Interfacing Architecture between Telemetry and On-Board Computer for a Nanosatellite

Abhishek Prasad
Team Anant
BITS Pilani
Rajasthan, India
f20180331@pilani.bits-pilani.ac.in

Yash Jain
Team Anant
BITS Pilani
Rajasthan, India
yashjain0333@gmail.com

Neelanchal Joshi
Team Anant
BITS Pilani
Rajasthan, India
neelj99@gmail.com

Nishant Gupta
Team Anant
BITS Pilani
Rajasthan, India
guptanishant113@gmail.com

Varsha Singhania
Team Anant
BITS Pilani
Rajasthan, India
f20170563@pilani.bits-pilani.ac.in

Yatharth Sreedharan
Team Anant
BITS Pilani
Rajasthan,India
yatharthsreedharan@gmail.com

Abstract— **Team Anant, a student team is developing a 3U Nanosatellite with a hyperspectral camera as the payload. The proposed satellite is divided into multiple subsystems. Linking and coordination amongst them have been done based on satellite constraints. The satellite follows a combination of centralized and distributed architecture. This paper details the architecture of the interface between the On-Board Computer(OBC) subsystem and the Telemetry Tracking and Command (TTC) subsystem. The Telemetry microcontroller handles data downlink and error control. OBC handles payload camera operations and image compression. To make the system full duplex, one of the cores on OBC is dedicated to the uplink circuitry. Hence, re-transmission requests and acknowledgements from the ground station are received by OBC. This necessitates a proper interface between On-Board SoC and telemetry microcontroller which is done using the UART protocol. The On-Board Computer of the satellite acts as the primary source of all the commands and monitors the status of the various subsystems. Its architecture performs hyperspectral image compression on a Field Programmable Gate Array (FPGA) before the image gets downlinked to the ground station. The algorithm used for the same is CCSDS 1.2.3. Storage and downlinking of payload and housekeeping data are aided by a subsystem specific interface architecture. The image is transferred from Payload to an on-board memory from where it is fetched by the FPGA in tiles for compression. The compressed data is pushed in the FIFO and stacked in its buffer. Subsequently, the telemetry microcontroller pops out the data whenever it is required to be packaged before downlinking. Therefore, FIFO hardware guarantees a sequential flow of data from OBC to telemetry without explicitly knowing the memory addresses of the compressed image. Moreover this FIFO memory is also used for sending housekeeping data. Thus, this paper will be elucidating the concepts of interfacing between the different subsystem's microcontrollers with the management of shared memory.**

TABLE OF CONTENTS

1. INTRODUCTION

The work presented in this paper was conducted in association with Team Anant, a group of undergraduate students working to build a 3U CubeSat using commercially available off-the-shelf components. The CubeSat houses a hyperspectral imager as its primary payload, for the categorization of various phytoplankton in the ocean. The bandwidth provided for such satellites for communication is very low. Hyperspectral cameras pose two challenges, the first being large image cube size and the second being the power consumed by it. The former is a big concern because of the limited downlinking capabilities possessed by a nanosatellite, and to counter this, we implement a compression algorithm on board. To speed up the compression process and minimize the energy spent (higher power but smaller duration), computationally intensive compression algorithm are implemented using FPGAs. FPGAs are a matrix of many configurable logic blocks (CLBs) that allow parallel algorithms to be implemented on them efficiently. To effectively downlink the image, the recommended standard was chosen, called the CCSDS (Consultative Committee for Space Data Systems) 123.0.B.1 [2]. A Xilinx 7 Series FPGA is being used to implement the compression algorithm.

The paper starts with giving an overview of the two subsystems (On-Board Computer and Telemetry). Then the following section goes over the details of interfacing between the two subsystems. It explains why UART was chosen over other interfaces and proceeds by elucidating the protocol chosen as well as the need for choosing the mentioned data types. It also elaborates on the flow of data between the two subsystems using a FIFO memory.

2. ON-BOARD COMPUTER ARCHITECTURE OVERVIEW

The On-Board Computer of the satellite acts as the primary source of all the commands and monitors the status of the various subsystems. The satellite follows an amalgamation of centralized and distributed architecture. The architecture comprises a system-wide I2C bus to which various sensors like magnetometer, temperature sensor, Inertial Measurement Unit (IMU), etc. are interfaced. The collected data is used for maintaining an information log which is used as input to algorithms that aid in pointing and detumbling the satellite. This data is also downlinked by the telemetry to the ground station. A UART (Universal Asynchronous

Receiver/Transmitter) interface is used between the telemetry microcontroller and the on-board Processing System (PS) to exchange information regarding housekeeping data, payload data, RRQ (Repeat Request), etc. which will be dealt with in the later sections. Figure 1 shows that there is a need of a interface between telemetry and the OBC PS. Similarly, an SPI (Serial Peripheral Interface) interface is used between the Power Subsystem microcontroller and the On Board SoC since a large amount of housekeeping data will have to be exchanged at high rates.

The satellite functions on different modes of operation and can be modeled as a "finite state machine"(FSM), in which the modes are the states of the FSM and the state transitions are the changes in the operation modes that depend on certain health conditions and parameters as measured by the sensors. The states broadly fall under two categories, Normal and Emergency. Each state has a predetermined set of logical tasks to be run, which are abstracted as separate processes in the memory. State transitions take place by polling the health metrics of the satellite. However, hardware interrupts are implemented on selected peripherals which ensure an asynchronous switching to the Emergency States as a safety measure.

Software and Hardware Specifications

Zynq-7000 integrates a feature-rich dual-core ARM Cortex-A9 based Processing System (PS) and 28 nm Xilinx Programmable Logic (PL) in a single chip. The ARM Cortex-A9 CPUs are the heart of the PS and also includes an on-chip memory, external memory interfaces, and a rich set of peripheral connectivity interfaces. One of the cores of the dual-core PS will be dedicated to run TTC processes during downlinking.

A FPGA is used specifically for image compression as it provides optimized datapaths which can improve computationally intensive tasks and reduce the power and energy consumption as compared to General Purpose Processors (GPPs). Moreover, [2] it was found that in comparison with GPPs, FPGAs have been observed to improve performance by more than 15 times at 50% of the energy consumption. FPGAs are also more versatile and configurable when compared to Application Specific Integrated Circuits (ASIC). The image after compression is stored in a flash memory shared between the camera and the FPGA.

Petalinux, a Linux based operating system has been chosen for this satellite because of ease of development coupled with the availability of many client drivers, bus controller drivers, and fairly good documentation even for kernel-level programming. One of the most important reasons for using Petalinux was the availability of documentation and its popularity within the Xilinx community.

For software development, C is chosen as it is by far the most powerful and versatile language that provides just the right amount of abstraction for the development of highly specific applications with constraints on running time and determinism. The implementation of the compression algorithm on the FPGA is done using the hardware description language, Verilog. The simulations and packaging of the IPs are done with the help of the software, Vivado. The development and synthesis of the algorithm were done on the ZedBoard, a development board by Digilent.

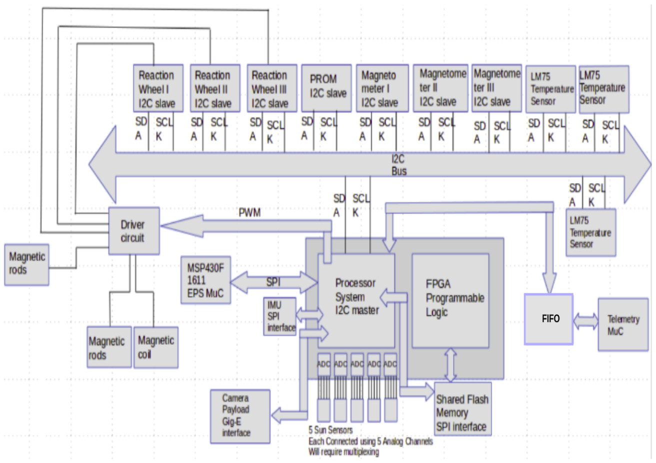

Reaction Wheel I I2C slave
Reaction Wheel II I2C slave
Reaction Wheel III I2C slave
PROM I2C slave
Magneto meter I I2C slave
Magnetometer II I2C slave
Magnetometer III I2C slave
LM75 Temperature Sensor
LM75 Temperature Sensor
SDA SCLK
I2C Bus
PWM
Driver circuit
Magnetic rods
Magnetic rods
Magnetic coil
MSP430F 1611 EPS MuC
SPI
IMU SPI interface
Processor System I2C master
FPGA Programmable Logic
ADC
LM75 Temperature Sensor
FIFO
Telemetry MuC
Camera Payload Gig-E interface
Shared Flash Memory SPI interface
5 Sun Sensors
Each Connected using 5 Analog Channels
Will require multiplexing

Figure 1. General architecture of the On-Board Computer subsystem

Summarizing the functions of the On-Board Computer subsystem

1. Determination of mode of operation and flow of control.
2. Storing the payload data.
3. Acquisition of the image and execution of the compression algorithm.
4. Acquiring housekeeping data from other subsystems.
6. Sending the data to the Telemetry for downlinking.
7. Processing of the Telecommand Data.
8. Acting as a bridge between telemetry and the Electrical Power Subsystem (EPS) to pass on information regarding malfunctioning components of the telemetry subsystem.
9. Executing attitude determination and control algorithms.

3. TELEMETRY ARCHITECTURE OVERVIEW

The Telemetry and Telecommand subsystem establishes the connection between the satellite and the ground station and is the only means of knowing the status of the satellite and communicating with it. It makes use of two separate transceivers for uplink and downlink to establish a full-duplex communication link. As discussed earlier, the Telemetry microcontroller is linked with the On-Board PS through a UART interface to receive specifications about the downlink data which is stored in memory shared between them. It further retrieves the data and processes them into packets, ready to be transmitted. To initiate a link with the ground station, the Telemetry microcontroller is interfaced with the downlink transceiver and supervises the flow of beacon, housekeeping and payload data (refer figure 2).

Apart from supervising the on-board communication system, the telemetry subsystem also maintains a ground station to complete a two-way communication. The goal of the ground station is to be able to communicate with the satellite without human interference. This means that the ground station must be able to receive data from the satellite and send some predetermined commands, TLE (Two-Line Element), re-transmission requests and other data to the satellite.

Software and Hardware Specifications

The telemetry subsystem makes use of the MSP430F5529 microcontroller because of its very low power consumption as nanosatellites have strict power constraints. It also meets other requirements of the system like the availability of a

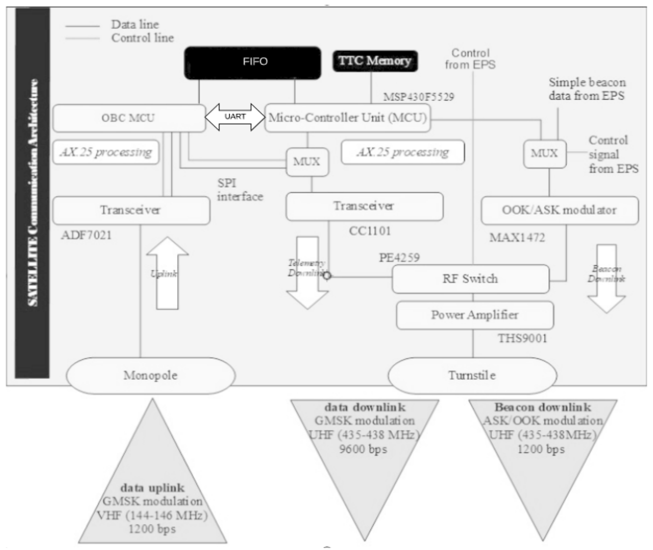


Figure 2. General architecture of the Telemetry Subsystem

UART interface. It receives the housekeeping and the payload data via FIFO from the on-board PS, processes it into AX25 packets and then stores them in its dedicated memory.

The Telemetry microcontroller is interfaced with the downlink transceiver, CC1101 and the OOK modulator, MAX1472. CC1101 transceiver was used because it works remarkably in terms of low power consumption and is also an economic option. The downlink transceiver is meant for transmission of payload data while the OOK modulator is responsible for sending beacon signal and housekeeping data. Both the transceivers are interfaced with the On Board turnstile antenna which operates in the UHF frequency range(435MHz - 438MHz). GMSK (Gaussian Minimum Shift Keying) modulation scheme has been chosen for the modulation of data to be transmitted as it provides improved spectral efficiency when compared to other phase shift keyed modes. An RF switch(THS9001) is used to toggle between the transmission of payload and housekeeping data.

Apart from two downlink transceivers, there is also an uplink transceiver, ADF7021, the control of which, has been given to the on-board PS. The uplink transceiver is interfaced with a monopole antenna that operates in the VHF frequency range (144MHz - 146MHz) and will be built in-house. On the ground station side, a Yagi-Uda antenna is interfaced with the Kenwood TS-2000 transceiver to receive downlink data and send TLE and re-transmission requests to the satellite. Yagi-Uda antenna was chosen because it provides the most cost effective option for getting gain and directivity and also has a relatively straightforward construction.

Summarizing the functions of the Telemetry subsystem

1. Transmission of tracking signal (beacon).
2. Packaging of downlink data.
3. Transmission of payload and housekeeping data.
4. Establishing a ground station to receive and process data.
5. Reception of commands from the ground station to be processed on board.
6. Collect feedback of antenna configuration and switch modes of operation.

4. INTERFACING

Communication

Different serial protocols like SPI, I2C (Inter-IC), and UART were considered while choosing the protocol for interfacing the two subsystems. And UART was chosen over other protocols because of the simplicity it provides. SPI and I2C use master-slave architecture, with a single master. This would require using MSP430 in slave mode, which is an arduous task. Even though I2C and SPI provide faster data rates as compared to UART, they were not a not judicious choice in this case as the amount of data to be transferred was very less and also infrequent.

UART is one of the most common serial interfaces used in embedded systems. It offers full duplex bidirectional way of communication, and uses two data lines, one for transmitting (Tx pin) and one for receiving (Rx pin) data. Since UART is asynchronous communication, the microcontrollers which want to communicate via UART have to agree on the transmission speed i.e. the baud rate which was chosen to be 9600. The number of data bits was chosen to be 8 along with no parity bits. Moreover, an interrupt-driven handler was chosen over a polling-driven handler as the amount of data to be transferred was asynchronous and infrequent.

MSP430 UART—The MSP430 provides a module called the USCI (Universal Serial Communications Interface) which supports multiple types of serial interfaces. There are two variants of the USCI module, each of which supports specific interfaces:

1. USCI_A : UART and SPI
2. USCI_B : SPI and I2C

In asynchronous mode, the USCI_A module connects the device to an external system via two external pins (Refer figure 3 for USCI_A UART block diagram). Since MSP430 runs on RTOS, the programming here is done on the register level.

Zynq-7000 SoC UART—The Zynq-7000 SoC provides two UART controllers (UART 0, UART 1), each with separate Rx and Tx data paths. Each path includes a 64-byte FIFO. The controller serializes and de-serializes data in the Tx and Rx FIFOs respectively and includes a mode switch to support various configurations like Local Loopback, Normal, etc. for

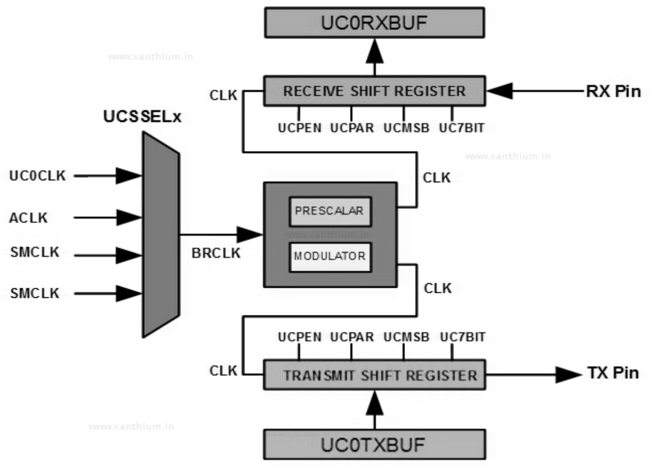


Figure 3. MSP430 USCI_A UART block diagram

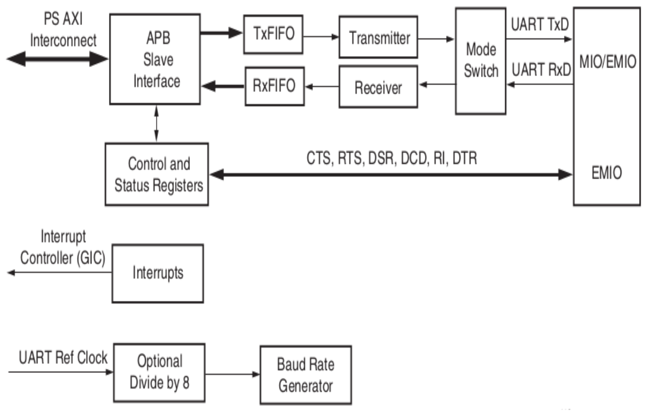


Figure 4. Zynq-7000 SoC UART block diagram

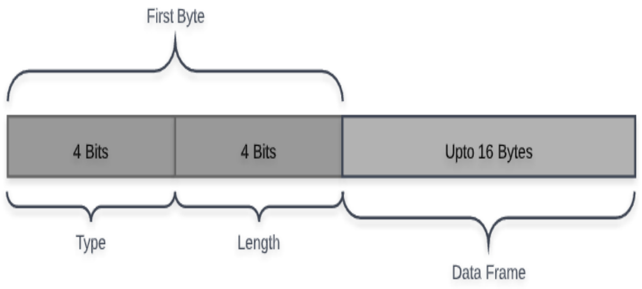


Figure 5. Protocol Definition

the RxD and TxD signals (refer figure 4). Since this runs on Petalinux, the programming here is done in C.

Protocol Definition

Even though the amount of data being transferred between the On-Board and Telemetry microcontrollers is not memory extensive, yet there is a wide range of information that needs to be conveyed. Thus, it is required to design a protocol that effectively distinguishes between various types of data, besides consuming memory judiciously. The protocol designed for UART communication consists of three components as mentioned (refer figure 5) -

i. Type - This is a stream of 4 bits that uniquely identifies the 'type' of information in the data frame.
ii. Length - This specifies the length of the data frame in bytes. 4 bits of memory has been allocated to it.
iii. Data Frame - Trailing the first byte of information is the actual data to be transmitted.

The various types of data being transferred and their type identifier have been listed below -

1. Payload Data - 0x0.
2. Housekeeping Data - 0x1.
3. Automatic Repeat Request - 0x2.
4. Antenna Configuration Info - 0x3.
5. Pass Info
   - [a.] Start of Pass - 0x4.
   - [b.] End of Pass - 0x5.
6. Acknowledgement of empty FIFO memory - 0x6.
7. Malfunction of a component - 0x7.

Payload and Housekeeping Data—The OBC memory contains the compressed payload data as well as the housekeeping data which is further required to be processed by the Telemetry into AX 25 packet frames. Hence, the OBC needs to send this data to the telemetry microcontroller. A FIFO is used for this purpose the details of which will be discussed later. As a result, the data frame essentially contains only the length of data that has been pushed into the FIFO.

Automatic Repeat Request—The OBC microcontroller is interfaced to the uplink transceiver. Consequently, it receives feedback from the ground station in case a part of downlinked data is erroneous or is lost during transmission. This information needs to be passed on to the telemetry microcontroller which then retransmits these packets. As the On-board PS has a core dedicated to processing the uplink data, it receives the Automatic Repeat Request from the ground station, unpacks the data, converts it into bitmasks indexed according to the packet numbers of the downlinked data and sends this information through UART to the telemetry microcontroller.

Antenna Configuration Info—As there exists a possibility of failure during antenna deployment, the Telemetry subsystem needs to switch between modes of operation accordingly. This information regarding the antenna configuration is transferred through UART by the On-board subsystem and it needs to be updated after regular intervals as there lies a probability that the mode of operation might change during flight. The data frame in this case only contains the bit sequence of the three possible modes of operation which depend upon the antenna configuration.

Pass Info—Apart from the antenna configuration, the Telemetry subsystem also needs to be notified about the start and end of a pass. A pass is a period over which the satellite is visible to the ground station. As the type frame itself defines whether a pass is about to begin or end, there is no need to include the data frame in this case. This is the only occasion when the data frame is absent.

Acknowledgement of empty FIFO memory—The amount of data to be downlinked is substantial and thus, would require several iterations of filling the memory by the OBC and emptying by the TTC. As a result, OBC needs to be notified each time the FIFO is emptied by the TTC microcontroller. This is the only instance when TTC communicates to OBC through UART while in all other cases the communication is established the other way.

Malfunction of a component—In case there is a failure in a particular component of the Telemetry subsystem, the Electric Power Subsystem (EPS) gets the information through its Over Current Protection Circuit (OCPC). EPS relays this data onto OBC through an SPI interface. In order to circumvent the failure and make sure that the telemetry functions properly, the OBC sends this data to the telemetry subsystem. Every component is marked with a label and the data frame essentially contains the label of the malfunctioning component.

Memory

The memory requirements for the dataflow between the On-Board Computer and Telemetry are significant. Telemetry houses a single microprocessor but the telemetry requirements call for a full-duplex system which needs additional processing power. Since the OBC SoC contains two cores, one of these cores will be dedicated to the uplink circuity during the uplinking process. It will receive the

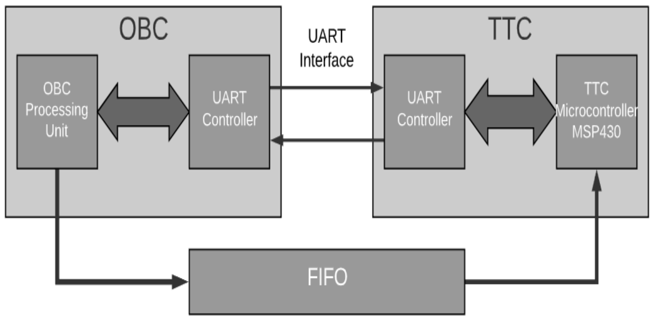


Figure 6. Flow of Data

ACKs(acknowledged) and NACKs(not-acknowledged) of the data packets sent from the satellite to the ground station.

For the payload and the advanced beacon data to be downlinked, it has to be first transferred to the telemetry microcontroller, where it is packeted and stored on the flash memory. A significant problem arises here. The operating system running on the OBC is Petalinux, an open-source operating system with a Linux Kernel, while the Telemetry microcontroller runs RTOS (Real-Time Operating System). Due to virtualization, the physical addresses are abstracted and are not known. But for data access, the Telemetry microprocessor requires the physical addresses of the payload data.

So to overcome this, instead of a traditional flash memory interfaced between these two subsystems, a FIFO memory is being used. This hardware can function as a memory that eases the task of data transfer and retrieval between the any two microcontrollers, provided one microcontroller always receives the data sent from the other.

This workaround reduces the complexity of the system. The OBC pushes the data into the FIFO memory, which is then popped by telemetry, packeted and stored into second flash memory. Communication regarding the type of data stored inside the FIFO is done using UART. As already mentioned, housekeeping data and payload data are transferred for downlink using this approach.

5. Discussion

The paper goes over a possible solution for providing an architecture of the interface between two subsystems of a nanosatellite (On-board computer and telemetry), one of which runs on RTOS and the other on a Linux based OS. We chose UART over other serial protocols because of the ease of implementation it provides with MSP430 (telemetry microcontroller). Hence, this type of interfacing was found to be efficient for communication between two microcontrollers that need to transfer data which is not memory extensive.

The paper also exemplifies an approach of using FIFO memory to simplify the architecture of the interface between any two microcontrollers or SoCs for serial data transfer. This is also an asynchronous mode of data transfer as the two subsystems can send and receive data at different rates.

The protocol defined above, though fairly efficient can be improved by reducing the number of type bits. Here some bits were left to accommodate any future additions in the type of data being sent.

Acknowledgments

The authors would like to thank the following:

- Dr. Meetha V. Shenoy, Parth Kalgaonkar, Shubham Sharma, Mohit Vyas, Umang Garg and Sakshi Agarwal for their constant support and technical expertise in perusing research for this paper.

- The administration of BITS Pilani and Indian Space Research Organization (ISRO) for giving us an opportunity to work on building a satellite.

References

[1] Shubham Sharma, Kushagra Aggarwal, Dhananjay Mantri, Tanuj Kumar, Saurabh M. Raje, Abhishek Goel. Development of On-Board Computer for a nanosatellite. In 68th International Astronautical Congress(IAC), 2017.

[2] Neelanchal Joshi and Parth Kalgaonkar, Implementation of CCSDS Hyperspectral Image Compression Algorithmon on FPGA on board a nanosatellite. In 8th European Conference for Aeronautics and Space Sciences (EUCASS), 2019.

[3] Zynq-7000 SoC Technical Reference Manual

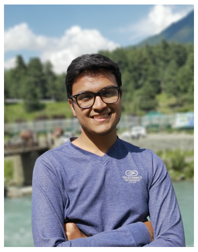
[4] Xilinx. Zedboard Hardware User's Guide 2014

[5] MSP430x5xx and MSP430x6xx Family User's Guide.

[6] Implementing a UART Function With the 8-Bit Interval - Texas Instruments Application Report

Biography

Abhishek Prasad *a student of Birla Institute of Technology and Science, Pilani, India is pursuing B.E.(Hons.) in Computer Science and Engineering. He has been a part of Team Anant since March 2019, contributing to the development of Telemetry, Telecommand and Ground Station Subsystem. He has been managing and working on various projects related to TTC and system integration. His interests lie in telecommunication and he wishes to pursue research in the same.*



Yash Jain *a student of Birla Institute of Technology and Science, Pilani, India is pursuing B.E.(Hons.) in Electrical and Electronics Engineering. He has been a part of Team Anant since May 2019, contributing to the development of On-Board Computer Subsystem (OBC). He has been managing and working on various projects related to OBC and system integration. His interests lie in the field of electronics and computer sciences.*

Neelanchal Joshi *a student of Birla Institute of Technology and Science, Pilani, India is pursuing Dual Degree in M.Sc. Physics and B.E.(Hons.) in Electronics and Electrical Engineering. He has been a part of Team Anant since May 2018, contributing to the development of On-Board Computer Subsystem (OBC). He has been sub-system lead of OBC since March 2019, managing and working on various projects related to OBC and system integration. His interests lie in Astrophysics and he wishes to pursue research in the same.*

Nishant Gupta *a student of Birla Institute of Technology and Science, Pilani, India is pursuing B.E.(Hons.) in Electronics and Instrumentation Engineering. He has been a part of Team Anant since January 2018, contributing to the development of Telemetry, Telecommand and Ground Station Subsystem. He has been lead of the subsystem since April 2019, managing and working on various projects related to Telemetry and subsystem integration. His interests lie in RF Microelectronics, Device Physics and Control Systems, and he wishes to pursue research in the same.*

Varsha Singhania *a student of Birla Institute of Technology and Science, Pilani, India is pursuing B.E.(Hons.) in Electronics and Instrumentation Engineering. She has been a part of Team Anant since May 2018, contributing to the development of Telemetry, Telecommand and Ground Station Subsystem. She has been a system engineer since March 2019, managing and working on various projects related to TTC and system integration. Her interests lie in digital and communication system and she wishes to pursue research in the same.*

Yatharth Sreedharan *a student of Birla Institute of Technology and Science, Pilani, India is pursuing Dual Degree in M.Sc. Chemistry and B.E.(Hons.) in Electronics and Electrical Engineering. He has been a part of Team Anant since May 2018, contributing to the development of On-Board Computer Subsystem (OBC).*