A Modular Command and Data Handling System Concept for Small Satellites

Abdullah Alsalmani[1], Mohammed Almehrezi[1], Abdul-Halim Jallad[1,2,*]
[1]*National Space Science and Technology Center (NSSTC)*
[2]*Department of Electrical Engineering*
United Arab Emirates University
Al Ain, United Arab Emirates
*Corresponding Author: a.jallad@uaeu.ac.ae

***Abstract*—The increasing demand for high-performance computer systems on board space missions applications, such as artificial intelligence and machine learning, are drivers of the need for higher on-board command and data handling specifications. Consequently, a modular Command and Data Handling Subsystem (CDHS) is required in order to manage the complexity of future CDHS hardware. Some systems have applied a similar approach; however, they either focus on large scale satellites or have a certain level of complexity that limits their usage by learners. This paper proposes a modular Command and Data Handling System that includes several layers providing more flexibility in design at a low cost. The system also potentially allows the integration of CDHS initially used in CubeSats in larger-scale satellites. Also, Through this paper highlights the interfacing options used in the proposed design, including the regular command and data interfaces, wireless command and data interfaces, high-speed data interfaces, and power interfaces. The results validate the usefulness and potential of the proposed system as a modular Command and Data Handling System that can be used in future missions. It also opens the door for further development, allowing the low-cost development of Command and Data Handling Systems.**

***Index Terms*—Space, Satellite Technology, Command and Data Handling Subsystems, Modularity.**

I. INTRODUCTION

In recent years there has been increasing demand for high-performance computer systems on board space missions applications, such as artificial intelligence and machine learning are drivers of the need for higher on-board command and data handling specifications. Consequently, a modular Command and Data Handling System (CDHS) is required in order to manage the complexity of future CDHS hardware. The Command and Data Handling System acts as the brain of the satellite, controlling its functions and operations; where some of its functionalities are managing data on the spacecraft, preparing data for transmission to earth, carrying out command maneuvers, and autonomously monitoring and responding to a wide range of on-board problems that might occur [1]. Currently, Command and Data Handling System (CDHS) hardware architectures have a limited ability to scale with mission needs. However, several modular standards began to introduce the concept of modularity in space applications where they aim to reduce the satellite development time and expenses.

In this project, we aim to design a modular Command and Data Handling architecture that allows the integration of up to four layers of modularity.

The main contribution of our work is an multi-level modular CDHS. To achieve this goal, we also propose: (1) designing modular hardware modules for each layer (2) defining standard physical properties of the modules, and (3) defining the standard interfaces between the modules.

The remainder of the paper is organized as follows: Section II presents our literature review, Section III presents our proposed system, Section IV discusses our experimental testing & results, and Section V concludes the paper.

II. LITERATURE REVIEW

The Space Plug-and-Play Avionics was one of the first approaches, where the sophisticated hardware and software concepts are combined to make plug-and-play possible. Due to the emergence of third-party USB developers, which has been proven in the personal computer market, the SPA has given a chance for developers to implement SPA in a custom way or to use pre-developed solutions as needs dictated. It provides a unified method for self-discovery and self-configuration of heterogeneous PnP networks, as it offers an elegant, robust and practical approach [2]. SPA components get allowed to communicate without prior knowledge of the corresponding component's location on the system or the type of interconnection network it uses. This has also opened the chance for developers to build Plug-and-Play Satellites (PnPSats) without having every component (or maybe any components) on one's shelf. Their PnP components would appear as selectable options in a rapid design flow. Also, they have created the Mission Spacecraft Design Tool in the responsive testbed as the first step for PnPSat, and a step towards the ultimate push-button toolflow [3].

Another modular standard is the Modular Architecture for Robust Computing (MARC) project, which aims to demonstrate the essential features of a heterogeneous, fault-tolerant, high availability distributed avionics system based on a SpaceWire network [4]. The MARC SpaceWire network architecture is built around a High Flexibility Cluster (HFC), which consists of two 8-port routers that provide redundancy and each module interfaces to both routers to allow for the failure of one router within the cluster and four modules that can be any function (processing, memory, I/O, etc.) that requires a network interface. The HFC has four spare ports, each with a redundant SpaceWire (SpW) link. These spare ports can connect clusters or other system components like the TMTC system or EGSE. Furthermore, several HFCs can be linked together in various topologies, and if increased bus bandwidth is required, the number of 8 port routers within a cluster can be doubled or increased to a larger number [5].

A third modular standard usually used in aircrafts is Integrated Modular Avionics (IMA). IMA defines an integrated system architecture that preserves the federated architectures' fault containment and 'separation of concerns' properties, where independent functional chains share a common computing resource. Each functional chain, or application, is protected against interference from other chains by a memory protection strategy and exclusive guaranteed access to the computing resources. Applications are isolated from each other in time and memory using software partitions and communicate through controlled channels [6].This has led the European Space Agency (ESA) to discuss the potential of using IMA in space by introducing the Integrated Modular Avionics for Space (IMA-SP). Where the software partitioning technology can be integrated into the spacecraft flight software architecture to improve the reliability and security of space systems, as well as the efficiency of the software development and validation processes [7].

Table I shows a comparison between the three currently available modular architectures mentioned above.

Several typical command and data interfaces are used internally on On-Board Computers; however, two transmission standards (switched topology) stand out among the rest, the Ethernet and SpaceWire, each having its own advantages, trade-offs, and specific uses.

Ethernet is a long-established technology that forms the basis of local area communication, as it can easily encompass thousands of users within a corporate or a facility. Ethernet was known for providing an interconnection between computers, but it gradually expanded to interconnect appliances and other personal devices. The Ethernet's evolution to include higher bandwidth, improved medium access control, and different physical media, as well as the replacement of the coaxial cable with a point-to-point link connected by Ethernet repeaters or switches, has led the Ethernet to start to rapidly replace legacy data transmission systems in the world's telecommunications networks. Typically, Ethernet transmits data up to 10 Mbits/s, but it can reach 10 Gbits/s, with a power consumption of slightly less than 1 W [8].

On the other hand, SpaceWire is a new technology discovered in 1995, and it is characterized by its ability to build highly fault-tolerant networks and systems. Point-to-point data links and routing switches can be used to build networks that suit particular applications. In contrast, the application information is sent along with a SpaceWire link in discrete packets. The SpaceWire standard can facilitate the construction of high-performance on-board data handling systems, help reduce system integration costs, promote compatibility between data handling equipment and subsystems, and encourage the re-use of data handling equipment across several different missions. Also, SpaceWire's transmission data rate ranges from 2 to 200 Mbits/s, with a power consumption of about 750 mW [9].

III. PROPOSED SYSTEM

A. System Overview

This research presents a modular Command and Data Handling System that provides interface ability, where each module should be seamlessly interfaced to all other modules, and scalability, where the system should be expandable based on the required configuration and complexity of the satellite.

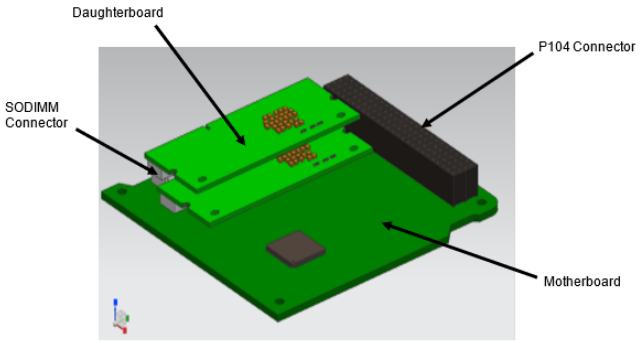

Fig. 1. Proposed modular concept approach - daughter-board and motherboard and their connector.

TABLE I
COMPARISON BETWEEN CURRENTLY AVAILABLE MODULAR STANDARDS

	SPA	MARC	IMA
Voltage Interfaces	– SPA-O = 28v @ 30 A max – SPA-S(LV) = 5v @ 2 A – SPA-U = 5 V (2-pin)* @ 4.5A – SPA-1 = 5 V (2-pin)* @ 2 A	3.3v and 5v is supplied to the components (Additional 1.8v can also be supplied.)	
Data Interfaces	– SPA-O = Part of SPA-S [Up to 10-Gbit/sec] – SPA-S(LV) = low voltage (5V) spacewire [400-600 Mbit/sec] – SPA-U = similar to USB [Up to 12 Mbit/sec] – SPA-1 = similar to I2C [Up to 10 kilobit/sec]	– UART – CAN – I2C – SLINK – SpaceWire	Ethernet and AFDX
Software	**xTEDS** (extended Transducer Electronic Datasheets)	**GenFas Software** (based on CCSDS-SOIS Architecture) including FDIR Manager	**ARINC 653** "Avionics Application Software Standard Interface" operating system specification
Scale of projects	Nano and Small Scale Satellites	Small and Medium Scale Satellites	Aircrafts
Fault Tolerance Scheme	– Health Check Monitoring – Error detection: - Hardware (primary command received before previous one processed; attempt to write into protected memory without override; processor sequencer reached an illegal state.) - Software (Primary output unit unavailable for more than 14 seconds, self-test routine not successfully completed; output buffer overflow.) - Special Reed-Solomon coding	– Health Check Monitoring – Fault Detection – Fault Diagnosis and Identification – Fault Recovery	– Health Monitoring Mechanism [The fault gets processed at Partitioning Kernel level or virtualised and channelled to the appropriate System or Application Partition.] – An application can be instantiated more thanonce, creating redundancy in the system.

As shown in Figure 1, the system consists of up to four layers, providing higher modularity. The four layers are defined as the following:

- Layer 1: A micro-module designed for a specific application. It is mounted on layer 2 boards.
- Layer 2: Boards with SODIMM or Bergstak connectors can also include the mounting of layer 1 modules on them.
- Layer 3: The motherboard that hosts the Layer 2 board and interfaces with it using one of the connectors (SODIMM, Micro-DIMM or other high-speed Mezzanine connectors). It follows the PC104 standard that is commonly used in CubeSats.
- Layer 4: It is an external motherboard that usually hosts the Layer 3 board and interfaces with it using a high-speed Mezzanine connector. The layer 4 boards usually come in larger form factors (3U, 6U, etc.).

The four layers successfully interface with each other using several typical command and data interfaces, wireless command and data interfaces, high-speed data interfaces, and power interfaces.

1) Typical Command and Data Interfaces: There are four different typical command and data interfaces used in our design. As illustrated in Table II, we can identify that in terms of speed, SPI has the highest, CAN bus comes next, then UART and I2C. While in terms of data rate and bus length, CAN bus and SPI have the same data rate at the same bus length, then I2C comes next, and UART has the lowest data rate but the highest bus length. Furthermore, the I2C is half-duplex while the rest are full-duplex. The rest of the details are included in the table.

TABLE II
COMPARISON BETWEEN REGULAR COMMUNICATION INTERFACES

	I2C [10] [11]	UART [10] [11]	SPI [10] [11]	CAN [12]
Speed	3.4 Mbps	5 Mbps	60 Mbps	1 Mbps
Data rate and Bus Length	1Mbps @ 0.5m	20kbps @ 15m	25Mbps @ 0.1m	25Mbps @ 0.1m 10 kbps @ 1 km
Protocol (Sync / Async)	Synchronous	Asynch-ronous (Serial protocol)	Synchronous	Synch-ronous (Multi master protocol.)
Duplex	Half duplex	Full duplex	Full duplex	Full duplex
Types of Lines / Ports	Two Lines: SCL (serial clock line) SDA (serial data line acceptance port)	Two Lines: TX RX	Four ports: MOSI, MISO, SCLK, and NSS	Two Lines: CAN High CAN low

2) Wireless Command and Data Interfaces: There are three different wireless command and data interfaces used in our design. As illustrated in Table III, we can identify that the WiFi has the highest data rate; however, it has the highest power consumption rate. While LoRa has the most extended transmission range but the lowest data rate.

TABLE III
COMPARISON BETWEEN WIRELESS COMMUNICATION INTERFACES

	Zigbee	LoRa	WiFi
Data Rate	- 20 kbps (868 MHz band) - 40 kbps (915 MHz band) - 250 kbps (2450 MHz band)	- 0.3 to 22 kbps (LoRa Modulation) - 100 kbps (using GFSK)	54 Mbps using 802.11a/g OFDM Technique
Range	10 to 100 meters	- 2 to 5 Km (urban areas) - 15 Km (suburban areas)	30 to 100 meters
Power Consumption	0.39 W	0.50 W	3 W
Standard (IEEE)	802.15.4	802.15.4g	802.11
Application	LP-WAN	WAN	LAN

3) High Speed Data Interfaces: In most satellites used nowadays, the SpaceWire data interface is used; therefore, we wanted to compare it with its alternatives to choose the best for our design. As illustrated in Table IV, we can identify that Ethernet has a higher data rate and speed than SpaceWire and relatively the same power consumption rate; however, SpaceWire is more developed in space, and it has a non-limited packet size, but Ethernet's minimum packet size is 64 bytes.

TABLE IV
HIGH SPEED DATA INTERFACES

	Ethernet	SpaceWire
Speed	10 Mbps to 400 Gbps	2 to 200 Mbps
Duplex	Full-duplex	Full-duplex
Links	Point-to-point	Point-to-point
Packet	- Ethernet frame contains Preamble and SFD. - Ethernet header contains Source, Destination MAC address and payload. - Last field is CRC which is used to detect the error. - Minimum packet size is 64 bytes.	- A SpaceWire packet shall comprise one or more data characters followed by an end of packet marker (EOP) or error end of packet marker (EEP). - Packet size is not limited.

4) Power Interfaces: Another essential type of interfaces besides the data interfaces is the power interfaces to successfully interfacing between our layers. Our design allows the power regulation in different parts of each layer to provide the highest level of modularity and flexibility in the design, providing voltages that range from 3.3v up to 12v. Table V shows what voltage ranges are included in every layer.

TABLE V
POWER INTERFACES

	Power Interfaces
Layer 1	3.3v
Layer 2	3.3v
Layer 3	5v and 3.3v
Layer 4	12v, 5v, and 3.3v

B. System's Prototype

For easy demonstration and representation of our concept, we will consider discussing Layers 1, 2, and 3. Figure 2 shows the individual modules of Layers 1, 2, and 3, while Figure 3 depicts the detailed hardware design of the Layers 1, 2, and 3 mounted over each other. The main components used for the system's prototype are discussed below:

1) Layer 1: As stated earlier, layer 1 is a micromodule designed for a specific application. As illustrated in Figure 3, we used the Xbee 3 TH module as an example for technology demonstration purposes. The Xbee 3 modules are micromodules made by Digi that follow the Zigbee protocol and are ideal for low-latency and predictable communication timing applications. They also provide quick, robust communication in point-to-point, peer-to-peer, and multipoint/star configurations. They are also supported by the MicroPython and Digi XCTU software tools that simplify the process of adding functionality, configuration, and testing, which aligns with our goals mentioned in this paper.

2) Layer 2: This layer is mounted on layer 3 and provides interfacing with layer 1. In this example, we added

female headers on the layer 2 modules to mount the layer 1 module (Xbee 3 TH module) and use it as a bridge between layer 1 and layer 3.

3) Layer 3: This layer is our On-Board Computer, containing the memories, connectors, and voltage regulators. It also contains the connectors used to mount layer 2 and allows the OBC to be mounted on layer 4. In our design, we choose the Bergstak Mezzanine Board-to-Board connector to mount the layer 2 modules, as we found that it is more reliable and provides better stability for the board. Our layer 3 board's connector provides the following interfaces to allow further modularity:

TABLE VI
LAYER 3 INTERFACES

	Number of interfaces
UART	4
I2C	3
SPI	3
CAN	2

As illustrated in Table VI, we have the following available interfaces that can be used when designing the layer 2 boards: four UART interfaces, three I2C interfaces, three SPI interfaces, two CAN interfaces.

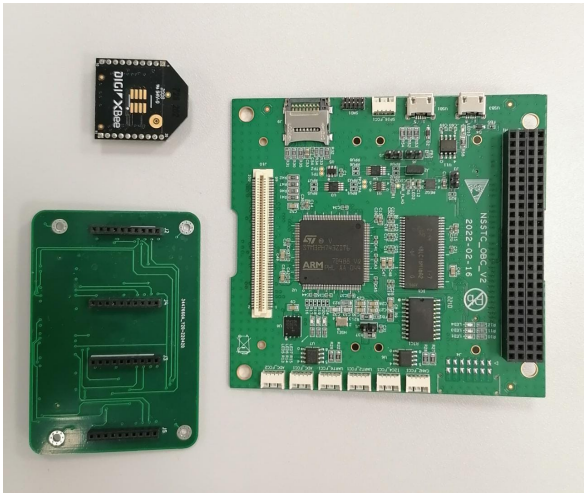

Fig. 2. The three separate layers used in the proposed design

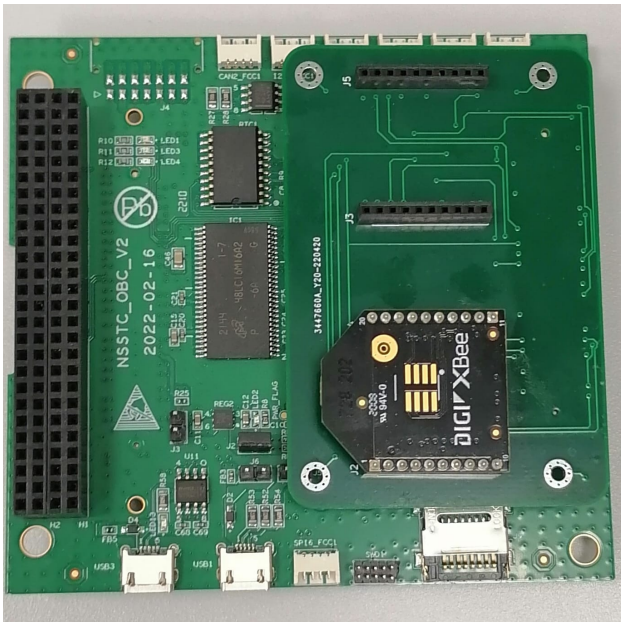

Fig. 3. Implementation of the three layers of the modular design

IV. EXPERIMENTAL TESTING AND RESULTS

A. Experimental Testing

The system was tested, in order to verify the operations of the all functional units and also the interfaces between the respective modules. We started by testing the interfacing between the layer 3 components, and we created a specific code that automatically does a full health check for the components. Figure 4 shows a practical demonstration of the code.

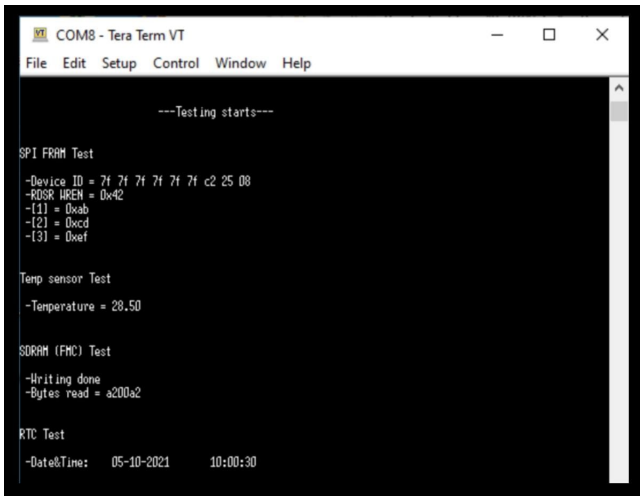

Fig. 4. Part of Layer 3 full functional test

Next, we tested the interfacing between layers 1 and 2 using a connectivity test, and we proceeded towards testing its interfacing with layer 3. Finally, we tested the interfacing between the CDHS and other satellite systems, which proved

that the proposed modular system is fully functional and can be tested in space.

V. CONCLUSION

To conclude, a modular Command and Data Handling system is proposed to reduce the satellite development time and expenses and satisfy the increasing demand for high-performance computer systems on-board space missions applications. This system includes four layers, a board that allows the usage of CDHS used in CubeSats in larger-scale satellites, a board that can be used in CubeSats as the OBC, a board that has SODIMM, Micro-DIMM or other high-speed Mezzanine connectors usually either functions on its own or provides interfacing with a smaller module that does a specific function and it is considered as Layer 1. This paper also highlights the interfacing options used in the proposed design, including the regular command and data interfaces, wireless command and data interfaces, high-speed data interfaces, and power interfaces. We also discuss several examples of each of the layers. The proposed system has been tested in different settings, proving its efficiency and ease of use. It can be deployed at a low cost, allowing for the development of a modular Command and Data System that can be used in different space applications.

ACKNOWLEDGMENT

The authors would like to thank the UAE Space Agency for sponsoring this project.

REFERENCES

[1] Johl, Shaina. (2014). A Reusable Command and Data Handling System for University CubeSat Missions.
[2] "Space Plug-and-Play Avionics", The Air Force Research Laboratory, 2021.
[3] M. Martin, D. Fronterhouse and J. Lyke, "The Implementation of a Plug-and-play Satellite Bus", 22nd Annual AIAA/USU Conference on Small Satellites, 2008.
[4] Modular Architecture for Robust Computation Session: SpaceWire On-board Equipment and Software - Short Paper, Dr. O. Emam, T. Jorden and R.Knowelden
[5] A. Senior, P. Ireland, S. D. Fowell, R. Ward, O. Emam and B. Greene, "MODULAR ARCHITECTURE FOR ROBUST COMPUTING (MARC)", 2008.
[6] Prisaznuk, P.J.. (1992). Integrated modular avionics. 1. 39 - 45 vol.1. 10.1109/NAECON.1992.220669.
[7] Silva, Cláudio & Cristóvão, João & Schoofs, Tobias. (2012). An I/O Building Block for the IMA Space Reference Architecture.
[8] Vinogradov, Alexey & Yablokov, Evgeni & Yachnaya, Valeria. (2019). Upgrade of Ethernet-SpaceWire Protocol. 486-492. 10.23919/FRUCT.2019.8711937.
[9] Cook, Barry & Walker, Paul. (2007). Ethernet over SpaceWire—Hardware issues. Acta Astronautica. 61. 243-249. 10.1016/j.actaastro.2007.01.007.
[10] Zibayiwa, Morelife. (2021). A Review on The Inter-Processor Communication: I2C, UART, and SPI interfacing techniques.
[11] Shanthipriya, S. & Lakshmi, S.. (2017). Design and verification of low speed peripheral subsystem supporting protocols like SPI, I2C and UART. ARPN Journal of Engineering and Applied Sciences. 12. 7386-7391.
[12] Rietz, S. & Schneider, B. & Bender, S. & Fischer, W.-J & Grätz, H. & Heinig, Andreas. (1997). Multisensor signal processing with CAN bus interface. Sensors and Actuators A: Physical. 62. 729-733. 10.1016/S0924-4247(97)01590-2.