# Design and Development of the Fault Tolerance Software for the OBC Subsystem of STUDSAT

Sreevani Nanjuri[1],Professor
Electronics & Communication Engineering Department
Nagarjuna College of Engineering & Technology
Bengaluru, India

Irfan Khan[2],Student
Electronics & Communication Engineering Department
Nagarjuna College of Engineering & Technology
Bengaluru, India

Krithika M[3],Student
Electronics & Communication Engineering Department
Nagarjuna College of Engineering & Technology
Bengaluru, India

Meghashree L[4],Student
Electronics & Communication Engineering Department
Nagarjuna College of Engineering & Technology
Bengaluru, India

Yashwanth P[5],Student
Electronics & Communication Engineering Department
Nagarjuna College of Engineering & Technology
Bengaluru, India

**Abstract:- STUDent SATellite (STUDSAT) is India's first Twin - Nano satellite project which aims to demonstrate in-orbit separation, inter-satellite communication and drag-sail mechanism. The mission life of this class of satellites is short mainly due to two reasons. Firstly, the constraints on the volume of the satellite limit the scope of energy capacity which in turn degrades the survivability of the satellite. Also, the adverse and hostile environment in space necessitate the need for space-grade components in the design. Hence there is necessity for a system which can withstand itself in faulty environment and thereby increasing mission life of satellite.**

**In this project we are implementing the Fault Tolerance Software for OBC Sub-system, the complete satellite system is divided into six subsystems out of which, the On Board Computer (OBC) subsystem plays an important role in the functioning of the satellite. Even if there is a malfunction in this system, it might result in failure of the entire mission. So, a fault tolerant design is developed by introducing fault handling methods and redundancy to both software and hardware. The redundancy architecture of the system is implemented at three different levels viz. system level, subsystem level and interface level to improve the reliability of the OBC of the satellite.**

***Keywords:-*** *STM32F407VG Discover Board, CAN ,UART.*

## I. INTRODUCTION

The STUDSAT-2 consists of two Nano satellites (STUDSAT-2A and STUDSAT-2B) each mass less than 5kgs with the dimensions of 30 x 30 x 10 cm. The satellites are separated in the orbit. The satellite STUDSAT-2A shall receive AIS messages from ships and sends the received message, telemetry data to STUDSAT-2B through Inter Satellite Link (ISL). STUDSAT-2A carries AIS system and beacon data relay with ISL facility while STUDSAT2B has beacon data relay system, Drag-Sail and ISL. Along with these payloads, each satellite hosts an On-Board Computer (OBC) subsystem, Attitude Determination and Control System (ADCS) subsystem, Communication subsystem, Command and Data Handling(C&DH) subsystem, Mechanical subsystem and an Electrical Power subsystem (EPS) that are required for a successful space mission. The satellite was designed to send the images and telemetry data to the ground station NASTRAC (Nitte Amateur Satellite Tracking Centre) NASTRAC, the ground station, was also developed at NMIT itself by students, which like the STUDSAT-1, was the first-of-its-kind.

The satellite system of STUDSAT-2 is shared a six sub-system i,e. Structure, Attitude Determination and Control System (ADCS), Electrical Power System (EPS), Command and Data Handling(C&DH), Communication and Payload. The OBC is a special-purpose on board computer, and is different in many respects from the conventional PCs and work stations. An on-board computer designed to monitor and control the real time operations of a satellite. It takes control of the satellite immediately after its launch, and remains in command until the end of the operation life of the satellite. The primary role of the OBC is an autonomous control of all subsystems of the satellite. The OBC consists of complex hardware and software to communicate with the on-board subsystems and the ground station. When the satellite is not in the cone of window of the ground station, all critical operational decisions and actions are taken by the OBC. It collects information from different subsystems of the satellite, analyses the information, and takes necessary and appropriate decisions as and when required. When the satellite is in cone of window, the telemetry data can be downloaded by commanding the satellite from ground station. The On Board Computer (OBC) which tightly integrates command data handling (C&DH) and other subsystems like communication subsystem, payloads, ADCS, deployments system, EPS. OBC is a special purpose on board computer designed to monitor and control the real time operations of a satellite. It takes control of the satellite immediately after launch and remains in command until the end of the operation life of the satellite. The primary role of OBC is an autonomous control of all subsystems of the satellite.

## II. RELATED WORK

First, In paper "How to validate software-side fault tolerance measures for spaceflight ." by, C. M. Fuchs [1], it describes about how realistic and systematic validation of software-implemented FT concepts can be conducted using ISA-level fault injection for space applications, and fields with a similar fault profile, example critical and irradiated environments. This is done based on the fault injection campaign which has been carried out to validate a novel thread-level coarse grain lockstep concept , developed for space applications. This paper includes not only concept validation but is meant as a template for other researchers who wish to validate their own software-implemented FT concepts. It provides a detailed description of the fault profile in the space environment, and a through description of the utilized tools and scripts, which have been made available to the public open source. Thereby, we hope increase acceptance of software implemented FT concepts by industry, but also to increase the share of concepts that are validated in a practically meaningful way.

In paper "Software-based fault recovery via adaptive diversity for COTS multicore processors" by Andrea Holler [2], has described the ever growing demands of embedded systems to satisfy high computing performance and cost efficiency lead to the trend of using commercial off-the-shelf hardware. However, due to their highly integrated design they are becoming increasingly susceptible to hardware errors (example caused by radiation-induced soft-errors or wear-out effects). Since such faults cannot be fully prevented, systems have to cope with their effects. At the same time there is the trend of multi-core processors in embedded systems. Approaches to achieve fault tolerance by using the multiple cores to establish redundancy have been presented in literature. However, typically only homogeneous redundancy techniques are considered to tolerate soft errors. However, there is a lack of appropriate reaction mechanisms for restoring the system in case of permanent hardware faults. Here, we propose the basic idea of enhancing multi-core redundancy techniques with a cost-efficient automated introduction of diversity in the executed software replicas. Recently, these automated software diversity techniques have attracted attention in the security domain. We propose to use these techniques to recover from permanent hardware faults. This is achieved by adapting the software execution in such a way that permanent faults are mitigated.

In paper "The N-Version Approach to Fault Tolerant Software," by Avizienis [3],Evolution of the N-version software approach to the tolerance of design faults is reviewed. Principal requirements for the implementation of N-version software are summarized and the DEDIX distributed supervisor and test bed for the execution of N-version software is described. Goals of current research are presented and some potential benefits of the N-version approach are identified. Index Terms-Design diversity, fault tolerance, multiple computation, N-version programming, N-version software, software reliability, tolerance of design faults.

In paper "Experimental analysis of binary-level software fault injection in complex software," by D. Cotroneo et al [4], using Off-The-Shelf (OTS) software components are the cornerstone of modern systems, including safety-critical ones. However, the dependability of OTS components is uncertain due to the lack of source code, design artifacts and test cases, since only their binary code is supplied. Fault injection in components' binary code is a solution to understand the risks posed by buggy OTS components. In this paper ,we consider the problem of the accurate mutation of binary code for fault injection purposes. Fault injection emulates bugs in high-level programming constructs (assignments, expressions, function calls,) by mutating their translation in binary code. However, the semantic gap between the source code and its binary translation often leads to inaccurate mutations. We propose Faultprog , a systematic approach for testing the accuracy of binary mutation tools. Faultprog automatically generates synthetic programs using a stochastic grammar, and mutates both their binary code with the tool under test, and their source code as reference for comparisons. Moreover, we present a case study on a commercial binary mutation tool, where Faultprog was adopted to identify code patterns and compiler optimizations that affect its mutation accuracy.

In "Fault Tolerance in Concurrent Object-Oriented Software through Coordinated Error Recovery", by Xu, J, Randell, B, Romanovsky, A, Rubira, C.M.F, Stroud, R.J, Wu, Z ,in Proceedings of the 25th International Symposium on Fault-Tolerant Computing Systems (FTCS-25) [5], This paper presents a scheme for coordinated error recovery between multiple interacting objects in a concurrent object oriented system. A conceptual framework for fault tolerance is established based on a general object concurrency model that is supported by most concurrent object-oriented languages and systems. This framework integrates two complementary concepts conversations and transactions. Conversations (associated with cooperative exception handling) are used to provide coordinated error recovery between concurrent interacting activities whilst transactions are used to maintain the consistency of shared resources in the presence of concurrent access and possible failures. The serialisability property of transactions is exploited in order to help prevent unexpected information smuggling. It has been discussed about the problem of providing fault tolerance in concurrent (object oriented)OO software systems and propose a general framework for fault tolerance that integrates two complementary concepts, conversations and transactions. Our framework encompasses strategies for dealing with hardware, software and environmental faults to provide coordinated error recovery between a set of interacting objects.

## III. METHODOLOGY

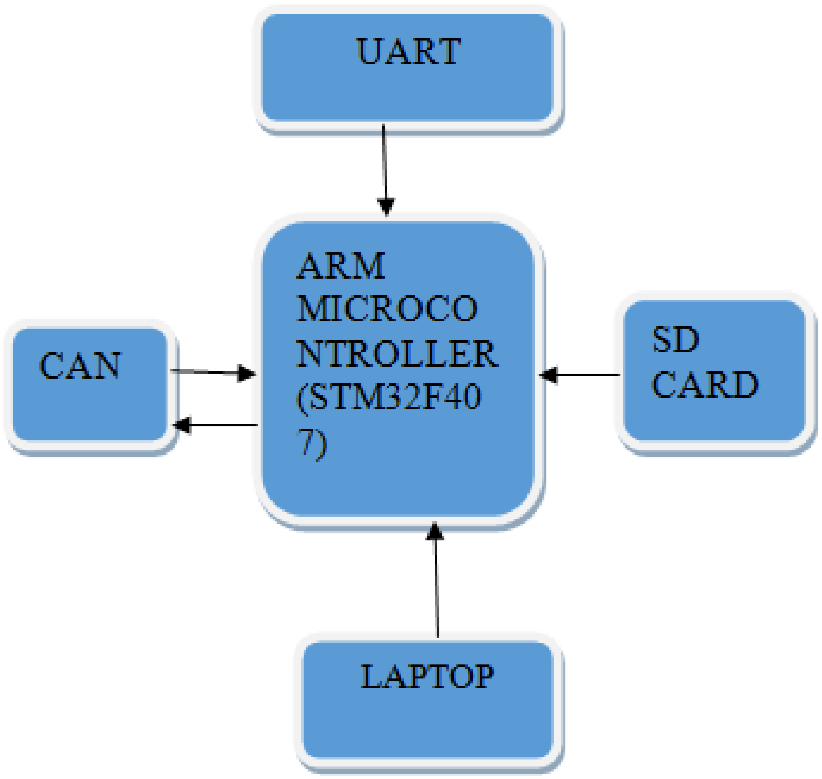

Fig 1: Fault Tolerance Software for OBC Subsystem

The Fault tolerance software for OBC subsystem in fig 1 shows the layout of the Fault tolerance software for OBC subsystem which is designed to be fault tolerant with respect to hardware as well as software. Fault Tolerant model of the system is realized by:

- First detecting the faults and errors in the system.
- Then handling those faults and errors.
- Finally providing redundancy to restore the system.

The software faults and errors in the system are handled by exception handling techniques to transfer the interrupt control to continue with the main sequence of the satellite. Important considerations are taken during exception handling to decrease interrupt latency, context switching period and stack frame size.

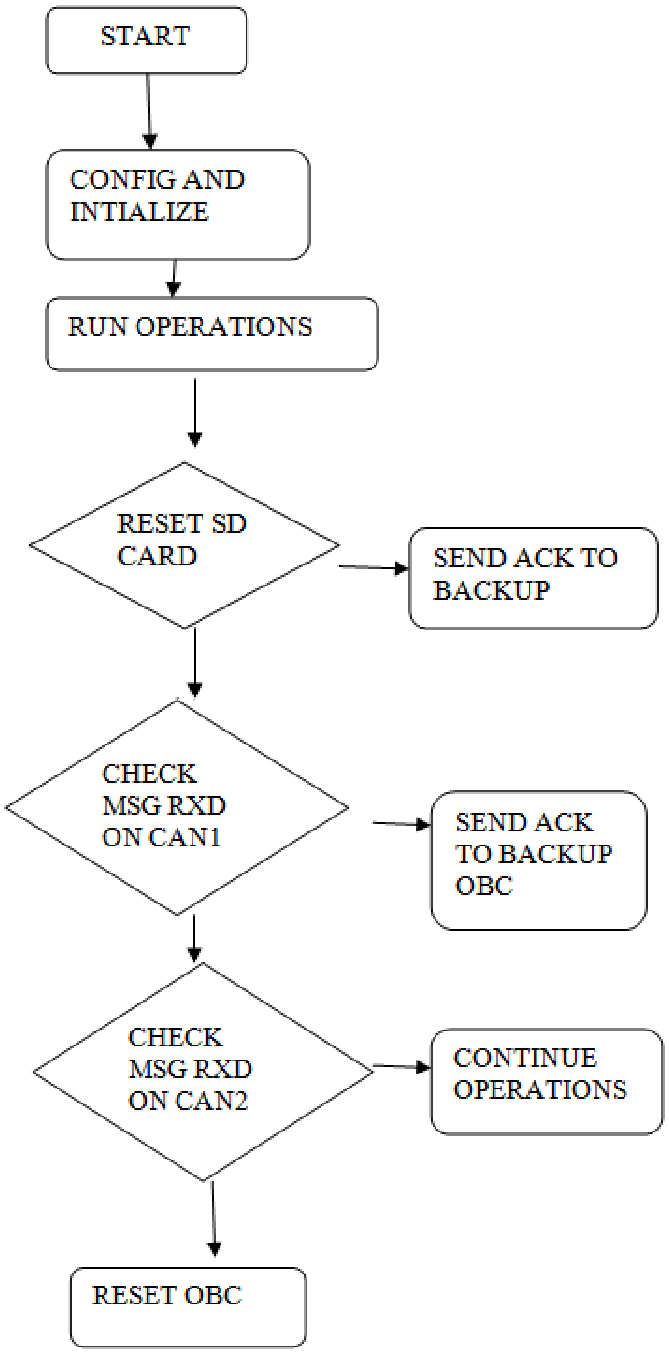

Fig 2:- Data flow diagram

## IV. WORKING

Using SD card in case of unexpected / unpredictable behavior. The priority for handling these fault exceptions are defines in interrupt vector table during the startup phase of the OBC. The hardware faults in the subsystem are handled by resetting the system providing functional redundancy to system, subsystem and communication interfaces and reconfiguring the redundant system as master. The hardware redundancy to an OBC is provided by an identical OBC architecture. Upon detection of hardware fault of master OBC the backup OBC will reconfigure as master OBC and perform satellite operations.

Initially OBC powers on, initialization and configuration of all the peripheral interfaces happen, then main mission sequences are executed, after this SD card is periodically refreshed by the OBC and continues mission operation. When refresh of SD card is failed, it will check for hello message from redundant OBC and CAN1 and upon reception of message, it will send acknowledgement message on the same line. If the message reception on CAN1 is failed, master will check the hello message on CAN2 line and acknowledges after the reception. Whenever it receives message from CAN1 on the failure of primary controller, the secondary controller will be in standby mode using the backup of SD card, while the primary controller is in sleep mode.

## V. CONCLUSION

In this project we are proposing the Fault Tolerance software for OBC subsystem of Nano Satellite (STUDSAT-2).Whenever there is a malfunction in the system, it might result in failure of the entire mission ,so a fault tolerant design is developed by introducing fault handling methods and redundancy to both software and hardware. The redundancy architecture of the system is implemented at three different levels viz. system level, subsystem level and interface level to improve the reliability of the OBC of the satellite. In this system it detects errors and gives backup to the subsystem. The Backup system will be shifted to standby mode whenever it encounters failure in the primary system. Using this backup system errors will be rectified. It implements a software backup system using SD card.

## REFERENCES

[1]. C. M. Fuchs et al., “How to validate software-side fault tolerance measures for spaceflight by example.” submitted, 2018

[2]. A. Höller et al., “Software-based fault recovery via adaptive diversity for COTS multicore processors,” 2015, arXiv:1511.03528.

[3]. Avizienis, “The N-Version Approach to FaultTolerant Software,” IEEE Trans. Soft. Eng., vol. SE-11, no.12, pp.1491-1501, 1985.

[4]. D. Cotroneo et al., “Experimental analysis of binary-level software fault injection in complex software,” in 2012 Ninth European Dependable Computing Conference. IEEE, 2012

[5]. Xu, J.; Randell, B.; Romanovsky, A.; Rubira, C.M.F.; Stroud, R.J.; Wu, Z.: “Fault Tolerance in Concurrent Object-Oriented Software through Coordinated Error Recovery”, in Proceedings of the 25th International Symposium on Fault-Tolerant Computing Systems (FTCS-25), pp. 499–509, Pasadena, California, 1995