the communication with the spacecraft and control the observation sequence of the channels. The packets involved in this procedure are dispatched by several managers, which use queues to control the data flow, see Figure 5.

The data flow from the spacecraft starts when a TC Start observation (see Table 1: NOMAD telecommands and telemetries for communication with the TGO spacecraft) arrives to the instrument. First of all it is handled by the MIL-STD-1553B manager which sends the packet to a queue in the TC manager. This manager checks the header and checksum and, if everything is correct, sends the packet to the Observation manager.

The Observation manager commands the observation tasks to the channels managers.

The channels managers analyze the commanded values and obtain the relevant parameters from Channel Observation Parameters (COP) tables (see Section 2.6: Channel observation parameters tables). During the observation, each channel manager gathers the data received from the channel, builds packets if it is necessary (this is done for SO and LNO channel, for UVIS, the SFS only checks the packet checksum) and sends them to the TM manager queue.

The TM manager analyses the headers of the packets in its queue and, depends on the telemetry type, it sends them to the HK manager or to the Spacewire manager.

The HK manager gathers the HK information and transmits it to the MIL-STD-1553B manager.

The packets are finally transmitted to the spacecraft from the buses managers.

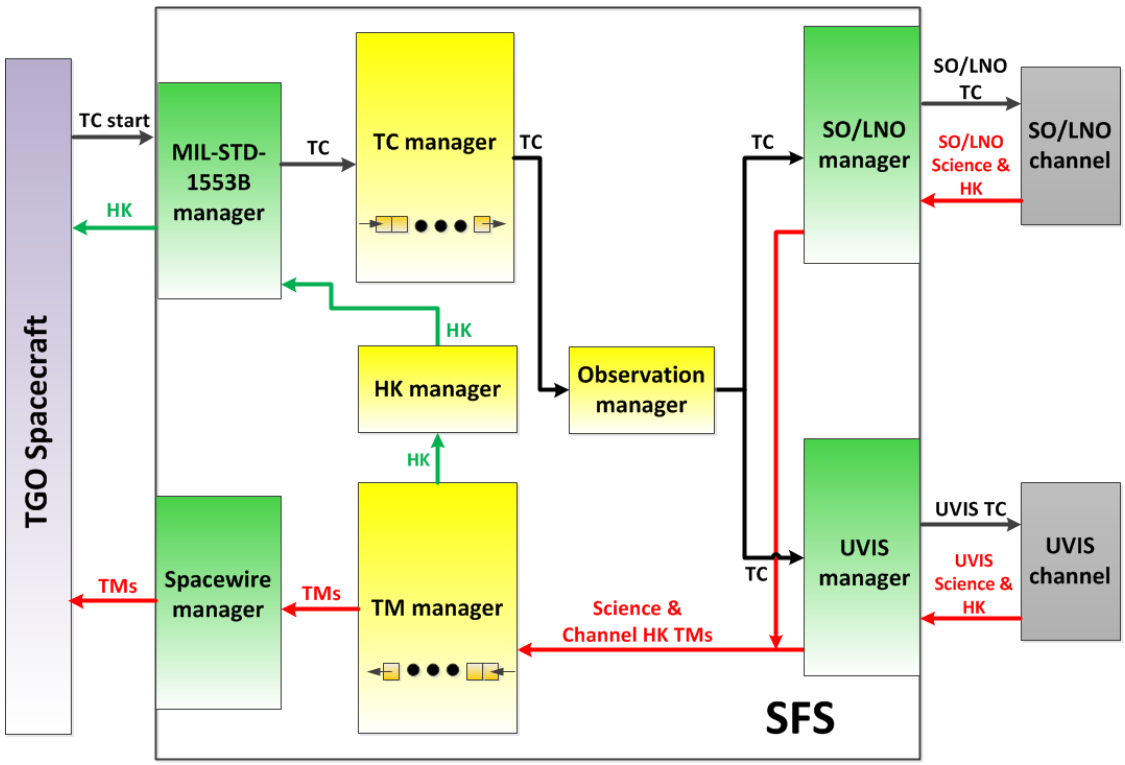

Figure 5. Observation procedure

2.4 Memory

SINBAD uses as main memory the Aeroflex 16 Megabit Non-Volatile Magneto Resistive Random Access Memory (MRAM), see Section 4.4: MRAM selection and configuration. The total size of the MRAM memory is 8 MiB. It also has an EEPROM memory of 512 KiB where permanent data are stored. The memory map is shown in the figure 6.

In first addresses is mapped the EEPROM memory. This memory contains a boot file with a compressed version of the SFS (in file `fs.prom`), one version of Context file and one version of COP tables (see Section 2.6: Channel observation parameters tables). Data in EEPROM cannot be updated during flight. These versions of data and program can execute a basic functionality of NOMAD.

Beyond the EEPROM area is placed the I/O space with Chimera Interface. Through these memory positions is established the communication between Shireen and Chimera. This space is used by the SFS to manage instrument devices as: sensor readouts, LNO flip mirror, operational heaters and the FIFO that communicates SINBAD with SO and LNO channels.

The remaining memory map accesses to MRAM. This memory is used to store the SFS version in execution and a permanent storage. The different areas are described below.

The execution area is used by the system to place the SFS version in execution (text, data, patch text and patch data sections) and the program areas heap and stack. The patch text and patch data areas are used to patching purposes (see Section 3.1: Patching the file system). Just after booting, the boot loader program cleans this area in order to avoid problems with data stored in the previous execution.

The storage area is dedicated to the massive storage of the system and contains different sets of data. In the first addresses are the boot data, which contains parameters used by boot loader. After that there are trap data; when a system trap is produced the SFS stores information in this area, this information will be available in the next run of the system to warn about the trap. Below is placed the logger data, where the system logger stores

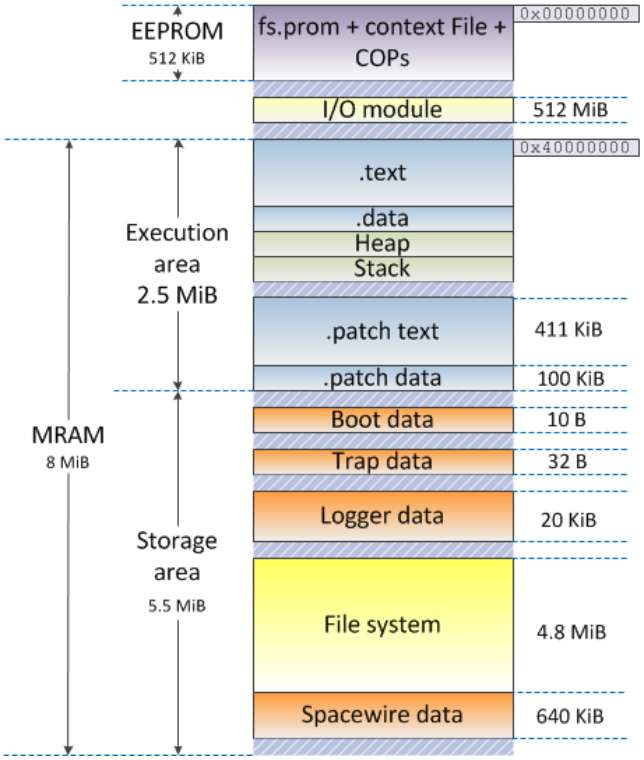

Figure 6. SINBAD memory map

messages to be sent when they fill a defined size. Afterwards there is the file system area, where the file system and all the files are stored (see Section 2.5: File system). Finally is located the Spacewire section that contains the Spacewire buffers used in the transmissions through this bus. This last area is also cleaned just after boot.

2.5 File system

The main goal of the SFS file system is to provide an easy tool to patch data and code during flight. All the updatable data are stored in system files. These files can be managed with a suite of TCs, which are designed to optimize the number of commands necessaries for the implementation of its operations during flight (see Section 4.1: Number and size of telecommands).

All file system operations are codified inside the TC File manager operation (see Table 1: NOMAD telecommands and telemetries for communication with the TGO spacecraft). The operations available are: copy file, update file, update column file, copy entire file, copy partial file, fill file with memory area, create empty file, delete last file, erase file content, reset file system, decompress file, download compressed file, print file status, print full status and print file system status.

The system flies with a list of default files previously stored in the file system (MRAM). These files are: the SFS current version files (Text, Data, Patch text and Patch data files), the Context file of the system and the files with the COPs (See Section 2.6: Channel observation parameters tables). There is also, created by default, a temporal file to be used internally by file system operations. The default files cannot be removed from the system, but their content can be updated during flight. It is also possible to store different versions of a file in the file system.

Other relevant aspect of the file system is the operation Reset file system. If the file system was completely un-recoverable, this operation resets the complete file system and builds again the default files from data stored in EPPROM. Hence the system can have a complete file system restored from EEPROM executing a single TC to the instrument. Also it is possible to get from the EPPROM the data of a single file using the operation Fill file

with memory area. Note that file system operations are not autonomously actions, they should be commanded from ground.

The file system has a general checksum and each file a single checksum. When the content of a file changes, both checksums, of the file and file system, are recalculated and updated. The checksum of files is checked by the SFS during initialization and before accessing to a file. With these mechanisms it is possible to detect a corruption in the file system.

2.6 Channel observation parameters tables

The NOMAD channels need a suite of parameters to perform each observation. UVIS uses 31 B of data parameters and the SO and LNO channels can use up to 232 B per observation. Taking into account the restrictions in TC size (64 B per TC) and the low rate of TCs allowed by the mission (750 TC every 5 days, see Section 4.1: Number and size of telecommands), it is not feasible to send all these parameters from ground, due to the fact that there was not enough margin to command other instrument operations. Because of that the system implements the Channel Observation Tables (COPs). These tables contain rows with multiple combinations of observation parameters. With these structures, only one TC is necessary to command a complete observation for a couple of channels. This is implemented by the TC Start operation (see Table 1: NOMAD telecommands and telemetries for communication with the TGO spacecraft), which only contains the pointer to the row that stores the relevant parameters for the current observation.

NOMAD has 13 COP tables, 6 for SO channel, 6 for LNO channel and 1 for UVIS channel, with a total space of 220 KiB. The COP tables are stored in files in the file system. This feature gives the opportunity to patch COP values easily using the relevant file system operations. Moreover, there is other copy of the tables (with default values) stored in EEPROM memory to be used for recovery purposes (see Section 2.5: File system). In this way NOMAD is highly flexible in order to configure observations.

2.7 Debug procedure

The debug procedure in the NOMAD instrument is done in ground gathering the information provided by HK, events, reports and the System log (see Table 1: NOMAD telecommands and telemetries for communication with the TGO spacecraft).

The HK, events and reports TMs are sent through the MIL-STD-1553B bus. The HK TMs contain information about readouts of the instrument sensors, devices status, statistics of the use of buses, etc. The event TMs reports contain asynchronous information, for instance mode changes or LNO flip mirror movements, and also error events like TC not allowed, contingency detected or file system errors. The reports TMs inform about the system status after applying an operation, e.g.: the TM File manager operation report informs about the status of a file or file system after executing a TC File manager operation.

The TM System log, sent through the Spacewire bus, contains a complete report about the system activities. It is built by the System logger manager that gathers information messages from all the managers in the system. The System log TM includes information about all the TCs processed, the TMs generated and in general each significant event that takes place during the SFS execution.

In case of one of the buses (MIL-STD-1553B or Spacewire) is not working, some debug information can be obtained from the TMs received through the other bus.

2.8 SFS versions

NOMAD has at least two versions of the SFS stored in the system, one in EEPROM and other in the file system (MRAM). After a nominal powering on, the system loads the SFS version stored in file system. A nominal power on is done if the instrument was nominally powered off in the previous run, this is when NOMAD receives a TC Ready to power off at least one second before being powered off (see Table 1: NOMAD telecommands and telemetries for communication with the TGO spacecraft). If this TC is not received in the previous execution, the system loads the SFS version stored in EEPROM. This mechanism is implemented to avoid loading a corrupted or erroneous SFS version and to have the opportunity to correct it. The load of the SFS from EEPROM can

only happen due to unexpected events: emergency switch off commanded from spacecraft or because a reset of the system.

In May 2014 was delivered to ESA the Electrical Interface Model (EIM) of NOMAD. It had the SFS version 0.1.0 stored in EEPROM and in the file system. This last one was patched to version 0.1.1 in January 2015. This model is completely representative of the instrument interfaces (relative to power consumption, electrical interfaces, data volume, TCs processing, patching, etc.). The channels behavior was simulated by the SFS to produce the flight representative data volume, due to the fact that the channels hardware was not present in this model.

In March 2016, the NOMAD Proto Flight model (PFM) inside TGO spacecraft began its flight with SFS version 2.0.0 in EEPROM and version 3.1.0 in the file system. The version 2.0.0 was record in April 2015; thereafter the EEPROM memory of that model was not accessible for updates anymore due to the integration of the NOMAD instrument in the TGO spacecraft. Afterwards the file system version was updated to 3.0.0 in July 2015 and finally, in January 2016, it was patched to version 3.1.0. The SFS version 2.0.0 located in EEPROM has a size of 402.8 KiB and the version 3.1.0 in the file system has a size of 400.7 KiB.

The company NTE-SENER was in charge of the Software Product Assurance tasks during the SFS development since the Critical Design Review (CDR) stage. They have controlled that the SFS fulfills the mission requirements and it complies with the relevant ESA standards[6,7].

3. PATCHING AND UPDATING

The patching and updating aspect is a very relevant task in the SFS. Every kind of data (software code, COP tables, Context file, etc.) can be patched during flight using TCs. The SFS implements two methods, depending of the data location, one to patch the file system files using file system operations (the nominal method) and the other to patch directly data out of the file system.

3.1 Patching the file system

The files in the file system (MRAM) can be patched or updated using file system operations such as: append to file, update file, update column file, copy file, etc. (see Section 2.5: File system), hence the SFS version stored in file system can be updated with these operations.

Upload a complete SFS version can take around 3000 TCs (with data compressed), to reduce the number of necessary TCs an extra solution has be implemented. Each SFS version is divided in four files: Text, Data, Patch text and Patch data. The Text and Data files contain the current version of text and data sections and the Patch text and Patch data are reserved to store the sections of patched code. Hence, in order to change the code of a function, it is necessary to upload the new code function to the Patch text and Patch data files and update the calling to the function in the Text file. This process reduces dramatically the number of necessary TCs (patches) to implement a change in a function.

3.2 Patching memory directly

Also it is possible to patch memory addresses directly using the TC Patch memory. A complete patch and check sequence can be commanded combining this TC with others like the TC Dump memory or the TC Check memory (see Table 1: NOMAD telecommands and telemetries for communication with the TGO spacecraft). This patching method is only recommended to modify areas out of the file system (see Section 2.4: Memory).

4. PROBLEMS AND LESONS LEARNED

This section summarizes some of the problems and restrictions that have emerged during the SFS development process and the solutions adopted to solve them.

4.1 Number and size of telecommands

The TCs received through MIL-STD-1553B bus to command the instrument have a limited size of 64 B. Each NOMAD TCs contains 6 B of protocol, so the useful number of bytes is 58 B per TC. Also there is an important restriction in number of TCs per day, 750 TCs are available per operating cycle, having an operating cycle a maximum of 5 days.

In NOMAD, the actions that require a greater number of TCs are observations and patching or updating files. In the case of the observations, the SFS implements the COP tables (see Section 2.6: Channel observation parameters tables) that were designed to avoid send all the observation parameters using TCs. In this way the TC to start an operation only includes pointers to COP rows; hence a complete observation can be commanded with a single TC. This system also makes the observation procedure more flexible.

Regarding of the files patching or updating, there is some features in the SFS that reduces the number of TCs needed. In the case of the code files, there is the patching method using the Patch text and Patch data files described in Section 3.1: Patching the file system. Moreover, if it is necessary to upload a complete file, it can be sent compressed to the file system and afterwards decompressed in the target file.

4.2 Autonomous Contingency and recovery plan

In the ExoMars TGO mission the instruments data are only monitored from ground. This means that the instruments shall operate autonomously without interaction from the spacecraft, therefore the Contingency and recovery plan shall be managed autonomously at instrument level. Because of these reasons, the SFS implements a Contingency and recovery plan designed to detect and handle directly the problems discovered during NOMAD operation. In general terms the SFS reacts to: buses problems, instrument sensors out of range, channels data generation errors, files corruption, system errors, processor traps and problems with LNO flip mirror movements.

4.3 Errors at boot

As it was previously mentioned, the NOMAD boot code has been generated using the `MKPROM2`. This tool includes a default trap handler to dispatch all traps. NOMAD boot loader implements a different handler for Error Detection And Correction (EDAC) traps. This handler looks for the failing memory address and writes its content to 0. It is implemented to prevent EDAC errors in memory when the boot loader reads address before writing, because this behavior can provoke a continuous reset of the system.

Other modification was included because of the use of the MRAM memory. The advantage of this memory is data are persistent without powering the system, but it can be also a disadvantage if some rubbish data remains in the memory after an error. To avoid that, the boot loader cleans the complete execution memory area (see Figure 6: SINBAD memory map) and also the Spacewire descriptors area to have both zones completely clean before loading the SFS.

Other issue to take into account is the default value of the timer register used by the watchdog. NOMAD boot loader code also configures the timer associated with the watchdog system to enable watchdog reset during boot. Without it the system could remain halted after a problem during boot.

4.4 MRAM selection and configuration

The first approach to the memory selection was to use a Flash memory as storage space. But one of the mission requirements asserts that instruments shall not rely on any command from TGO spacecraft to entry in safe mode when the power input is going to be switched off. This means that in non-nominal cases the instrument can be switched off without previous warning. Hence the Flash memory was discarded, because this kind of memories can be permanently damaged if the power supply is interrupted during an access operation. In spite of that it was selected a MRAM. It is the first time that an MRAM is used in a European flight mission.

One of the problems of using the MRAM was that, even though the fact that the MRAM chips manufacturer defines that data are persistent in memory with temperature range from -40°C to 105°C, tests with flight models of NOMAD have experimented a data corruption with temperatures below -20°C. This problem was solved increasing to one the wait states of the memory during readings, which adds one clock cycle to the read operations.

5. CONCLUSIONS

NOMAD stores the observation parameters in COP tables in the system memory. With these structures the SFS is capable of handling a great number of parameters combinations to command the channels receiving a single TC per observation.

The SFS code and COP tables can be easily and safety patched during flight, with a set of defined operations to modify files in file system. These operations have been designed to comply with the mission restriction about the rate of telecommands per instrument.

The SFS detects and recovers autonomously a set of defined problems to protect the instrument until receiving a reaction commanded from ground.

In conclusion, the SFS complies with the NOMAD instrument requirements, and it is able to change its behavior programming new functionality or correcting new errors detected. These modifications can be updated in the system with the patching methods previously described.

6. CURRENT STATE AND FUTURE WORK

The SFS 3.1.0 is the last version issued. This version was updated in NOMAD PFM on January 2016, two month before flight and just after instrument integration in the TGO spacecraft. This last updating process was especially significant because it made use of the complete communication chain with TGO, it means, commanding the patching TCs through the spacecraft communication system.

ExoMars TGO mission was launched in March 2016. Up to now, all the operational actions relatives to the NOMAD instrument have been completed with successful results.

It is programmed that the ExoMars TGO arrives at Mars in October 2016. Afterwards there are planned unless two years (probably four) of scientific operations with the on board instruments. During these phases the SFS can be updated or patched to provide: improvements in the NOMAD functionality, optimization in the current instrument behavior and corrections for new errors discovered.

ACKNOWLEDGMENTS

The NOMAD instrument is led by the Belgian Institute for Space Aeronomy (BIRA-IASB, Brussels), assisted by Co-PI from the IAA team (CSIC, Granada).

NOMAD has been made possible thanks to funding by the Belgian Science Policy Office (BELSPO) and financial and contractual coordination by the ESA Prodex Office. The IAA participation is funded by MICIIN through Plan Nacional Ref. AYA2009-08190 and AYA2012-39691. The UVIS support is provided by the Open University.

Thanks to all engineering, scientific and supporting staff who have worked in NOMAD project. Special thanks to the BIRA-IASB and Open University teams for providing a valuable feedback during the SFS development.

The NOMAD Team: Science Team: Vandaele A.C.; López-Moreno J.J.; Bellucci G.; Patel M.; Allen M.; Altieri F.; Aoki S.; Bolsée D.; Clancy T.; Cloutis E.; Daerden, F.; Depiesse C.; Fedorova A.; Formisano V.; Funke B.; Fussen D.r; García-Comas M.; Geminale A.; Gérard J.C.; Gillotay D.; Giuranna M.; Gonzalez-Galindo F.; Ignatiev N.; Kaminski J.; Karatekin O.; Lefvre F.; López-Puertas M.; López-Valverde M.; Mahieux A.; Mason J.; Mumma M.; Neary L.; Neefs E.; Renotte E.; Robert S.; Sindoni G.; Smith M.; Thomas I.R.; Trokhimovsky S.; Vander Auwera J.; Villanueva G.; Whiteway J.; Willame Y.; Wilquet V.; Wolff M. - **Tech Team:** Alonso-Rodrigo G.; Aparicio-del Moral B.; Barzin P.; BenMoussa A.; Berkenbosch S.; Biondi D.; Bonnewijn S.; Candini G.; Clairquin R.; Cubas-Cano J.; Delanoye S.; Giordanengo B.; Gissot S.; Gómez-López J.M.; Jerónimo-Zafra J.M.; Leese M.; Maes J.; Mazy E.; Mazzoli A.; Meseguer J.; Morales-Muñoz R.; Orban A.; Pastor-Morales M.C.; Perez-Grande I.; Ristic B.; Robles-Muñoz N.; Rodríguez-Gómez J.; Saggin B.; Samain V.; Sanz-Andres A.; Sanz-Mesa R.; Simar J.-F.; Thibert T.

REFERENCES

[1] Daerden, F., Vandaele, A. C., Lopez-Moreno, J. J., Drummond, R., Patel, M. R., and Bellucci, G., "Science objectives of the NOMAD spectrometer on ExoMars Trace Gas Orbiter," in [*EPSC-DPS Joint Meeting 2011*], 1300 (Oct. 2011).

[2] Thomas, I. R., Vandaele, A., Robert, S., Neefs, E., Drummond, R., Daerden, F., Delanoye, S., Ristic, B., Berkenbosch, S., Clairquin, R., Maes, J., Bonnewijn, S., Depiesse, C., Mahieux, A., Trompet, L., Neary, L., Willame, Y., Wilquet, V., Nevejans, D., Aballea, L., Moelans, W., Vos, L. D., Lesschaeve, S., Vooren, N. V., Lopez-Moreno, J.-J., Patel, M. R., Bellucci, G., and the NOMAD Team, "Optical and radiometric models of the nomad instrument part ii: the infrared channels - so and lno," *Opt. Express* **24**, 3790–3805 (2016).

[3] Neefs, E., Vandaele, A. C., Drummond, R., Thomas, I. R., Berkenbosch, S., Clairquin, R., Delanoye, S., Ristic, B., Maes, J., Bonnewijn, S., Pieck, G., Equeter, E., Depiesse, C., Daerden, F., Ransbeeck, E. V., Nevejans, D., Rodriguez-Gómez, J., López-Moreno, J.-J., Sanz, R., Morales, R., Candini, G. P., Pastor-Morales, M. C., del Moral, B. A., Jeronimo-Zafra, J.-M., Gómez-López, J. M., Alonso-Rodrigo, G., Pérez-Grande, I., Cubas, J., Gomez-Sanjuan, A. M., Navarro-Medina, F., Thibert, T., Patel, M. R., Bellucci, G., Vos, L. D., Lesschaeve, S., Vooren, N. V., Moelans, W., Aballea, L., Glorieux, S., Baeke, A., Kendall, D., Neef, J. D., Soenen, A., Puech, P.-Y., Ward, J., Jamoye, J.-F., Diez, D., Vicario-Arroyo, A., and Jankowski, M., "Nomad spectrometer on the exomars trace gas orbiter mission: part 1 - design, manufacturing and testing of the infrared channels," *Appl. Opt.* **54**, 8494–8520 (2015).

[4] Vandaele, A. C., Willame, Y., Depiesse, C., Thomas, I. R., Robert, S., Bolsée, D., Patel, M. R., Mason, J. P., Leese, M., Lesschaeve, S., Antoine, P., Daerden, F., Delanoye, S., Drummond, R., Neefs, E., Ristic, B., Lopez-Moreno, J.-J., Bellucci, G., and Team, N., "Optical and radiometric models of the nomad instrument part i: the uvis channel," *Opt. Express* **23**, 30028–30042 (2015).

[5] Division, E. S. E.-E. R. . S., [*ECSS-E-ST-50-12C Space engineering. SpaceWire Links, nodes, routers and networks*], The European Cooperation for Space Standardization, Noordwijk, The Netherlands (2008).

[6] Division, E. S. E.-E. R. . S., [*ECSS-E-ST-40C Space engineering. Software*], The European Cooperation for Space Standardization, Noordwijk, The Netherlands (2009).

[7] Division, E. S. E.-E. R. . S., [*ECSS-Q-ST-80C Space product assurance. Software product assurance*], The European Cooperation for Space Standardization, Noordwijk, The Netherlands (2009).