

This is the author's copy of the publication as archived with the DLR's electronic library at <http://elib.dlr.de> . Please consult the original publication for citation, see e.g. <https://ieeexplore.ieee.org/document/10115791>

Fault Detection, Isolation and Recovery in the MMX Rover Locomotion Subsystem

J. Skibbe and E. Aitier and S. Barthelmes and M. Bihler and G. Brusq and F. Hacker and H.-J. Sedlmayr

In any mechatronic system, faults can occur. Likewise also in the MMX rover, which is a wheeled rover mutually developed by CNES (Centre national d'études spatiales) and DLR (German Aerospace Center), intended to land on Phobos. An essential part of the MMX rover is the locomotion subsystem which includes several sensors and eight motors actuating the four legs and the four wheels. In each of these components and their interfaces, there is a possibility that faults arise and lead to subsystem failures, which would mean that the rover cannot move anymore. To reduce this risk, the possible faults of the MMX locomotion subsystem were identified in a FMECA study and their criticality was classified, which is presented in here. During this examination, the criticality was graded depending on different mission phases. With the help of this study, the hardware, firmware and software design were enhanced. Further, certain fault detection, isolation and recovery strategies were implemented in the locomotion firmware and software as well as in the full rover software.

Copyright Notice

©2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

J. Skibbe et al., "Fault Detection, Isolation and Recovery in the MMX Rover Locomotion Subsystem," 2023 IEEE Aerospace Conference, Big Sky, MT, USA, 2023, pp. 1-9, doi: 10.1109/AERO55745.2023.10115791.

Fault Detection, Isolation and Recovery in the MMX Rover Locomotion Subsystem

Juliane Skibbe¹
Juliane.Skibbe@dlr.de

Elise Aitier²
Elise.Aitier@cnes.fr

Stefan Barthelmes¹
Stefan.Barthelmes@dlr.de

Markus Bihler¹
Markus.Bihler@dlr.de

Gabriel Brusq²
Gabriel.Brusq@cnes.fr

Franz Hacker¹
Franz.Hacker@dlr.de

Hans-Juergen Sedlmayr¹
Hans-Juergen.Sedlmayr@dlr.de

¹ German Aerospace Center (DLR)
Robotics and Mechatronics Center
Münchener Str. 20
82234 Weßling
Germany

² Centre national d'études spatiales (CNES)
18 avenue Edouard Belin
31401 Toulouse
France

Abstract—In any mechatronic system, faults can occur. Likewise also in the MMX rover, which is a wheeled rover mutually developed by CNES (Centre national d'études spatiales) and DLR (German Aerospace Center), intended to land on Phobos. An essential part of the MMX rover is the locomotion subsystem which includes several sensors and eight motors actuating the four legs and the four wheels. In each of these components and their interfaces, there is a possibility that faults arise and lead to subsystem failures, which would mean that the rover cannot move anymore. To reduce this risk, the possible faults of the MMX locomotion subsystem were identified in a FMECA study and their criticality was classified, which is presented in here. During this examination, the criticality was graded depending on different mission phases. With the help of this study, the hardware, firmware and software design were enhanced. Further, certain fault detection, isolation and recovery strategies were implemented in the locomotion firmware and software as well as in the full rover software.

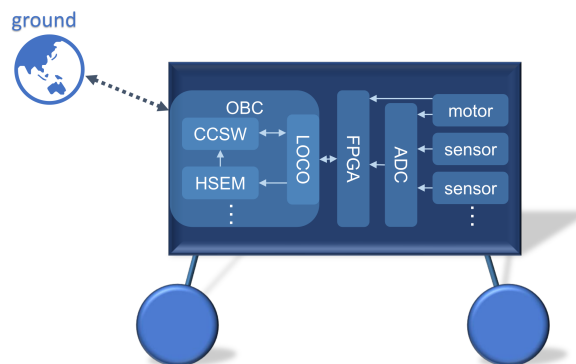


Figure 1: Overview of the communication interfaces that are relevant for the MMX locomotion subsystem.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. MISSION PHASES.....	4
3. FAULT DETECTION IN THE FPGA	4
4. FAULT DETECTION IN THE SOFTWARE	5
5. FAULT RECOVERY IN THE SOFTWARE	6
6. FAULT MANAGEMENT ON ROVER LEVEL.....	7
7. TESTING	8
8. CONCLUSION	8
REFERENCES	8
BIOGRAPHY	8

1. INTRODUCTION

The goal of JAXA's Martian Moons eXploration (MMX) mission is to scout both Mars moons Phobos and Deimos. For the larger of the two moons, Phobos, it is planned to deploy a rover, jointly developed by CNES and DLR [1]. This rover serves to explore the surface of Phobos, which is up to now unknown. Despite - or rather particularly because of - the milli gravity on this moon, namely around $0.004 - 0.007\text{m/s}^2$, the rover is designed with four fully rotatable legs and four wheels, each actuated by one motor. The locomotion subsystem, which is developed at the Robotics and Mechatronics Center (RMC) at DLR, also includes sensors

which are necessary for monitoring and traceability of the rover behavior. These sensors include:

- one commutation sensor (Hall sensors) per motor
- one absolute position sensor (potentiometer) per leg
- one torque sensor per leg
- current sensors
- voltage sensors
- four 3-axis-accelerometers
- two single-axis-gyroscopes
- numerous temperature sensors
- a radiation sensor.

Analog sensor values are digitalized by A/D converters (ADC) to make them usable for the FPGA (Field Programmable Gate Array). The FPGA is forwarding the digital values to the locomotion software on the on-board computer (OBC). The locomotion software (LOCO) can then forward the sensor data, but also already detected faults, to the Command- and Control Software (CCSW) and the Hardware- and Software-Events-Manager (HSEM). An overview of this communication path is depicted in Fig. 1.

The sensor data from the motors are monitored in the FPGA, but also forwarded to the locomotion software. The sensor values are evaluated and monitored on the OBC in the locomotion partition. This software partition - hereinafter called LOCO - is also responsible to forward the sensor- and motor data via higher level software to ground as housekeeping data.

To evaluate faults across the MMX locomotion subsystem,

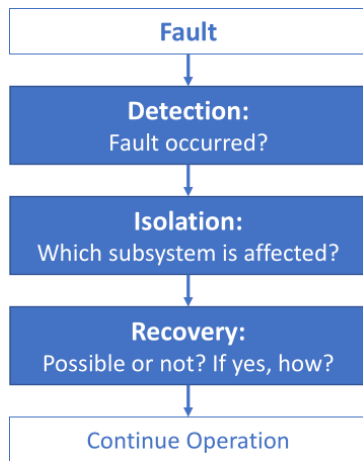


Figure 2: The FDIR principle.

the FMECA approach was used, which is explained in the remainder of this chapter. The faults have different impacts in the distinct mission phases, which are described in the second chapter. The third chapter presents the fault detection in the firmware, followed by the fault detection in the software in the fourth chapter. The fifth and sixth chapter explain the fault recovery in the locomotion subsystem and the full rover software, respectively.

FDIR

The design of a complex system like the MMX rover is a challenging task. Even with highly sophisticated design processes, the existence of faults is unavoidable. Fortunately, not every existing fault evolves into a system failure. A systematic practice to handle such faults is commonly referred to as FDIR (Fault Detection, Isolation and Recovery). This methodology attempts to prevent faults from turning into failures. Fig. 2 shows the FDIR principle.

Of course, the addition of FDIR mechanisms in the hardware or software of the rover leads to a more complex system. Either additional hardware elements are needed or software algorithms have to be added in order to provide the outlined mechanism. The fault detection and isolation is a reactive subsystem, which is triggered for instance by an abnormal operating state or odd sensor values. During the fault processing, multiple scenarios are feasible to overcome this exceptional operating state. The range measures from a simple rejection of sensor values up to a switch over to the redundant system path, if available. In general, all FDIR measures are added in order to increase the reliability and availability of the system. An overview of FDIR methodologies in space application is well explained in [2].

FMECA

For a space project, the existence of FDIR mechanisms is mandatory. In 1966, a procedure for failure mode, effects and criticality analysis (FMECA) was developed and first deployed for the NASA's Apollo program [3]. Every project which is in line with the ECSS (European Cooperation for Space Standardization) standards shall implement a dependability assurance by means of a systematic process. More information about these requirements can be found in the ECSS-Q-ST-30C standard [4] which is provided by the ESA - ESTEC (European Space Research and Technology Centre) division.

Table 1: Severity numbers, taken from [5]

Severity level	Severity category	SN
1	Catastrophic	4
2	Critical	3
3	Major	2
4	Negligible	1

Table 2: Probability levels, taken from [5]

Level	Limits	PN
Probable	$P > 1E-1$	4
Occasional	$1E-3 < P \leq 1E-1$	3
Remote	$1E-5 < P \leq 1E-3$	2
Extremely remote	$P \leq 1E-5$	1

One possibility of such a systematic process is the so called FMEA (Failure Mode and Effects Analysis) and the FMECA (Failure Mode, Effects and Criticality Analysis) which is an extension of the FMEA. Both processes are performed to identify potential failures in products and processes. The difference between both is in the classification of the identified failures. The FMEA classifies the failures according to the severity of their consequences while the FMECA classifies the failures according to their criticality.


Since the FMEA / FMECA are very generic processes, instructions on how such an analysis can be performed for the MMX rover locomotion system are not available. Only guidelines, which topics should be considered, could be found. One document which provides some basic information on the FMEA / FMECA is the ECSS-Q-ST-30-02C standard [5]. Therefore, in the following the most important details of the performed FMECA in the MMX project are highlighted.

As already stated, the FMEA and FMECA classify failures according to the severity with respect to the criticality. Therefore, a scale must be defined and agreed by all involved parties. Fortunately, the standard ECSS-Q-ST-30-02C provides solutions, which are typically used without modification. Table 1 shows the definition of the severity numbers "SN" which are defined in reversed order to the severity level. The severity level is defined from "Catastrophic" down to "Negligible". The meaning of these levels are described more in detail in the standard.

Similarly to the severity levels, the probability levels are defined. The meaning of the numbers "PN" go from "Probable" down to "Extremely remote"(see table 2).

Finally, the criticality number is calculated as the product of the severity number and the probability number. As shown in table 3, an unacceptable risk is defined by a criticality number greater or equal to 6, or if the severity is classified as "Catastrophic".

These definitions according to the ECSS standard are the baseline for the table which is part of the FMECA process in the MMX locomotion subsystem explained below.

	Design (Hardware) FMECA		FMEA Owner Eigener t.b.d.	Project Project MMX-LOC	Equipment Gerät t.b.d.	Participants Teilnehmer t.b.d.
	Potential Failure Mode & Effect analysis Fehler Möglichkeits u. Einfluss-Analyse		Date Datum Start of the FMECA	Model Modell t.b.d.	Operational Mode Betriebszustand t.b.d.	Version Änderungsstand (see modification log)

Number Laufende Nr.	Module Modul	Function Funktion	Failure Mode möglicher Fehler	Failure cause Fehlerursachen	Failure effects (local/System) Auswirkung (lokal/System)	Present state / Derzeitiger Zustand		Probability NUMBER	Criticality NUMBER	Remarks / Recommendations Kommentar / Empfehlungen		
						Severity NUMBER	Failure detection / observable symptoms Fehlererkennung / Symptome					
1	Modul	Function 1	Failure 1	Cause 1	S	3		3	9			
				Cause 2	I	2		1	2			
			Failure 2	Cause 1	S	2		3	6			
				Cause 2	S	2		1	2			
			Failure 3	Cause 3	S	1		3	3			
				Cause 1	S	2		1	2			
		Function 2	Failure 1	Cause 1	S	3		3	9			
				Cause 2	I	2		1	2			
			Failure 2	Cause 1	S	2		1	2			
				Cause 2	S	2		1	2			
			Failure 3	Cause 3	S	2		1	2			
				Cause 1	S	2		1	2			
			End of Module "Modul"									

Figure 3: FMECA table for the MMX locomotion subsystem.

Table 3: Criticality matrix, taken from [5]

Severity category	SNs	Probability level			
		10 ⁻⁵	10 ⁻³	10 ⁻¹	1
		PNs			
		1	2	3	4
catastrophic	4	4	8	12	16
critical	3	3	6	9	12
major	2	2	4	6	8
negligible	1	1	2	3	4

The FMECA process was initiated in parallel to the hardware and software design process. This ensures that last minute modifications in the hardware or in the software are avoided which are based on upcoming possible critical failures. These modifications are cost effective when considering their long-term benefits with respect to the relatively light efforts. Since the rover is a mechatronic system, where mechanics, electronics and software have to work together, the FMECA table covers all these domains. This leads to a seemingly large table on the one side, but on the other side, effects which trigger failures in different domains compared to the root cause can be monitored in a very effective way. Figure 3 shows a screen copy of the used FMECA table which was designed based on the needs of the MMX project by DLR-RM.

For a better overview and traceability, each module in the FMECA table was identified by a module ID, see Fig. 3. These modules are either a complete subsystem, a sub module or a software module. Furthermore, all important information like failure mode, cause and effects as well as the numbers and the failure handling are provided for each entry.

The whole FMECA followed a bottom-up approach. Depending on the module, the function could be a simple procedure in the software like “read sensor data” or a more elaborate function like the motor driver which is an electronic part on one of the PCBs (Printed Circuit Boards). Every module

can have several functions. The next step is the definition of possible failure modes, their root causes and whether the failures have only local effects or system wide effects, which is usually more critical.

After these preliminaries, the next step is the core of the FMECA which assesses the failures and its probability. Based on these values the criticality numbers were calculated. Then, for each possible failure, where the criticality was greater or equal to 6 (cf. table 3), a second run was performed, where the failure recognition and countermeasures were examined in detail. During this second run multiple changes in the design were found and defined, which lead to a decrease of the severity level and therefore of the criticality value.

As an example, let’s look at a sensor that measures the temperature of the drive train electronics in order to avoid damage due to overheating. Possible failures are part failure, damaged harness or broken solder link. Even if the probability level is given by “2” (remote), the criticality was set to “3” (major) due to the very important role of the drive train electronics for a rover. Finally, these numbers lead to a criticality number of “6”, which means that a second run was necessary.

The only possible failure detection within the locomotion subsystem is given by value comparison, since the listed defects lead to invalid temperature levels (extremely low or high). If a second, redundant temperature sensor was available for monitoring of the drive train, the criticality could be reduced to “1” (negligible), which leads to a criticality of “2”. Finally, the outcome of this example was a modification of the circuits and some additional items in the software target specification. Since the FMECA was performed during the design stage, these modifications could be realized without serious effort.

FMECA Results

Since the FMECA was performed in an interdisciplinary way, interactions across numerous interfaces were found and defined. This led to important modifications of the hardware that were necessary in order to enable certain failure detection and recovery capabilities of the software. In addition, failures which have to be propagated to the system controller due

Table 4: FMECA metrics of the locomotion subsystem

10	identified modules
440	possible failures in total
69	entries with a criticality number equal to 6
0	entries with a criticality number > 6
27	entries (out of the 69) are in the responsibility of the system controller
8	entries (out of the 69) are avoidable by QM and testing

to severity or criticality, were defined, too. To give some numbers for the MMX locomotion subsystem: In total 440 items are listed in the FMECA table which are grouped in 10 modules. Each of them has at least one subgroup which are listed in the function column and at least one possible failure mode per subgroup. After optimization of the failure detection and countermeasures, only 69 entries with a criticality number of “6” out of 440 entries are remaining. No higher values of the criticality number was determined. For 27 of the 69 critical entries, countermeasures were identified. These 27 items are related to the system controller. Hereby the locomotion subsystem is able to detect the failure, but the root cause for these failures (e.g. mission supply voltage) is not in the responsibility of the locomotion subsystem. For 8 of the remaining 42 entries, a sophisticated quality management (QM) and testing phase is able to detect and overcome them. Finally, 34 possible failures with a criticality number of “6” have to be served by the system. Whereby a power cycle of the locomotion subsystem, which is requested by the subsystem itself after such an failure was detected, could recover the subsystem from the failure state. Table 4 shows the numbers at a glance.

Finally, the FMECA process, which was performed in parallel to the development process, helped to develop a lean system with increased failure tolerance. The effort for the whole FMECA process was limited to some additional team meetings and minor modifications of existing hardware or software modules.

2. MISSION PHASES

The rover mission consists of several phases, the most important ones from the perspective of the locomotion subsystem are the cruise phase, the separation, landing, uprighting and deployment phase (SLUD), the locomotion checkout and the science phase. In the context of FDIR, these mission phases influence the different levels of failure notifications and recovery measures.

Cruise phase

During the cruise phase, several passive and active health checks of the locomotion subsystem are planned. During these health checks, pre-programmed sequences are run and all sensor values are recorded and sent to ground. If an error occurs during this phase, there is in general no need for autonomous recovery. The potential error is instead analyzed in depth on ground by the engineers of the subsystem to be sure to understand all possible causes of that failure. This ensures the best possible failure analysis since there is enough time until the next health check or the start of the SLUD

phase.

Separation, Landing, Uprighting, Deployment (SLUD)

The rover itself only communicates with the main spacecraft, which has altering communication slots to the MMX rover and the earth. Due to these slots, a round-trip communication between the control center on earth and the rover can take more than a day. Unfortunately, the rover’s battery can only be recharged after the SLUD sequence, which involves the uprighting process of the rover with the locomotion subsystem. Therefore, it must be ensured that the rover and therewith the locomotion subsystem recovers autonomously without ground interaction from all faults whenever possible. If a fault leads to failure of the subsystem, this would automatically lead to the end of the mission. This phase thus has the highest demands for the FDIR of the subsystem.

As an example, if a motor does not move as intended, that would usually result in a fault state, see section 3. However, during the SLUD phase, this problem needs to be tackled by the autonomous initiation of the stuck recovery mode.

Post-SLUD phase

Once the solar array is deployed, the battery can be recharged and a downtime of multiple days is not as critical anymore as during the SLUD phase. Therefore, a lower level of autonomy is foreseen in these phases. Faults can be assessed on ground rather than having to autonomously tackle them on the rover since much more information is available on ground and more sophisticated decisions can be taken.

In the example from above, the stuck motor is probably not recovered autonomously, since other problems like a faulty sensor might have led to the fault as well. However, even during the post-SLUD phase, autonomous decisions might be needed in certain situations, e.g. if the rover is in a pose which would not provide enough power generation of the solar arrays.

This phase consists of the locomotion checkout and the science phase, in which the locomotion subsystem is used for driving and to enable scientific measurements. Both sub-phases are handled similarly, although in the checkout phase, extreme care is taken when it comes to the commands. Only very short and slow drives are executed since it is unclear at that point how well the rover driving works on the Phobos surface.

3. FAULT DETECTION IN THE FPGA

Whether a fault can be detected on the FPGA or needs to be detected by higher control layers depends on several factors. Some faults have to be detected at the motor control loop frequency of 40kHz, e.g. high motor currents, and therefore cannot be monitored by the LOCO software which is operating at a 10 Hz control loop frequency. Other faults cannot be detected by FPGA firmware due to lack of information, e.g. during cruise phase the legs and wheels are not free to move. Additionally, the FPGA is limited in its resources and not all features that could be implemented in the FPGA firmware actually fit into it.

Error detection

For each motor, a motor controller is implemented by the FPGA firmware operating at a control loop frequency of

Table 5: Errors monitored by the FPGA firmware grouped by their source and monitoring frequency

Module name	Error
Power inverter chip	overtemperature warning
	fault
Motor current	overcurrent
Hall sensors	Hall sector sequence
	invalid Hall signal
Motor controller	sector generator skipped
	sector generator invalid
	controller saturation
	theta overflow
Communication	timeout
	wrong packet count

40 kHz. A detected error causes the motor controller to go to the state FAULT in which the motor is stopped. In the following, the errors detected by the motor controller are described in detail.

Motor faults have to be detected on the FPGA since the control loop frequency between FPGA and OBC is at 10 Hz. This is too slow for motor faults, since they might be detected either too late or missed entirely and inflict damage to the locomotion subsystem or the whole rover as a result. A list of all error flags monitored within the FPGA is shown in table 5.

The power inverter chip, the motor current measurement and the Hall sensors can be monitored by the motor controller implemented by the FPGA firmware. Those errors are detected by monitoring sensor inputs, such as the motor current or the Hall sensor signals. For example, the motor controller can decode the Hall sensor signals and derive a motor position. If there is an unexpected sequence of Hall signals, the Hall sector sequence flag is raised. Another group of errors are internal errors of the motor controller. For example, if the motor cannot match the commanded movement speed, the controller saturation error flag is raised.

While the faults described above are individual for each motor, the communication errors cause all motors to go to the state FAULT. First, there is an error flag for a communication timeout. If no successful communication cycle between the OBC and the Locomotion E-Box has been performed for a configured amount of time, this error flag is raised. Second, a packet counter keeps track of received communication packets. If the internal packet counter does not match the received packet identifier, the error flag for a wrong packet count is raised. In both cases, all motor controllers revert to the state FAULT and stop their motors. This behavior prevents unintended movements that might inflict damage to the locomotion subsystem or the rover. Additionally, if there has been no successful communication for a longer period of time, a reset of the FPGA firmware can be performed by means of an external watchdog chip. This allows to recover e.g. from an FPGA firmware internal deadlock and to reestablish a connection with the OBC.

Fault Flag Mask

An essential feature of the FPGA firmware is the fault flag mask. It allows to identify invalid sensor signals and then ignore them by using a fallback control mode. For example, failure of the Hall sensors of one motor would stop operation of the motor. However, due to the FPGA's firmware capability to detect such failures, the motor controller can be commanded to ignore (i.e. to mask) the faulty Hall sensor signals, e.g. invalid signal or sector sequence, and use a fallback control mode. Fault flag masks have to be set by the LOCO software.

4. FAULT DETECTION IN THE SOFTWARE

The LOCO software reacts to the fault flags raised by the FPGA and also monitors the sensor values to detect and isolate sensor faults. Furthermore, due to the control allocation calculations in the LOCO software, it can also detect geometric risks like wheel collision. There are two different approaches to fault detection: either the monitoring is continuous, or active depending on the function which is executed. For example, the wheel collision only has to be monitored when the motors are actuated.

Monitoring of the FPGA fault flags

The motor statuses are continuously monitored. When a motor is in fault mode, depending on the fault flag, a soft reset of this motor is initiated. If the motor stays in fault mode, a ground loop to examine the fault on earth is initiated.

Not all motor fault flags shall be active the entire time. For example, the "invalid Hall signal" is not relevant if the motor is in feedforward mode (= open loop mode). In the same way, also other motor fault flags are ignored or only active when certain activities are commanded.

All switches and command bytes such as the fault flag mask and resets can be configured via a telecommand to LOCO.

Monitoring of the sensor data

There are two basic types of A/D converter (ADC) signals: ratiometric signals and absolute signals.

Ratiometric signals are in a fixed ratio to the reference voltage of the ADC. With the ratiometric signals, the input signal is derived from a supply voltage that has a fixed ratio to the reference voltage of the respective ADC. The absolute value is therefore not critical and the raw value of the converter provides a valid result regardless of fluctuations in the supply voltage. In the MMX locomotion subsystem, this category includes the temperature sensors, torque sensors, rover angular rate sensors (gyroscopes) and joint position sensors (potentiometers). If a fault in the reference voltage is detected, this has to be considered for all other ratiometric sensors.

For absolute signals, the absolute value is of interest. In the MMX locomotion subsystem, this category includes current measurements, voltage measurements and accelerometers. With these signals, a deviation in the reference voltage leads to an incorrect measured value. The actual value of the reference voltage must therefore be determined for these signals. The raw values from the ADC must then be corrected

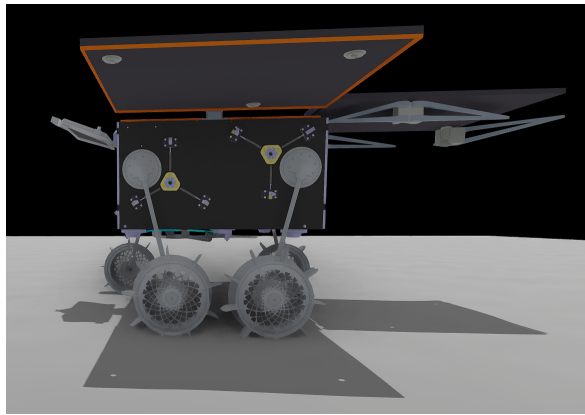


Figure 4: Wheels close to a wheel-wheel collision.

mathematically in order to get a valid absolute value, which is done in the LOCO software. The reference voltage of an ADC is determined by measuring a known voltage with the ADC.

Wheel collision detection

There are different kinds of collisions that could happen:

1. collision of one wheel against another
2. collision of the wheels against the solar panels

The calculations of these two collisions are explained in more detail below.

All possible wheel collisions have to be monitored after the uprighting of the rover was performed and while the motors are actuated. During uprighting, no collision of a wheel against a solar panel is possible since they are still retracted.

The leg positions are measured with two redundant sensors each: a motor Hall sensor and a classic potentiometer. The hall sensors provide the most exact values for a relative leg position, but in the case of an E-Box restart, they are zeroed. The potentiometer measures the absolute leg position, see also [6]. The sensor values are compared in the software and if both sensors indicate different values with a certain deviation, the hall sensor is taken into account since it is more reliable.

Wheel-wheel-collision—Since the legs have only one rotational degree of freedom, a wheel can only collide with the other wheel on the same side - right side or left side. First, the distance of the two wheels hubs on one side is calculated, respectively, by the knowledge of the leg angle. If that distance is lower than two times the wheel hub plus a safety distance, then a warning is thrown.

With the help of the leg commands, a future wheel position is calculated which can be used to predict if the wheel will soon collide. If this would be the case, an error is reported and the movement is interrupted.

Solar-array-wheel-collision—After the uprighting of the rover and the deployment of the solar panels, the collision of the wheels against the solar panels is detected simply by comparing the current leg position with the allowed leg position. In the marginal position as depicted in figure 5, the

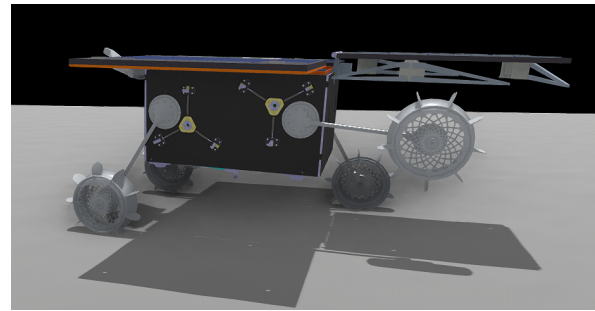


Figure 5: Wheel close to the plane of the solar arrays.

wheel is not very close to an actual collision, but the range of movement for the leg is sufficient and the risk that the rover loses its balance is reduced.

5. FAULT RECOVERY IN THE SOFTWARE

The software was designed to be as robust as possible. Hence, autonomous recovery was built in only where necessary. Here again, one has to distinguish between the different mission phases. For example, the potentiometer values are not used and the thresholds for the torque sensors to stop the motors are higher during the SLUD phase. Afterwards, in most cases, the locomotion system will wait for the next ground loop before continuing its motions. Possible recovery telecommands could be a motor reset, a full software reset or initiating an algorithm for stuck recovery of the motors. A reset of the LOCO software, handling the temperatures or restarting the power supply for the locomotion E-Box can be handled by the higher-level software and is explained in the next section.

Recovery of the FPGA fault flags

The fault flags raised in the FPGA were explained in section 3. As the FMECA process made clear, all these faults can have different sources, hence a human interpretation of these faults on ground is preferable.

Handling the sensor data

The sensor data are treated differently, depending on the type:

motor current—Since the motor currents need to be monitored in the frequency of measurements, they are already observed on the FPGA. Their fault flags are monitored by LOCO and in case of a fault, movements are stopped and internal recovery will be started.

torque sensors—If the torque is too high, the movement will be stopped and a ground loop is requested.

gyroscopes and accelerometers—The gyroscope and accelerometer measurements are used during separation and landing, which will be analyzed on ground. It is not planned to monitor them on-board.

potentiometers—The potentiometer per motor is compared with the related motor Hall sensor. In the case a hall sensor fault was raised by the FPGA, the potentiometer values are used to estimate the leg position.

radiation monitor—It is not planned to evaluate the radiation monitor on-board.

temperature—If the temperatures are too low or too high, an event will be sent to HSEM (Hardware and Software Events Manager, see next section) to demand for a heat-up or for waiting to cool down.

voltages—If the voltages are not in range, an event will be sent to HSEM to demand for a restart of the power supply.

Leg positions

Faults in the leg positions will only be recovered after the SLUD phase. If a wheel appears to collide with another wheel or the solar panel, the movement is stopped. Only a command that moves the wheel away from the other wheel or the solar panel is then accepted.

6. FAULT MANAGEMENT ON ROVER LEVEL

Some faults within the locomotion system cannot be recovered within the subsystem itself, but have to be reported to a higher lever in the on-board software. This applies for example when the LOCO partition has to be rebooted or the power supply for the Locomotion E-Box has to be restarted. How the handling of these fault reports work is described in the following.

Introduction to the rover software architecture

CNES has developed a TSP (Time and Space Partitioning) framework named LVCUGEN [7] as a suite of software components that can be reused for the software embedded on space vehicles, providing basic features to platforms or payloads, and allow the independent execution of embedded applications as software partitions.

Spatial and temporal isolation are key aspects in a partitioned system, and provide independent execution of system functions executed within different partitions. Among others, it allows fault containment, i.e. to prevent any partitioned function from causing a failure in another partitioned function, facilitates software re-use, validation and maintenance of partitioned applications with respect to their own criticality level and integration of new partitions.

As software and programmable logic developments were to be shared between CNES and several teams of the DLR, LVCUGEN solution was chosen as the basis for the MMX Rover software so that all parts can work as autonomously as possible, in the respect of the short schedule of the rover project.

The LVCUGEN environment offers generic partitions: MMDL (Modes Management and Data Load) for memory management, HSEM (Hardware and Software Events Manager) for event management and FDIR features, IOS (Input/Output Server) as peripheral server to manage shared access to devices, CCSW DevKit (Command and Control SoftWare Development Kit) as generic command and control partition basement, embedding CNES PUS library LibPUS for TM/TC and AUTH for authentication features.

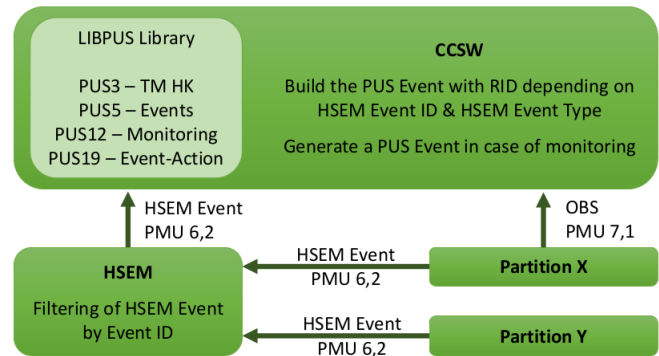


Figure 6: MMX rover FDIR flow: Communication between the relevant partitions.

Rover FDIR principles - partitions at stake

The relevant partitions for understanding FDIR at rover level are:

HSEM partition—Hardware and software event manager. Within a LVCUGEN based software, HSEM partition is in charge of software events management and filtering, FDIR management, PUS 5 information gathering, other partitions monitoring and memory scrubbing.

CCSW partition—Command and control software. The main role of the CCSW partition is the on-board control command management. As such, it includes and runs a PUS services library, that includes in particular the PUS service 12 for on-board monitoring of parameters.

Mission partition—There can be multiple mission partitions, as for instance LOCO. The mission partitions may be in charge of a particular function and, as in the case of the MMX rover, of the interface with an equipment. In the MMX rover software, mission partitions are responsible for forwarding all the data from the equipment to CCSW for generation of housekeeping telemetry and parameters monitoring, and to HSEM for partition health check and events monitoring.

There are two ways to manage FDIR: Either by providing observable values to monitor on an OBS (OBServable) sampling port and using the monitoring service of the LibPUS, or by sending events to HSEM and using the Event-Action service 19 of the LibPUS.

Eventually, the HSEM Event ID can be optionally filtered and/or forwarded to CCSW. Filtering means that an event is forwarded to CCSW after the event was received n times by HSEM. When the event is forwarded, CCSW is then able to associate a RID (Report Identifier) to the HSEM Event ID and, if required for FDIR purpose, to consider the HSEM event type which is sent with the payload of the event package.

Both of the explained mechanisms can be coupled, and sometimes have to. Indeed, particular attention must be taken with the dynamics of event generation in order not to saturate the system: messages could then be lost on the communication channel between the mission partition and HSEM. Hence, if too many events may be generated when an anomaly occurs (such as a dialog break with a device), it is preferable to implement a counter of the events occurrences and to publish

this counter on an observer port to CCSW; this counter may then be monitored by PUS service 12. This mechanism not only allows mitigating the event port saturation, but also adds the ability to program a FDIR depending on the value of this counter, thanks to PUS service 12.

7. TESTING

The implemented fault detection, isolation and recovery algorithms, from which some examples were described before, are tested either on hardware or in simulation. Tests ensure the reliability of the software and ensure, that faults are actually detected at the right moment.

On hardware, only algorithms that do not include destroying hardware can be tested. It also needs to be possible to induce a fault easily. These tests include for example disconnecting particular motor harness and sensor harness. Also, applying a certain torque carefully to the legs to see if the overtorque is detected at the correct magnitude is possible by using a spring scale as helping tool.

For simulation, the MMX Rover Simulator [8] was used. Herein, also hardware-critical faults and faults that are difficult to induce can be tested. This includes tests such as detecting and reacting to too high or low temperatures, currents or voltages. If these tests were made on hardware, there would be no guarantee that a fuse blows, for example.

Results

Having internal recovery functionalities in the software increases the continuity of operation of the MMX rover. For example, when a fault on the Hall sensor is detected, the software can switch this motor automatically into the motors feedforward control mode so that the ongoing movement can be fulfilled until the end. Without autonomous recovery, it would be not possible to perform any locomotion on the rover further until the next ground loop, which could take up to three of the 100 mission days.

8. CONCLUSION

The FDIR strategy of the MMX rover locomotion team was presented. The FMECA approach was explained in detail and showed that these efforts lead the team to more efficiency and the locomotion subsystem to more reliability. Further, it was explained why the distinction in the different mission phases is important. The fault detection in the FPGA and in the LOCO software, as well as the recovery in the LOCO software and the synergy of the individual software parts were presented.

The testing of the fault detection and isolation is ongoing. Further work will include testing of the recovery strategies, in particular the interaction between the OBC software partitions.

REFERENCES

- [1] S. Ulamec, P. Michel, M. Grott *et al.*, “A rover for the JAXA MMX Mission to Phobos,” 2019.
- [2] A. Wander and R. Förstner, “Innovative fault detection, isolation and recovery strategies on-board spacecraft: State of the art and research challenges,” 2013.

- [3] W. J. Willoughby, “Procedure for failure mode, effects, and criticality analysis (fmeca),” National Aeronautics and Space Administration, Washington, D. C, Tech. Rep., Aug. 1966.
- [4] European Cooperation for Space Standardization (ECSS), “ECSS-Q-ST-30C - Space Product Assurance - dependability,” 2017.
- [5] European Cooperation for Space Standardization (ECSS), “ECSS-Q-ST-30-02C - Space Product Assurance - failure modes, effects (and criticality) analysis (fmea/fmeca),” 2009.
- [6] H.-J. Sedlmayr, S. Barthelmes, R. Bayer *et al.*, “MMX - Development of a Rover Locomotion System for Phobos,” in *2020 IEEE Aerospace Conference*, 2020.
- [7] P. Arberet, J. J. Metge, O. Gras, and A. Crespo, “TSP-Based Generic Payload On-Board Software,” in *DASIA 2009 - DATA Systems In Aerospace*, ser. ESA Special Publication, L. Ouwehand, Ed., vol. 669, May 2009, p. 72.
- [8] F. Buse, A. Pignède, J. Bertrand *et al.*, “MMX Rover Simulation - Robotic Simulations for Phobos Operations,” in *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 1–14.

BIOGRAPHY



Juliane Skibbe studied Mathematics at the University of Wuerzburg and the Université d’Orléans. After finishing her Master’s degree in 2018, she started working as a research associate at the Institute of System Dynamics and Control of the German Aerospace Center (DLR). Today, she leads the MMX Locomotion Software team. Her main research focus is on developing locomotion control algorithms for rover for planetary exploration, in particular fault detection, isolation and recovery.



Elise Aitier Elise Aitier graduated from ISAE (space and aeronautics engineering school in Toulouse, France) in 2005. After 10 years preparing and running the satellite operations of CNES scientific missions, she joined the Flight software department as the responsible for Microcarb flight Software in 2016. She is now responsible for the Flight software and Monitoring and Control for MMX rover at CNES.



Stefan Barthelmes received his Dr.-Ing. degree in Electrical Engineering at TU Darmstadt and his B.Sc. and M.Sc. degree in Mechanical Engineering from TU München. He is currently working as a research associate at the Institute of System Dynamics and Control of the German Aerospace Center (DLR). His main research focus is model-based chassis control and simulation model development of planetary exploration rovers. Within the MMX mission, he leads the development of the rover’s locomotion subsystem.



Markus Bihler received his M.Sc. degree in computer science from the University of Applied Science Augsburg, Germany in 2015. Since then, he has been with the German Aerospace Center, Institute of Robotics and Mechatronics. His main research focus is the field of FPGAs, communication and System-on-Chip architecture inside robots. Since October 2017, he is group leader for digital electronics within the department Mechatronic Systems.



Gabriel Brusq Gabriel Brusq received its engineer degree from ESTIA, and its advanced master degree in embedded systems from ISAE-Supaero, France in 2017. Since then, he has been with the flight software department of the French Space Center (CNES). His main activity focus is the field of real-time embedded flight software, TSP and RTOS based architectures.



Franz Hacker received his Dipl.-Ing. degree in Electrical Engineering and Information Technology from TU München. Since 1994, he works as research associate at the German Aerospace Center, Institute of Robotics and Mechatronics. His main research focus is the design and optimization of analog electronics and sensors used for robotic systems in terrestrial and space applications. He contributed to the electronic system of the space qualified force feedback joystick used for the KONTUR-2 ISS-to-ground tele-manipulation experiments. Within the MMX mission, he is responsible for the central electronic box of the rover's locomotion subsystem.



Hans-Juergen Sedlmayr received his Dipl.-Ing degree in Electrical Engineering from the University of Applied Science Munich in 1992. Since 2001, he has been with the German Aerospace Center, Institute of Robotics and Mechatronics. His main research focus is in the field of radiation testing of electric and electronics parts and embedded software development inside robots for terrestrial and space applications.