FDIR Handling in Eu:CROPIS

Olaf Maibaum (1), Ansgar Heidecker (2), Fabian Greif (2), Markus Schlotterer (2), Andreas Gerndt (1)

(1) German Aerospace Center (DLR), Simulation and Software Technology
Lilienthalplatz 7, 38108 Braunschweig, Germany
Phone: +49 531 295 2974, Mail: Olaf.Maibaum@dlr.de

(2) German Aerospace Center (DLR), Institute of Space Systems
Robert Hooke Str. 7, 28359 Bremen, Germany
Phone: +49 421 24420 1166, Mail: Ansgar.Heidecker@dlr.de

Abstract: Fault detection, isolation, and recovery (FDIR) mechanisms in on-board software are essential to guarantee the survival of the satellite in case of a hardware malfunction. E.g., outage of essential attitude control system (ACS) actuators or sensors can lead to mission loss. The on-board software has to handle such situation autonomously by switching to cold redundant devices or by isolation of information from hot redundant devices. The FDIR implementation for the ACS of the spin stabilized small satellite Eu:CROPIS (Euglena Combined Regenerative Organic food Production In Space) is shown in this paper.

1. INTRODUCTION

This paper shows the implemented software architecture and methods for FDIR handling in the Eu:CROPIS (Euglena Combined Regenerative Organic food Production In Space) ACS. It starts with a short introduction of the Eu:CROPIS mission and of the implemented ACS. Sections 3 and 4 show the underlying execution platform and the used services from the Package Utilization Standard (PUS) [5]. The software architecture and FDIR handling is presented in section 5. The paper closes with the conclusion and outlook on future work.

2. EU:CROPIS

The mission Eu:CROPIS is the demonstration of the feasibility of restartable and sustainable life support systems on a pure biological basis. Such systems enable the production of food and atmosphere from waste like urine and phosphate [1]. Furthermore, the system should be reliable enough for long duration missions. The biological experiments require different levels of gravity. The target gravities of Eu:CROPIS are 0.16 g (Moon) and 0.38 g (Mars) respectively in the biological experiment compartments. This is achieved by a spin stabilized satellite which can change its rotation speed and thereby the centrifugal force (gravity) in the biological experiment compartments during mission time. Experiment compartments are placed at a reference radius of 0.35 m measured from the designed spin axis. The launch of the mission was at 03.12.2018 as part of Spaceflight's SSO-A rideshare mission launched from Vandenberg Air Force Base with a Falcon 9 launch vehicle.

The used satellite bus is based on the DLR compact satellite bus program, which is a research and development platform using a component-oriented design. The bus for the Eu:CROPIS mission is a spin stabilized platform with a cylindrical body. Diameter of the body is 1000 mm and height is 1100 mm [1]. The bus has two sections: one with the separated payload compartments at the upper deck, and one with the bus section. Figure

1 shows both sections. The mass of the satellite is around 230 kg. The orbit is sun synchronous at 580 km altitude.

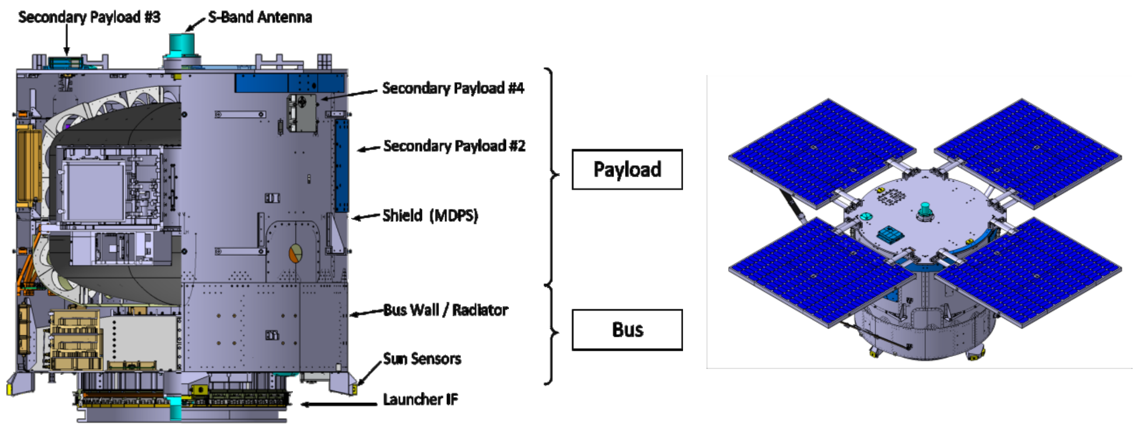

Figure 1 Eu:CROPIS Satellite Bus

2.1 ACS

The main requirement to be fulfilled by the ACS is to generate gravity in the biological compartments and orient the z-axis into sun direction to ensure power generation by the solar panels. The satellite bus is spinning around the z-axis between 5 to 31 rpm. To keep sun orientation of the solar panels, the spin axis has to be reoriented by ~1 deg/day.

As actuators, the ACS uses three magnetic torquers to control the rotation and spin axis. As sensors it uses two magnetometers, ten sun sensors and four angular rate gyroscopes. Sensor data is filtered by an Unscented Kalman Filter (UKF) which is designed for spin stabilized satellites. Table 1 shows the key figures of the ACS hardware components. For a detailed description of the used UKF and the ACS, see [3] and [4].

Amount	Hardware	Key figure
3	Magnetic Torquer	30 Am^2
2	Magnetic Field Sensor (MFS)	noise < 1000 nT (3σ)
10	Sun Sensor	noise < 0.3 deg (3σ)
4	Angular Rate Gyroscope	RW 0.08 deg/$\sqrt{h}$, Bias 9 deg/h (3σ)
2	GPS Receiver	DLR-Phoenix

Table 1 Attitude control system hardware devices

3. TASKING FRAMEWORK

The communication and processing is realized by the Tasking Framework inside the Eu:CROPIS ACS. The Tasking Framework is a reactive asynchronous execution platform with support for multicore processors and extensions for distributed on-board systems. For the Eu:CROPIS ACS, a non-distributed single core configuration is used. The mean for communication in the Tasking Framework are channels, which will specialized to different types of data buffers by overwriting the basic channel class. E.g., data buffers are single buffers, double buffers, or FIFO queues. For application code, a channel can be specialized with own implementations, like interfaces to devices. Beside different types of data buffers as channel, the Tasking Framework provides events as specialization of a channel. An event is connected to the system clock and provides means to start task executions in a periodic or relative manner.

For computations in a system, tasks have to be implemented by overriding the execute method of the basic task class. An instance of a task implementation is connected to at least one input. Inputs control the execution of a task implementation and interface to incoming channels of a task. The number of expected data items, which have to be pushed on the associated channel to start the execution of a task, have to be configured for the input. For example, figure 1 shows a task connected to two incoming channels, one event and one outgoing channel with computation results of the task. The input of channel 1 is configured to start the task execution with one data item, whereas the input of channel 2 expects two data items on the channel. The input of the event is configured as optional input with the final attribute. In this example, the task execution is initiated by one data item pushed to channel 1 and two data items pushed to channel 2, or if one event is fired independently of the status of different inputs. Thus, the event in this example acts as time out mechanism for the task execution, e.g. when a device is faulty and did not respond with a data message. Further computation steps can be connected as tasks to the outgoing channel 3, which are triggered for execution when the task pushes data on channel 3.

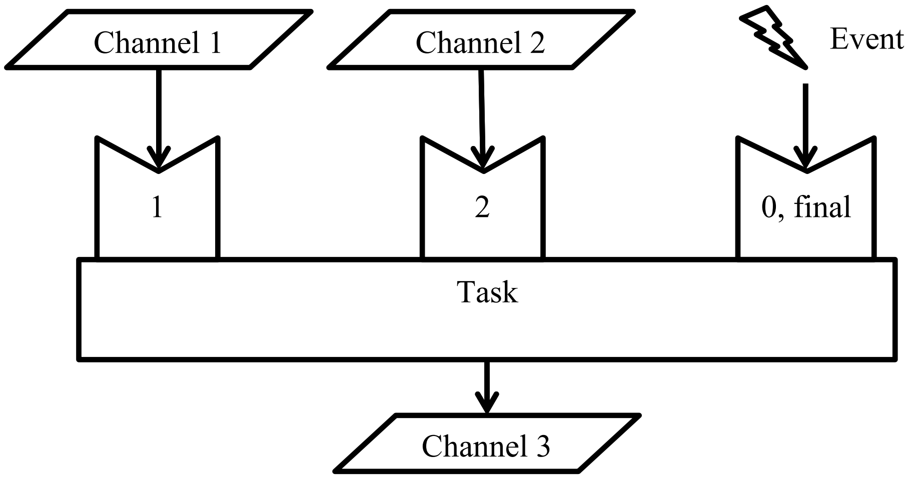

Figure 2 Task Configuration Example

4. PUS SERVICES

The implemented services in the Eu:CROPIS command and data handling system (CDH) following the Package Utilization Standard (PUS) [5]. Each application of a subsystem has to register service handler at the CDH system to provide the service implementation of the subsystem. The CDH system can call the subsystem specific implementation of a service request by the application and service identification of a PUS data package.

4.1 Device Command Distribution Service

To access parameters of the ACS, the “Distribute Register Load” Command from the “Device Command Distribution Service” (PUS service 2, subservice 2) is used in lack of the Parameter Management Service. By default, this service was foreseen to configure hardware systems but can also be used to configure software systems. The service provides means to load a parameter with specific data from the load command and, as extensions of the PUS service, a parameter dump request, and the possibility to safe parameter settings as boot configuration to a context memory.

The updated PUS [6] published in 2016 introduces the parameter management service as implemented in Eu:CROPIS. Following the updated PUS, the functionality is available by the PUS service 20. In case of software reuse of the Eu:CROPIS implementation, the service will be assigned to service 20 in following missions.

4.2 Diagnostic Reporting Service

Beside regular “Housekeeping Parameter Reports” (PUS service 3, subservice 25), the PUS Diagnostic Reporting Service provides also “Diagnostic Parameter Reports” (PUS service 3, subservice 26). For Eu:CROPIS, all ACS channels are addressable by the service “Define New Diagnostic Parameter Report” (PUS service 3, subservice 2). This allows a deep inspection of the current software state during the operational phase, which can be adapted by ground control commanding.

The generation of diagnostic parameter reports is controlled by different generation modes. The Eu:CROPIS ACS implementation provides two modes: “Periodic Diagnostic Parameter Report Generation Mode” (PUS service 3, subservice 18), and “Filtered Diagnostic Parameter Report Generation Mode” (PUS service 3, subservice 20). Instead of using only percentage parameter values and absolute delta values threshold types, the Eu:CROPIS ACS extends the standard definition of the filtered diagnostic parameter report generation by threshold types equal, bigger than, and lesser than.

The updated PUS removes filter functionalities for diagnostic reports. In case of software reuse in upcoming missions, the filter functionality will not be removed because the functionality is needed by the FDIR reporting functionality implemented in the Eu:CROPIS ACS.

4.3 Event Reporting Service

To announce outstanding events in a system, PUS defines the Event Reporting Service with different severities (PUS service 5, subservices 1, 2, 3, 4). For this purpose, the Eu:CROPIS ACS uses the implementation of the CDH subsystem. The satellite bus internal interface is a topic provided by the outpost-core library. [7]

4.4 Function Management Service

The PUS Function Management Service (PUS service 8, subservice 1) provides the mean to initiate special functionalities in the software, e.g. reset error counter values or change control modes. Each special functionality is addressed by an enumeration followed by the functionality specific parameters.

5. FAILURE HANDLING

5.1 Software Architecture

The whole software architecture of the Eu:CROPIS ACS is based on the Tasking Framework. The software architecture concepts for the implementation of hardware interfaces and the UKF are described in [4].

The FDIR handling in the Eu:CROPIS ACS has several logical layers. In the first two logical layers, which are software interfaces to actuator and sensor hardware as well as to the UKF, failures are detected and isolated. The mechanism is described in the next section. All detected failures are reported on the error channel for a further error handling and the collection of diagnostic data to analyze error cases by the satellite support team. Both tasks are activated by pushing error identifications to the error channel.

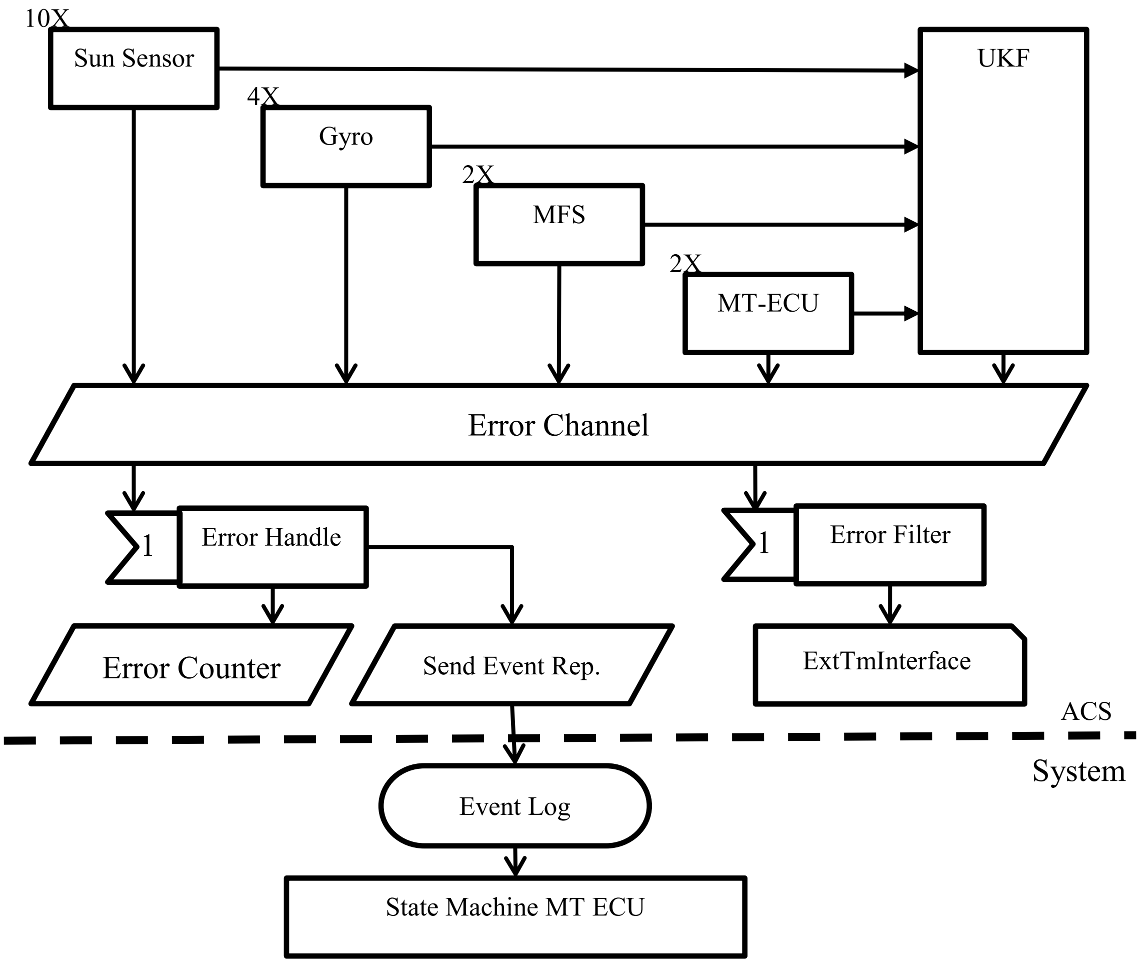

Figure 3 FDIR Software Architecture in the Eu:CROPIS ACS

The error filter sends diagnostic data by serializing data from a set of channels related to the error identification as diagnostic parameter reports. E.g., in case of a failure of the magnetic torquer, all channels related to the sender and receiver tasks of the magnetic torquer software interface are packed to a diagnostic report. The mean of error filtering is realized by the filter functionality of the "Filtered Diagnostic Parameter Report Generation Mode”. The used diagnostic reports are predefined with its filter in the ACS and cannot be altered during operation, except by a software update.

The extended telemetry interface can block the sending of diagnostic reports to limit the number of diagnostic reports in case of a permanent failure. The amount of diagnostic reports is controlled by a parameter, modifiable by the Distribute Register Load Command, and the counter of sent reports is reset as standard operation during contact by one function call of the function management service.

Error handling by the error handle task has two steps. The first step is increasing the error counter of the error counter channel. The second step is to evaluate the criticality of failures. For this step, different critical failure cases are identified for each error counter, e.g. a device reports an electrical error or the communication with the device is permanently disturbed. In case of a critical failure case, an event report is sent to the event channel, which also pushes a message to the event log topic to initiate a recovery operation by the FDIR state machine on system level.

Figure 3 shows all related software components, tasks, and channels responsible for FDIR handling in the Eu:CROPIS ACS.

5.2 Detection and Isolation

Failure detection is executed by the software interfaces to the sensor and actuator hardware and, in a second step, in the UKF. In all failure cases, the fault detection is reported by an error identification on the error channel.

At first step, the communication with the sensor and actuator hardware is checked in the corresponding software interfaces. These checks are: check on successful sending of messages to the hardware; successful receiving of messages from the hardware; and format of received messages is valid. The validity check depends on the message format and encloses checksum checks and consistency of message type and message length. Further checks are performed on the message data by checking the status information reported by the device and the validity information and consistency of data. E.g., validity information are albedo state information from sun sensors or range exceed information from gyroscopes.

Depending on the failure and validity state, detected by all checks on interface level, data is not sent to the UKF or it is marked as inaccurate data to isolate them from further processing. If possible for a faulty hardware device, a failure handling is initiated by the software interface to bring the faulty hardware device back into an operational state, e.g. by sending a command sequence for the failure case advised by the manufacturer. Single failures are not handled. Consistent failures over a specified number of control cycles are reported in the error channel to initiate recovery actions, depending on the redundancy level of the faulty hardware device.

The second step in failure detection is performed by the UKF. First, it checks sensor data on validity, e.g. measured sun angle data during eclipse is invalid. For all valid sensor data, the Mahalanobis distance is computed to detect inconsistent state information from a sensor device. Implausible data will be isolated from the computation of the high accurate attitude state information which is sent to the controller task.

5.3 Recovery

Error identifications reported on the error channel are read by the error counter task. For each error identification exists an error counter in the error counter channel and an entry in the critical failure report list, which is filled in case of critical failures. By default, the error counter task reads all error identifications from the error channel and increases the corresponding error counter. When an error counter reaches a limit, the error counter task checks the list of critical failure reports and if one is defined for the error identification, the critical failure report is published on the event log to inform the system level and to report the event by the PUS event reporting service. The error counter limit can be changed by the "Distribute Register Load Command" service.

Reported critical failure events are managed by the system state machine and can initiate a power reset of hardware devices or a switch to cold redundant hardware devices. Figure 4 shows the state machine for recovery actions of the MT-ECU (magnetic torquer electronic control unit).

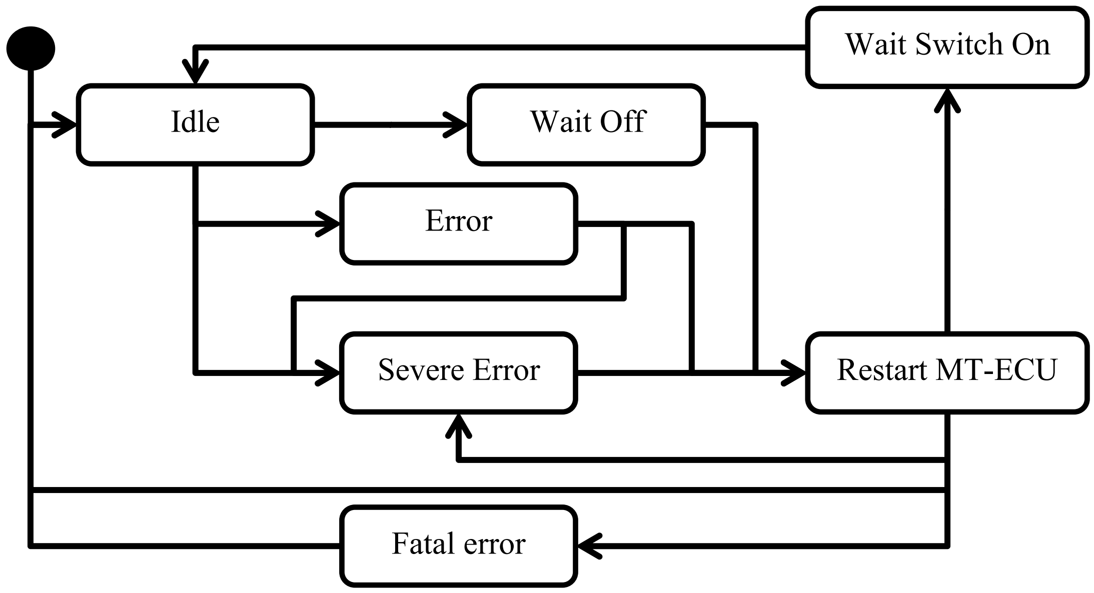

Figure 4 State Machine for Recovery Actions of Magnetic Torquer

The state machine starts in state *idle* and waits on three conditions: the cold redundant device is switched on by ground control command or an electrical or software error is received as critical error report on the event log. In the first case, the state machine goes into state *wait off* to wait until the active MT-ECU is powered off. For an electrical error, the state changes to the *severe error state*. For the software error, reported by the MT-ECU, the state changes to *error*. The state *error* switches off the active MT-ECU and commands a restart of the inactive MT-ECU, until the maximum number of allowed restarts without ground interaction on an MT-ECU is not reached. If the maximum number of restarts is reached, the state changes to *severe error*. In state *severe error,* the active MT-ECU is marked as broken and a switch over to the cold redundant MT-ECU is initiated.

The state *restart MT-ECU* switches on the MT-ECU, which is marked as active. If a failure is detected, go to state *severe error* so long the cold redundant MT-ECU is not marked as broken. Otherwise go to the state *fatal error* to wait on ground interaction. The remaining state *wait* changes to the state *idle* when the selected active module is powered on.

6. CONCLUSIONS AND OUTLOOK

The paper showed the architecture and means for the FDIR handling of the Eu:CROPIS ACS. First checks for failure detection on communication status and data from the actuators are applied at interface level. At the UKF, the Mahalanobi distance is used to isolate implausible data. Each detected fault is reported to two tasks in the ACS, one to send diagnostic data, and the other to count the detected faults and to initiate the recovery actions. On system level, a state machine is used to manage the power states of devices. First experiences in the LEOP and commissioning of the Eu:CROPIS mission show that the implemented FDIR mechanism works as expected. The analyses for the reasoning of increasing FDIR counters on the flight model in space is supported by the diagnostic reporting service itself and the reporting actions triggered by FDIR events, because we get an insight of all internal ACS data with up to an 10 Hz resolution by the diagnostic reporting service. All applied services for FDIR handling can be reused as configured in future missions.

7. REFERENCES

[1] S. Kottmeyer, C.F. Hobbie, F. Orlowski-Feldhusen, F. Nohka, T. Delovski, G. Morfill. The Eu:CROPIS Assembly, Integration and Verification Campaigns: Building the first DLR Compact Satellite. IAC 2018, Bremen (2018).

[2] F. Dannemann, F. Greif, Software Platform of the DLR Compact Satellite Series. In: Proceedings of the 4S Symposium, Mallorca, Spain (2014).

[3] A. Heidecker, T. Kato, O. Maibaum, M. Hölzel, Attitude Control System of the Eu:CROPIS Mission. IAC 2014, Toronto, Canada (2014).

[4] O. Maibaum, A. Heidecker. Software Evolution from TET-1 to Eu:CROPIS. In: Digest of the 10th International Symposium of the International Academy of Astronautics. pp. 195-198. Wissenschaft und Technik Verlag, Berlin (2015)

[5] ECSS-E-70-41A: Ground Systems and Operations – Telemetry and Telecommand Packet Utilization, 30 January 2003.

[6] ECSS-E-ST-70-41C: Telemetry and Telecommand Packet Utilization, 15. April 2016.

[7] https://github.com/DLR-RY/outpost-core: Outpost Core. Link checked at 13. December 2018.