19th IFAC Symposium on Automatic Control in Aerospace
September 2-6, 2013. Würzburg, Germany

Robust and Efficient OBDH Core Module for the Flexible Picosatellite Bus UWE-3

Stephan Busch *, **Klaus Schilling**

* *Julius Maximilians Universität, Würzburg, Germany*
(e-mail: {busch, schi}@informatik.uni-wuerzburg.de)

Abstract: Pursuing the ambitious goal to establish formations of cooperating small satellites, the University of Wuerzburg established a road map for technology implementations leading towards the realization of coordinated formations of picosatellites. Within the recent developments of the third generation of UWE satellites, special focus was put on the design of a robust and efficient Onboard Data Handling core module. The core module plays a central role in the design philosophy of the UWE-3 bus as it can be seen as the fundamental core which is the only subsystem continuously operating. It is built around two redundant ultra low power microcontrollers in dynamically decided master-slave configuration. One of the main features of the core module software is its flash protection and recovery, where both microcontrollers mutually protect each other by periodically comparing the content of the flash memories with each other and perform corrections if required. This contribution gives a short overview of the UWE-3 picosatellite bus and elaborates in detail on design and test of the OBDH core module.

Keywords: picosatellite; onboard data handling; protection; redundancy.

1. INTRODUCTION

Pursuing the ambitious goal to establish formations of cooperating small satellites, the University of Wuerzburg established a road map for technology implementations leading towards the realization of coordinated formations of picosatellites (Schilling et al., 2011). The UWE (University of Wuerzburg Experimental Satellite) program was initiated in 2005 with the successful launch of the first German picosatellite UWE-1, which had the main objective to establish a robust communication link to the ground segment. The successor UWE-2 was launched in 2009, aiming to demonstrate the capabilities of picosatellites in the field of attitude determination (Schmidt et al., 2008). Taking forward the research activities centered on the UWE program, the next satellite UWE-3 is scheduled to be launched in late 2013. With UWE-3 for the first time a modular and flexible design of the picosatellite bus was introduced. The design complies with the CubeSat specifications and is optimized to support robust and rapid development, integration and testing of the satellite as well as easy maintenance, extension and replacement of subsystems even after integration. Thus, the system constitutes a robust, extendable, and flexible basis for the future UWE satellites (Busch and Schilling, 2012).

The modular UWE picosatellite bus (see Fig. 1) is based on a stack of several subsystem printed circuit boards (PCB) designed according to an internally defined standard in order to support compatibility and extensibility for future developments. The set of subsystems include all typical satellite subsystems such as power, onboard data handling, communication and attitude determination and control, but are optimized with respect to size and mass. Each subsystem is implemented on an individual PCB, interconnected with the other subsystems by a backplane with

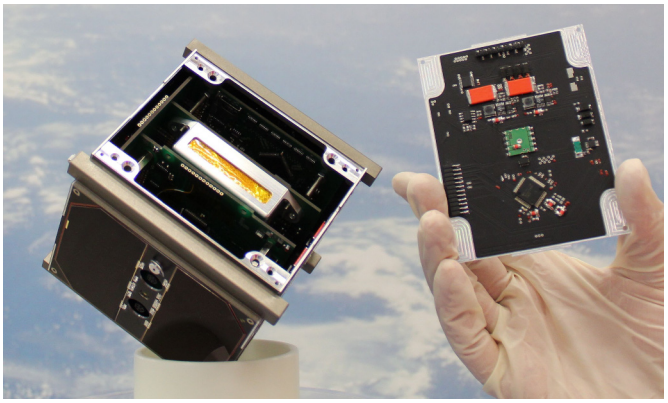

Fig. 1. Modular UWE-3 Satellite Bus with removed multifunctional plug-and-play side panel (Prototype).

standardized connectors. Thus, the subsystems can be stacked densely to achieve a compact design without any need for further elaborate wiring. Moreover, the satellite design consists of six side panels. Featuring an aluminum core, the panels function as structure, radiation shields and thermal regulators. Electrical wiring is carried out by two thin PCB layers located on both surfaces of the aluminum core and interconnected by PCB vias. These PCB layers carry solar cells and additional electronics (e.g. sensors which need to be exposed to the outer environment) and are automatically connected to the satellite's backplane and identified by the central data handling module when the panels are mounted on the main structure.

Within the recent developments special focus was put on the design of a robust and efficient Onboard Data Handling (OBDH) core module. The core module plays a central role in the design philosophy of the UWE-3 bus. It can be seen

as the fundamental core dedicated to satellite housekeeping operation which is the only subsystem continuously operating from the very first instant of operation until the satellite ceases operation. All other subsystems, which usually carry their own processing units, might be powered down, power cycled or even reprogrammed under the control of the core module, in order to save power (partial power down) or to ensure proper operation considering the extremely limited power budget on picosatellites and enhanced threats due to radiation induced single event effects in COTS components. The design of the core module focuses on energy efficiency and robustness.

Especially ionizing radiation can affect the operation of spacecraft systems significantly, ranging from temporary disturbances like data corruption to fatal failures resulting in total mission loss. Modern device technologies can be quite sensitive to radiation effects due to feature miniaturization and power optimized semiconductor designs. The utilization of radiation-hardened devices can increase robustness but is often no option due to cost, availability, efficiency or capability. For this reason, special care has to be taken in both, hardware and software design, in order to mitigate severe effects despite the sensitivity of some components.

It is nearly impossible for a typical University Cubesat Project to rely on hardened components only or to analyze each single COTS component for its radiation tolerance. Instead, the design strategy of the UWE-3 project for achieving robustness is mainly failure tolerant design using mainly COTS components. Thus, short failure periods are accepted as long as the system protects itself from permanent damages, detects the fault state and recovers in a timely manner.

Fig. 2. OBDH Core Module with two redundant microcontroller units measuring only $9x9\,\mathrm{cm}^2$ with an average power consumption of less than 10 mW.

2. HARDWARE DESIGN

The OBDH core hardware is build around two redundant ultra low power microcontroller units (MCU) in a master-slave configuration which is dynamically decided by a separate watchdog controlled arbitration unit (see Fig. 3) Beside several cross connections, like DMA supported high speed communication, the active controller (master) can access the other (slave) via its embedded emulation module (EEM) using a JTAG interface implementation. This gives full control of the remote hardware, independent of the current state of its programmed software. Thus, the slave can be accessed and controlled or even completely reprogrammed for memory recovery or secure software updates after launch by the current master MCU. The system further provides a temperature compensated high precision real-time clock for clock synchronization between the MCUs and two redundant flash memory based mass storage devices as well as several housekeeping sensors for monitoring currents, voltages and temperatures. Two Spy-By-Wire interfaces and a serial debug communication link are routed to the backplane to allow comfortable access to the system for software flashing and in- system debugging via the umbilical line, even when the satellite is completely integrated.

Besides robustness, the OBDH core module is designed for very low power requirements. In nominal operation, the complete subsystem consumes less than 10 mW (average). This ensures the operability of the housekeeping system at all times, even in severe fault situations, e.g. when the satellite runs into low energy. A redundant power converter directly supplied from the unregulated power bus circumvents the electrical power subsystem in case of a malfunction. This ensures the continuity of the proper control of the satellite to deal adequately with the situation and recover the system for example by a controlled power cycle of specific subsystems or even of the whole satellite bus (except the core module). The very low power requirement further allows to bridge temporary supply voltage drops, maybe caused by high current peaks of a subsystem, for several seconds using some high capacity backup capacitors.

2.1 Toggle Watchdog Unit

In normal operation, the current master periodically triggers the so-called toggle watchdog unit (TWU). In case the master fails to serve the watchdog within a specified timeout, i.e. due to MCU hangs, the TWU would perform a reset on both MCUs and subsequently swap the master-slave configuration by asserting the $\overline{RESET}$ signal of the former slave MCU, leaving the former master in reset state. The same logic would control isolation switches in order to connect various signal lines of the new master like general purpose IO lines, analogue to digital converters or serial data interfaces like I^2C, SPI or UART, to the satellite bus. However, the slave controller might be enabled afterwards by the master if necessary, e.g. for processing assistance, software checks or reprogramming.

The TWU itself is implemented using an external low power watchdog IC, which is connected to the CLK input of a D-Type flip flop (see Fig. 4). Feeding back the flip flop output to its data input, the generated reset impulse is translated to a state toggle of its opposing output signals. A subsequent combination of OR gates include the slave enable inputs which allow the master to additionally wakeup the slave unit. The final AND gates reflect the reset impulse on both outputs to ensure that, besides the toggling, both MCUs are reset before the master is activated. This is especially important if the former slave unit was activated by assertion of the slave enable signal.

This configuration enables the OBDH core module to monitor itself, recover from fault conditions like system hangs, identify faults sources in the hardware like SEU induced

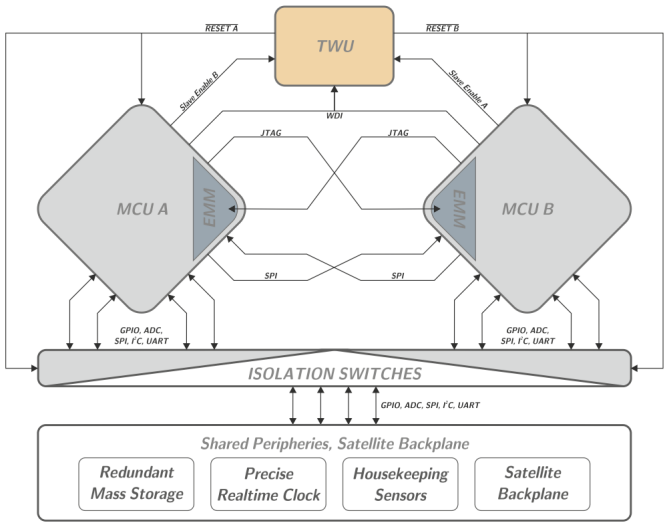

Fig. 3. Block diagram of the core modules TWU architecture for dynamic master slave configuration arbitration.

memory corruptions, and perform recovery mechanisms on software level.

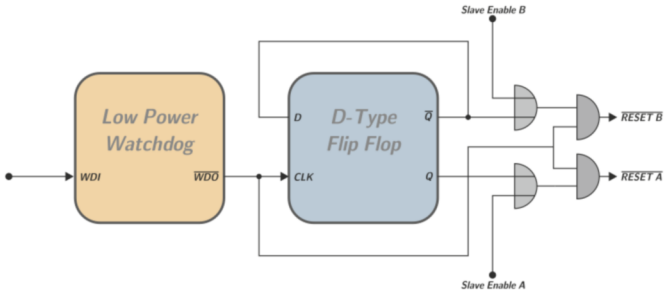

Fig. 4. Block diagram of the toggle watchdog unit implementation.

2.2 Quad-Redundant Power Cycling Unit

However, not all malfunctions can be resolved by simple resets and/or flash memory maintenance. Especially latch-up conditions or so-called functional interrupts might end up in fault states on semiconductor level which usually requires a power cycle to recover in case the damage is reversible at all.

For this reason the core module provides an independent time-limited power supply. This enables the control of complete power cycles of the entire satellite or allows the detection and recovery from sudden voltage drops of the power bus i.e. caused by latchups, and thus prevents from uncontrolled resets of the core module.

In order to autonomously recover from latchup states inside the core module itself, which could not be recovered by a controlled TWU reset, the system provides an independent quad-redundant power cycling unit (PCU) with implicit majority voting arbitration. The watchdog controlled circuit autonomously performs a power cycle of the core module by disconnecting it completely from the backplane using the aforementioned bus isolators. This way complete voltage isolation of the low power circuits is ensured to recover from non-destructive fault states in the semiconductor circuits.

The PCU can be seen as a watchdog controlled switch, which would perform a controlled power cycle with about one minute duration in case its WDI signal is not triggered by the master MCU for a specified timeout. As the PCU itself is a critical system, not being monitored by any further instance, its design comprises a redundancy without a need for a central arbitration unit to avoid single point of failures. This is achieved by the utilization of four individual watchdog controlled high-side switches in serial-parallel configuration (see Fig. 5).

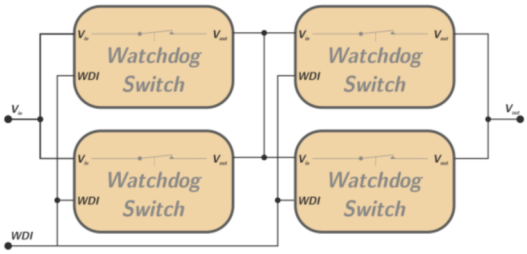

Fig. 5. Block diagram of the quad-redundant power cycling unit with implicite majority voting.

This way, the output signal reflects an implicit majority voted decision, conducting power when most of the individual switches are on, and isolating when most of the switches are off. Hence, the circuit allows at minimum one switch to fail without restricting the overall operation.

The individual watchdog switches are implemented in a similar way as done for the TWU unit (see Fig. 6).

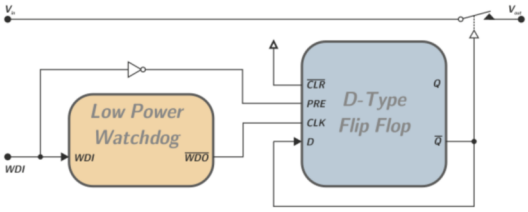

Fig. 6. Block diagram of a single PCU watchdog switch unit.

A watchdog switch unit is realized using a D-Type flip flop in a toggle configuration. A low power watchdog would swap the state of the flip flop in case its WDI signal is not triggered for a timeout in the order of a minute. The following watchdog timeout would activate the on-state of the connected high side switch again. A signal change on the WDI input not only resets the watchdog but also synchronizes the flip flop to a defined default on-state and ensures proper bootstrapping.

The high side switches itself further provide an over current protection which is tuned to the expected maximum current consumption of the OBDH core module. In case of a severe malfunction due to a latch up in the core module the switches would limit the current by dropping the voltage at their output and hence protect the semiconductors from permanent damages due to burnouts. After a while, the PCU watchdogs would shut down one after another until the core module is completely potential-free. After

another minute the switches continue the operation of the core module again.

For proper recovery from non-destructive latchups it is essential that the corresponding circuit is completely potential free. In the case of the ultra low power microcontrollers in the present design, this is not a trivial task as the controllers can operate with extremely low current consumption, which implies that the off-resistance in the power path is high enough to drop the supply voltage completely.

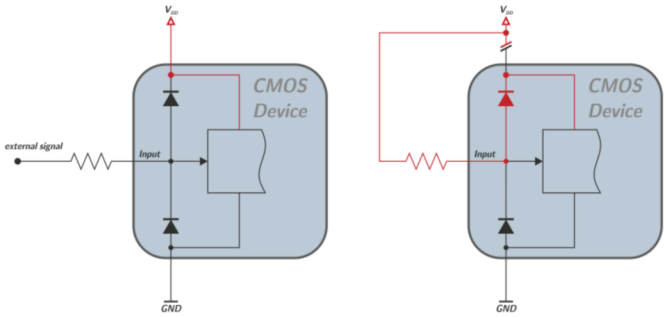

Fig. 7. CMOS device with clamping diodes connecting to the supply rails can provide undesired power supply paths in partial power down operation (right).

Standard CMOS devices typically provide electrostatic discharge protection in terms of clamping diodes connecting to the supply rails on each input. Powering off the device by disconnecting VDD without isolating its signal inputs from parts of the circuit which are not powered down might result in undesired power supply paths to the device as can be seen in Fig. 7. Due to the minimal current drain of the utilized ultra low power microcontrollers, this path would still be sufficient to prevent the supply voltage via the signal input resistance and the protection diode from dropping significantly. In the design of the OBDH core module the isolator switches shown in Fig. 3 isolate the MCUs from the rest of the satellite in case the PCU performs an internal power cycle. This way it is assured that the semiconductors are made potential free in order to recover from latch-up states.

3. SOFTWARE DESIGN

One of the main features of the core module software is its mutual flash protection and recovery in the master-slave configuration. To protect the flash memory's floating gate cells of the COTS microcontrollers from corruption due to single event effects (SEE) or total ionizing dose (TID) induced prompt or anneal effects (Micheloni et al., 2010), several check and recovery algorithms are executed periodically. Many error detection and recovery methods for memory protection can be found in literature, most of them focus on minimizing the necessary memory overhead. In this project the limiting factor is the execution time on the low power microcontrollers for mutual protection and recovery operation via the JTAG cross connection.

For this purpose, the entire software image is maintained in redundant copies on both MCUs. As can be seen in Fig. 8, the flash memories of both MCUs are partitioned into three regions. The first region of the master LP (local program) stores the executable program memory, which is actively used by the microcontroller for operation. A second region LR (local replication) stores a clone of the program memory. A third smaller region LC (local checksums) stores two bytes pseudo signature analysis [1] (PSA) checksums of each segment of the program memory. This configuration allows a single MCU to verify its own flash memory locally and recover corrupted segments if necessary. In normal operation, when both MCUs are physically operational, also the redundant information on the slave (remote) device is used. Using the remote microcontroller's EEM via the JTAG cross-connection a complete pseudo signature analysis checksum of the entire flash memory of the remote device (slave) can be calculated within less than 3 seconds to be compared with the local checksum on the current master device. The advantage of the PSA is that it is part of the EEM hardware implementation and can therefore be calculated remotely without the need for transmitting all the memory content.

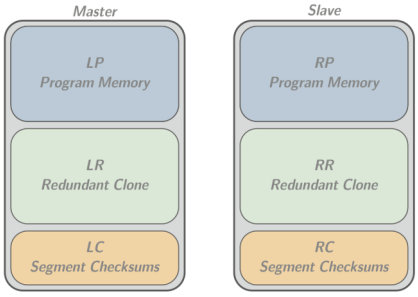

Fig. 8. Flash memory organization storing redundant copies and checksum tables on each microcontroller unit.

If the quick checksum comparison of the entire remote flash fails, corrupted segments are quickly identified by fast segment wise checksum comparisons. Erroneous segments are recovered by using quick bitwise majority voting incorporating local and remote redundancies.

Let LP be a bit in LP, LR be a bit in LR and so on. Then, Table. 1 represents the quadrivalent majority voting (QMV) incorporating all four redundant copies of the program memory image.

Table 1. Bitwise majority voting incorporating four redundancies.

QMV		$\overline{RP}$		RP	
		$\overline{RR}$	RR	$\overline{RR}$	RR
$\overline{LP}$	$\overline{LR}$	0	0	0	x
	LR	0	x	x	1
LP	$\overline{LR}$	0	x	x	1
	LR	x	1	1	1

The red marked x-values denote the combinations of LP, LR, RP, and RR, which are not unambiguously decidable as no clear majority emerges.

However, the quadrivalent majority voting function can be quickly calculated by evaluating the following two expressions:

$$C1 = (LP \wedge LR) \vee (RP \wedge RR) \quad (1)$$

[1] see Texas Instruments - MSP-GANG430 User's Guide

$$C2 = (LP \vee LR) \wedge (RP \vee RR) \quad (2)$$

Regarding the Karnaugh diagrams for $C1$ and $C2$ (see Table. 2 and 3) it can bee seen, that the quadrivalent majority voting is decidable if and only if $C1 = C2$ whereas its result is then simply defined by $QMV := C1\ (= C2)$.

Table 2. Karnaugh diagram for $C1$.

$C1$		$\overline{RP}$		RP	
		$\overline{RR}$	RR	$\overline{RR}$	RR
$\overline{LP}$	$\overline{LR}$	0	0	0	1
	LR	0	0	0	1
LP	$\overline{LR}$	0	0	0	1
	LR	1	1	1	1

Table 3. Karnaugh diagram for $C2$.

$C1$		$\overline{RP}$		RP	
		$\overline{RR}$	RR	$\overline{RR}$	RR
$\overline{LP}$	$\overline{LR}$	0	0	0	0
	LR	0	1	1	1
LP	$\overline{LR}$	0	1	1	1
	LR	0	1	1	1

The advantage of this approach is that it can be calculated rapidly on large data arrays as the expressions can be evaluated just using AND and OR operations on whole data words which would calculate the bitwise result with only 6 instructions simultaneously for 16 bits on the used architecture.

The utilized microcontroller architecture further allows reading memory content in so-called marginal read mode (Quiring, 2008). This refers to a technique were the floating gate cells of the flash memory can be read with adjustable charge thresholds. This technique allows to reveal memory cell bit-corruptions in terms of insufficiently charged or insufficiently erased memory cells even when the corruption does not yet take effect in normal read mode.

4. PROTOTYPE TESTS

Testing the described hardware and software recovery mechanisms by exposing the system to sources of ionizing radiation i.e. exposition to partical accelerators, is more than unrealistic in the context of this project. Fortunately sufficient testing can in most cases be done using Software Implemented Fault Injection (SWIFI) techniques as a simple and low-cost alternative (Arlat et al., 2003). Various approaches for different platforms are available (Ziade et al., 2004) and some have been especially developed and used for Space applications like JIFI from JPL [2] (Some et al., 2001).

In order to enable SWIFI analysis of the embedded core module a lightweight fault injector service was implemented in software. This service basically runs in the background and receives commands via the core modules debug interface. Upon request, the service modifies either the remote or the local memory content at any specified address location in flash memory or RAM. For this purpose the current program execution is paused for a moment while the modification is performed. In case the fault injection addresses code segments of currently executed instructions, the code which injects the modifications is copied to and executed from RAM in order to enable erasure and reprogramming of the corresponding segment.

[2] Jet Propulsion Laboratory

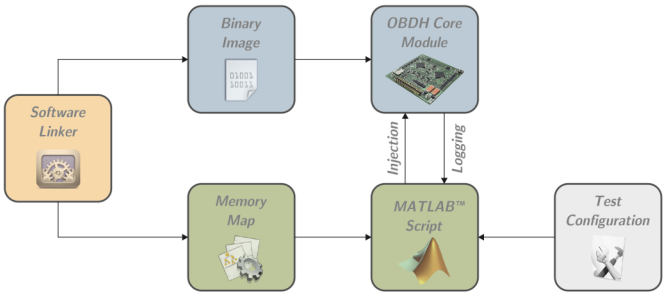

Fig. 9. Software implemented fault injection and fault logging test setup.

The embedded SWIFI service is controlled via a PC which coordinates the fault injections and logs the systems response. The complete test setup is depicted in Fig. 9. After compilation of the OBDH core module software, the linker generates the corresponding binary image and a memory map file with the memory linking information. Once the binary image is flashed to the OBDH core module it is automatically replicated locally according to the mentioned redundancy configuration and further cloned to the remote device via the JTAG interface crosslink. A MATLAB^{TM}script automatically parses the memory map including the symbol addresses of all linked variables and functions. A configuration file stores the parameters of a test campaign.

According to the configuration, the script requests bit error injections at individual rates for different regions in the entire address range of both microcontrollers via the debug communication link. With the help of the memory map information, specific regions like RAM, TEXT, or even individual modules, functions or variables can be exposed to systematic fault injections whereas other memory regions like the link location of the fault injection service can be excluded. The script further monitors the core module, logging events like reset, system hangs, TWU toggle resets, PCU power cycles, etc. during the test execution.

In several extensive long term test campaigns more than 1 million bit error injections have been performed in the entire memory address space of the running core module during normal operation. The tests have been conducted separately for the individual memory regions like periphery registers (PER), statical allocated RAM (BSS), stack, and local program area (LP) in the microcontrollers flash memory. Thus, deviations in failure sensitivity to bit errors can be revealed. Table 4 gives an overview of the recently performed test campaign. Figure 10 exemplarily shows a more detailed view on the reset statistics during BSS injections.

As the most important result the system could demonstrate to recover from a huge number of injected memory failures. Only in four cases a sequence of injected bit errors in the local program memory resulted in a self-locking fault state which could not be recovered neither by the redundant microcontroller nor by the vendors program-

Table 4. SWIFI test campaign overview.

Target:	PER	BSS	STACK	LP
Size (bytes)	4096	4332	4096	131072
Runtime (hrs)	93	138	141	43
Injections	299741	290580	436947	6340
Resets	3444	1584	651	415
Toggle	580	1350	513	221
Direct	2863	233	137	190
MIBR	87	183	671	15

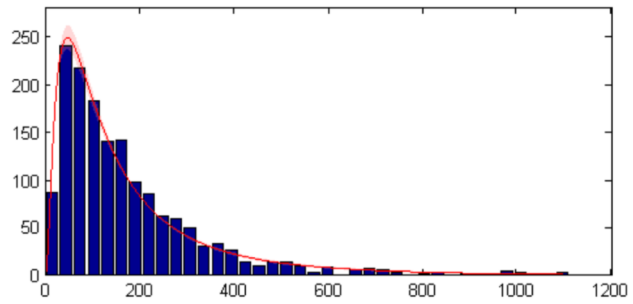

Fig. 10. Reset statistics of BSS injections showing the distribution for the number of injections before a reset occurred. The distribution is fitted to a log-normal survival function.

ming adaptor. However, performing flash erasure with significantly reduced supply voltage of the target microcontroller allowed recovery. It is assumed that undesired code executions alters the state of the JTAG interface preventing its EEM from being accessed via JTAG.

Further, statistical analysis of the system response logs showed that not all memory regions show the same sensitivity to software hangs, resulting in a microcontroller toggle, and direct resets, typically caused by memory violations or illegal instruction fetches. As expected, the most bit error sensitive region is the active program memory showing both, direct resets and software hangs already after about 15 injections. The PER region is significantly more sensitive to direct resets, mainly caused by register password violations while injecting the BSS region mostly results in system hangs, probably caused by information corruptions in the schedulers statically allocated memory. The most robust region with a mean number of injections before a reset occurred (MIBR) of 671 seems to be the stack. However, this number might be misleading as the average used stack size at runtime is only about 20 % of the reserved region so it can be assumed that many injections hit a currently unused address.

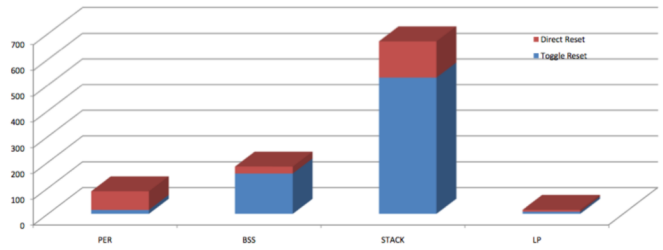

Fig. 11. Mean number of injections (MIBR) between resets.

Figure 11 summarizes the results by indicating the mean number of injections before a reset occurred (MIBR). The bars further indicate the relative distribution of observed toggle and direct resets.

5. CONCLUSIONS

This contribution describes the design and test results of an OBDH core module for picosatellites. The design focuses on robustness and power efficiency. Basing on COTS technology, the ultra-low power hardware design implements failure tolerance by a redundant microcontroller architecture with watchdog controlled master-slave topology. A simple runtime-efficient error detection and recovery algorithm allows mutual program memory protection against potential hazardous memory corruptions caused by SEEs. The system demonstrated robustness against more than 1 million bit errors injected to the system during extensive SWIFI test campaigns.

6. ACKNOWLEDGEMENTS

The authors appreciated the support for UWE-3 by the German national space agency DLR (Raumfahrt-Agentur des Deutschen Zentrums für Luft- und Raumfahrt e.V.) by funding from the Federal Ministry of Economics and Technology by approval from German Parliament with reference 50 RU 0901.

REFERENCES

J. Arlat, Y. Crouzet, J. Karlsson, P. Folkesson, E. Fuchs, and G.H. Leber. Comparison of physical and software-implemented fault injection techniques. *Computers, IEEE Transactions on*, 52(9):1115–1133, 2003. ISSN 0018-9340. doi: 10.1109/TC.2003.1228509.

Stephan Busch and Klaus Schilling. Uwe-3: A modular system design for the next generation of very small satellites. In *Proceedings of Small Satellites Systems and Services - The 4S Symposium, Portoroz (Slovenia)*, June 2012.

R. Micheloni, L. Crippa, and A. Marelli. *Inside NAND Flash Memories*. Springer Netherlands, 2010. ISBN 9789048194308.

Keith Quiring. Flash techniques: Design for system robustness. In *MSP430 Advanced Technical Conference*, 2008.

K. Schilling, M. Schmidt, S. Busch, and M. Drentschew. Uwe: A roadmap to pico-satellite formation flying. In *Proceedings of International Astronautical Congress, IAC-11.D1.4.5*, 2011.

M. Schmidt, K. Ravandoor, O. Kurz, S. Busch, and K. Schilling. Attitude determination for the pico-satellite uwe-2. In *Proceedings IFAC World Congress, Seoul*, pages 14036–14041, 2008.

R.R. Some, W.S. Kim, G. Khanoyan, L. Callum, A. Agrawal, and J.J. Beahan. A software-implemented fault injection methodology for design and validation of system fault tolerance. In *Dependable Systems and Networks, 2001. DSN 2001. International Conference on*, pages 501–506, 2001. doi: 10.1109/DSN.2001.941435.

Haissam Ziade, Rafic A. Ayoubi, and Raoul Velazco. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, 1(2):171–186, 2004.