Proposed OBC Prototype for CubeSat

Mohamed Elazab Abdalrahman, Ahmed Ehab Fikry, and Mahmoud Nasr Eldeen
Department of Computer Engineering and Artificial Intelligence, Military Technical College, Cairo, Egypt,
mohamed.elazab080@gmail.com, ahmedfikry059@gmail.com, sirghandy@gmail.com

Supervisors: Dr. Sherif Hussein and Dr. Mohamed A. Elshafey
Department of Computer Engineering and Artificial Intelligence, Military Technical College, Cairo, Egypt,
s.hussein@mtc.edu.eg and m.shafey@mtc.edu.eg

Abstract– The Cube Satellite (CubeSat) is an ideal solution to gain access to space in terms of budget and integration time for research and experimental science payloads. The On-Board Computer (OBC) is considered to be the main subsystem of the CubeSat among others. It manages all the tasks taking place within the satellite. Its main function is to interpret the orders from Earth, treat and return the results. This paper surveys a previously developed CubeSat OBCs with focus on the characteristics of the embedded system used in the design and implementation phases. Then, it presents the design, development and implementation process of a prototype OBC hardware and software for CubeSat. The proposed software architecture is implemented as a prototype on one of STMicroelectronics boards. The choice of this development board was highly motivated by the mission of the CubeSat. Finally, a series of tests are successfully conducted to evaluate the proposed architecture.

Keywords-- CubeSat, OBC, OBC Hardware, OBC Software.

I. INTRODUCTION

In recent years, the advancement of space technology could not pass unnoticed, thus, a lot academic institutions across the world focus on space science. Specifically, more attention were given to the CubeSat platform that was proposed by the California Polytechnic State University (CalPoly) in 1999 [1]. CubeSats comprise of several subsystems, each is performing a dedicated task. According to Gildeh's basic design, the CubeSat normally includes the following subsystems [2]:

- An On-Board Computer (OBC), which is the most important subsystem in the CubeSat.
- An Attitude Determination and Control Subsystem (ADCS).
- A Radio Frequency (RF) Communication Subsystem (CSS).
- Electrical Power Subsystem (EPS) supplied by solar panels.
- A memory module generally associated with the OBC.
- A payload (e.g. beacon transmitter or camera) with its control unit.

Many commercial companies provide modules for OBC but these modules cost too much, so we are intending to do the same purpose but with a less cost and make this development available for all engineers to participate in a more development in this field.

OBC is the brain of the CubeSat. It consists of a controller that is connected to other subsystems through a serial data bus. The surface area of the solar panels of the CubeSat is proportional to the CubeSats' size itself. This limitation requires that the controller primarily consumes low average power and at the same time handles all data transfers according to the mission requirements. The major functions of the OBC are as follows [3]:

- Store telemetry and satellite data for transmitted to the ground station for analysis.
- Encodes and decodes packets sent/received to and from the ground station.
- Process of the telecommands received from the ground station.
- Monitoring all of other subsystems and resetting certain critical subsystems if necessary.

Currently Cubesat projects are in international collaborations of more than 100 universities, colleges, agencies and private firms that gather all types of science. This in return requires system engineers, in our Satellite localization centre in Military technical college (MTC) in Egypt, we are concerning with the aspect of participation and collaboration of all engineers to build different types of integrated systems and the Cubesat was chosen as an application for our criteria.

This paper proposes a prototype OBC subsystem for CubeSat applications, based on ARM Cortex M4 processor using STM32F446RE NUCLEO board. Our design was based on two main milestones. The first one is the set of requirements and specifications for the OBC in order to fulfil all requirements from OBC subsystem in the CubeSat. The second is to implement and test the OBC architecture with all its drivers according to the specifications defined in the first phase. We concerned in our design with the development of all drivers in a great fashion that makes APIs are very simple, by which the user requirements can be accomplished very quickly, with less memory and more power saving as well. The rest of this paper is organized as follows. In Section 2, we give a brief on the history of OBC design and development. Then, in Sections 3, we introduce the proposed frame work of the OBC prototype. In Section 4, we evaluate and test our design.

Finally, section 5 concludes our work and gives recommendations for the future work.

II. Related work

CubeSats' architecture is similar to what is shown in Figure 1. The OBC is located at the centre of all other subsystems, where the communication between systems is conducted via a serial bus interface. The OBC includes enough memory that supports the functions of other subsystems, as it is used to record housekeeping parameters and telemetry data collected at given coordinates, before starting the transmission to the ground station [4].

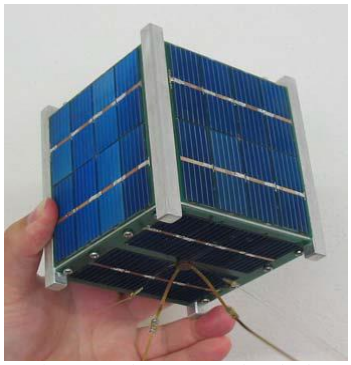

Fig. 1 A CubeSat general outlook.

Each subsystem is assigned to a dedicated task. A brief description of the function of the main subsystem as follows:

- Electrical Power Subsystem (EPS) supplies a regular source of electrical power to all satellite subsystems during the mission.
- Communication Subsystem provides the communication link between the satellite and the ground station.
- ADCS provides a stabilization mechanism in order to keep the satellite's orientation steady and ensure it operates efficiently [5].
- Payload subsystem, based on the satellite mission, can be classified as communications, imaging or scientific missions.
- OBC, which is our concentration in this paper, acts as heart of this data transfer and as the communications link between all other subsystems in the CubeSat.

To date, a number of successful CubeSat missions orbiting Earth were very successful and there are still a large number of them currently under development by different universities, research institutions and organisations all over the world.

One of the most famous trails for OBC development was César A. Perdomo, Julián R. Camargo& Albeiro Cortes Cabezas [6] in their trial for implementing OBC-OBDH (Onboard-Computer On-Board Data Handling) to meet requirements of Cubesat QB50 project and the implementation of FreeRTOS. In this work they developed a Printed Circuit Board (PCB) that includes two SD-modules as redundant systems and make its circuit embedded with their PCB but all other sensors are represented as modules. Moreover, Innovative Solution in Space (ISIS) was developed in 2006. Its scientific idea was mainly based on the Delfi-C3 Dutch CubeSat mission (Delft University of Technology). ISIS was successfully put into its orbit in 2008 and is still operational.

III. The proposed OBC prototype

A. *Hardware perspective*

In this section we will discuss the proposed system for the OBC and the specification of the Microcontroller Unit (MCU) used for it. This section describes the multiprocessing and real time availability of the selected MCU [7]. The final connections of the integrated framework are also showed.

1) The proposed OBC module

A Cubesat system consists of certain subsystems; each subsystem has a certain specification function. OBC is the main brain of the system that controls all subsystems hanging on the Cubesat. In the proposed prototype, the OBC has the following major functionalities:

- Recording and storage of telemetry and satellite payload data that it transmits to the Ground Control Station (GCS) to be analyzed.
- It acts as an encoder and decoder of data packets from and to the GCS.
- It also process telecommands from the GCS including the time delay that is received on the uplink channel.
- Monitors several subsystems and implementing watchdog function.
- It may be used for power supply management (checking battery level) and shutting down subsystems (power saving mode).

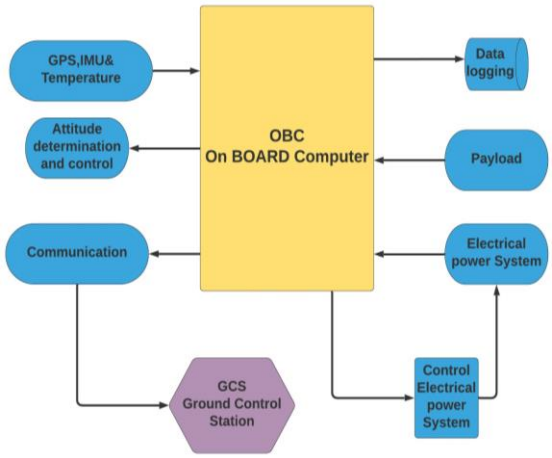

Fig. 2 Functional architecture of the OBC module.

The first subsystem, which we will discuss is the EPS. It is responsible for feeding the power for the whole system. This

power comes from solar panels and it is controlled to be sun synchronous, OBC can monitor the usage of the power through ADC peripheral of the MCU. The second subsystem is the ADCS. In which, we setup sensors to be ready for reading measures and make calibration for all sensors on the system then we do sensor fusion in order to make attitude determination and control, the output of attitude determination and control is fed to either magnetorquer or reaction-wheel. The third subsystem is the Payload. Our payload is a VGA camera. This camera is assigned to operate between two sets of certain latitudes and longitudes to take either number of shots or continuous recording of images and log these images on the SD-card. The fourth subsystem is the CSS. It is responsible for sending telemetry data and payload data, stored before in the SD-card, when it is in the line of sight of GCS. It is also responsible for receiving telecommands from GCS, by which the desired location to be monitored will be determined. It also handles all errors occurs in the system. The following figure shows the hardware connections of the OBC subsystem with other subsystems in a proposed framework.

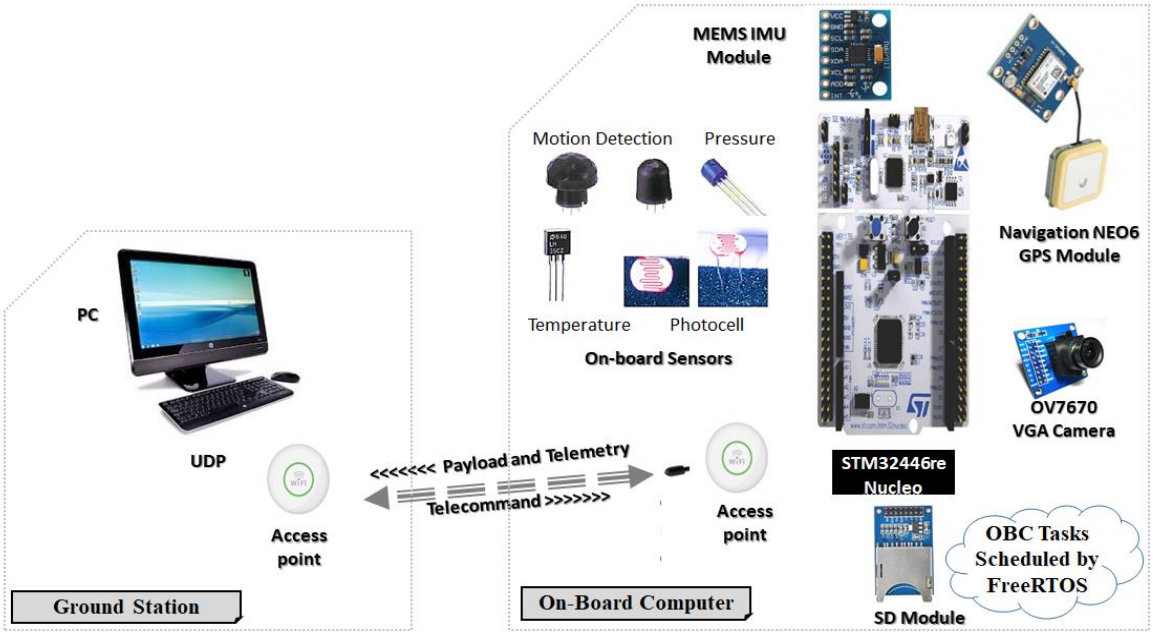

Fig. 3 The hardware connections in the proposed framework containing the OBC module.

2) Hardware specification of OBC subsystem

It is necessary that our selected MCU should meet the basic functional requirements of the OBC subsystem (i.e. it should has all the peripherals needed to accomplish our mission). Since we have many sensors that operates on SPI, I2C, USART, GPIO, etc., our board must have these control units. In the proposed prototype, we select STM32F446RE Nucleo ARM Cortex M4 based MCU. In the upcoming section we will preview our board specifications and its merits that leads us to this choice. The following data, shown in Table 1, are taken from the manufacturer datasheet [8, 9].

TABLE I
GENERAL FEATURES FOR STM32F446XX

Processor core	ARM® 32-bit Cortex®-M4 CPU 180 MHz
Memory	512 kB of Flash memory
Clock, reset and supply management	1.7 V to 3.6 V application supply and I/O_s – POR, PDR, PVD and BOR – 4-to-26 MHz crystal oscillator – Internal 16 MHz
Low power	Sleep, Stop and Standby modes– VBAT supply for RTC, 20×32 bit backup registers + 4 KB backup SRAM.
General-purpose DMA	16-stream DMA controller with FIFOs and burst support.
Timers	Up to 17 timers: 2x watchdog, 1x SysTick Timer.
Debug mode	SWD & JTAG interfaces.
I/O Busses	Up to 114 I/O ports with interrupt capability – Up to 111 fast I/Os up to 90 MHz – Up to 112 5 V-tolerant I/Os.
Interfaces	Up to 20 communication interfaces – SPDIF-Rx – 4 × I2C interfaces (SMBus/PMBus) – 4 USARTs/2 UARTs (11.25 Mbit/s, ISO7816 interface, LIN, IrDA, modem control) – 4 SPIs (45 Mbits/s), 3 with muxed I2S for audio class accuracy via internal audio PLL or external clock – 2 x SAI (serial audio interface) – 2 × CAN (2.0B Active) – SDIO interface.
RTC	sub second accuracy, hardware calendar

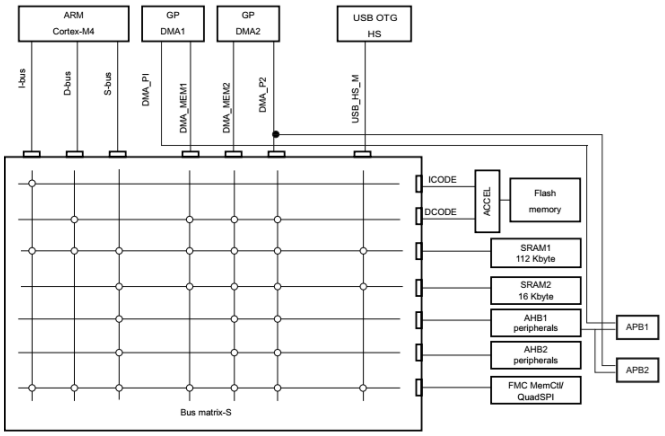

Fig. 4 Bus system Architecture for STM32f446xx [8, 9].

3) Real System in Workshop

We applied the concept of Flat sat design as a proof of concept for the proper operation of the system in the work shop, usually this concept comes before any manufacturing of boards or PCBs. Tests are made in the lab by many ways such as calibrating sensors and checking all the peripherals connected to OBC against other calibrated sensors. So the following figure shows the real connections after finishing the system in the work shop.

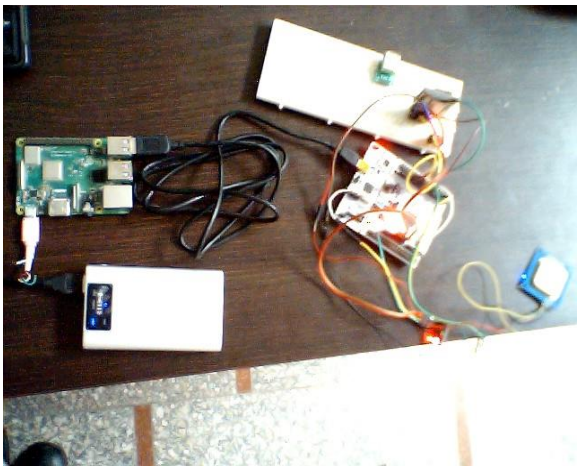

Fig. 5 System Wiring in Workshop (Flat sat).

B. Software and drivers' development

In this section we will show the development of all necessary drivers required for the proper operation of all sensor and modules connected to the OBC. The first one will be the GPIO which is most important one as it is used with all drivers, then we will present I2C, SPI, and UART/USART drivers, consequently. We presented a flowchart for each driver that describes most of its developed APIs. Additionally, we gave a brief concept or vision about each driver and why it was necessary for our work.

1) GPIO Driver

This driver is developed to allow user to read and write any digital and/or analog value. It is used as well to configure pins for other protocols of communications (I.e. I2C, USART and SPI). Our board contains 7 types of GPIOs marked from A to H each of 15 pins. We developed many API_s to allow user from manipulating the GPIO peripheral efficiently. Figure 5 indicates some of developed API_s in GPIO driver.

2) SPI Driver

We need to setup SD-Module so we developed this driver to manipulate this module. In the following we will give a brief explanation about SPI operation that we used to create its APIs. The master initiates the communication, determine the clock speed and select the devices to talk to while the other slave devices wait for the master device to communicate with them. The master initiates communication by dropping slave pins to 0V.The master device controls the clock signal by creating a pulse from logic 0 (0 volts) to logic 1 (3.3/5 volts) to control the speed and time the data passing through the MISO and MOSI lines. Figure 6 indicates some of APIs in SPI driver.

3) I2C Driver

We need to setup MPU6050 so we developed this driver to manipulate this sensor. I2C combines the best features of SPI and UARTs. With I2C, you can connect multiple slaves to a single master (like SPI) and you can have multiple masters controlling single, or multiple slaves. This is really useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD. Like UART communication, I2C only uses two wires to transmit data between devices. Figure 7 indicates some of developed APIs in I2C driver.

4) USART/UART Driver

We need to setup GPS, Serial communication with USB with Raspberry pi and RF Module so we developed this driver to deal with these peripherals. UART stands for Universal Asynchronous Receiver-Transmitter. It is a hardware peripheral that is present in microcontroller. Its function is to convert the incoming and outgoing data into serial binary stream. By using serial to parallel conversion, an 8-bit serial data received from the peripheral device is converted into parallel form and parallel data received from the CPU is converted using serial to parallel conversion. This data is present in modulating form and transmits at a defined baud rate. It is used when the high-speed data transfer is not required. It is a cheap communication device with a single transmitter/receiver. It requires a single wire for transmitting the data and another for receiving. It can be interfaced with a PC (personal computer) using an RS232-TTL converter or

USB-TTL converter. The common thing between RS232 and UART is they both don't require a clock to transmit and receive data. The UART frame consists of 1 start bit, 1 or stop bits and a parity bit for serial data transfer. Figure 8 indicates some of developed API$_s$ in UART/USART driver.

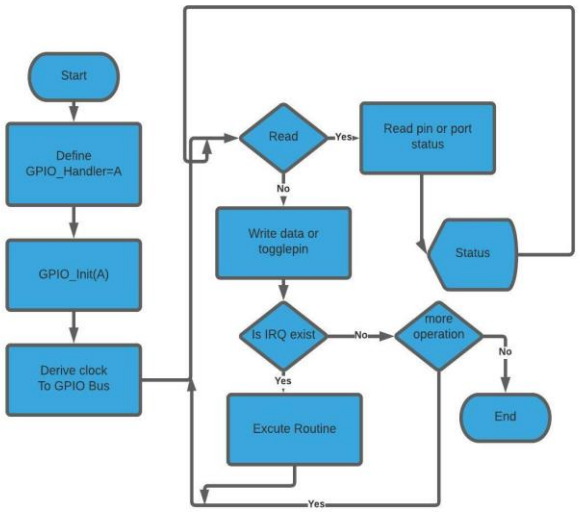

Fig. 6 Flowchart for GPIO APIs for STM32f446xx devices.

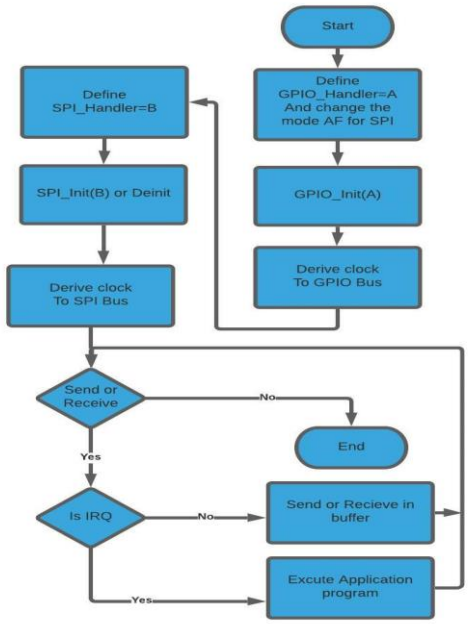

Fig. 7 Flowchart for SPI APIs for STM32f446xx devices

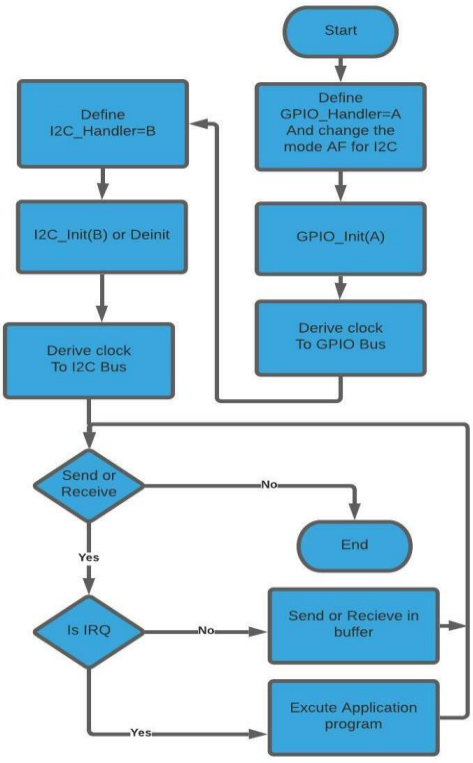

Fig.8 Flowchart for I2C APIs for STM32f446xx devices [4, 3].

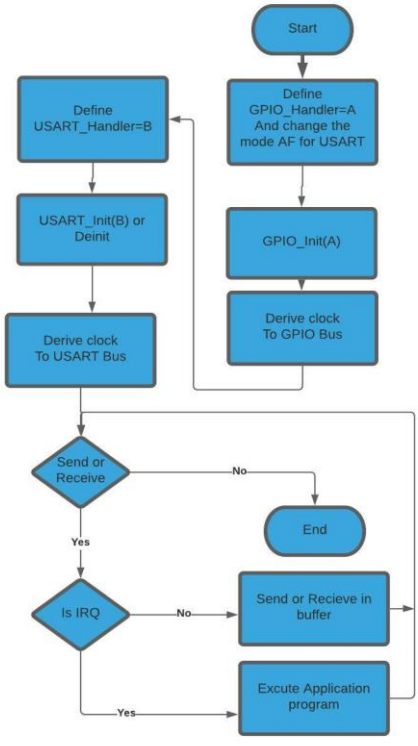

Fig.9 Flowchart for UART/USART APIs for STM32f446xx devices [3].

IV. TESTS AND RESULTS

After finishing the development of these drivers we can say that each driver is tested by many ways to make sure that drivers' works properly, we made that by two ways:

1) We made a code to test every peripheral on the IDE used for developing drivers. That shows us that our target has accomplished but that wasn't sufficient enough to proof that the driver works well so we need another way to increase our reliability of the proper operation of drivers.

2) This way we used the debugging mode and we could monitor all registers in memory and trace all changes with these registers and compared the result with the described operation discussed in reference manual.

In the following figure we will show an example of code based on GPIO that has been developed in debugging mode and we compared the results of values set in registers to that in reference manual to make sure that the board works properly and it is obvious from the result we had got that the LED is blinking as the users changes the time delay.

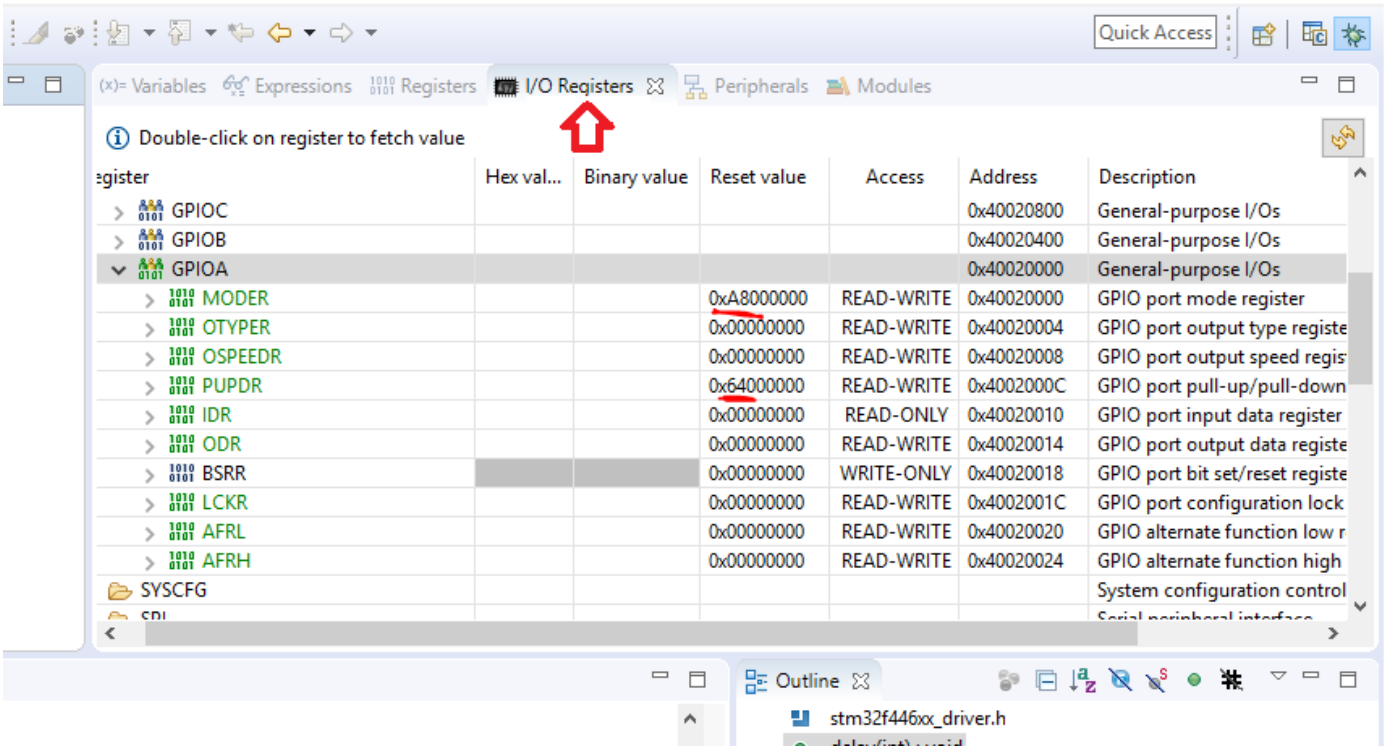

Fig. 10 Debugging mode from eclipse IDE (GPIO Registers shown)

V. Conclusion and Future Work

We have designed all necessary drivers for all required peripherals for OBC and we tested each driver with a separate application to proof the proper operation of the driver. The result of that work led to an integration of these peripherals in the form of sensor operation as an integrated code with a memory a less consumption than drivers developed by CubeMx S/W or any generic libraries for this boards, In my opinion that's because that in my driver I could initiate only the required register for example to be configured while other libraries as the don't exactly know what is the user requirements they initiate may be all the configuration of all registers. Hence, our developed drivers consume less energy than other libraries, since it doesn't consumes all MCU peripherals or PINs for any driver. In the future work we are seeking for implementing FreeRTOS to manage task operation and we will try to get rid of the computer (Raspberry pi) and use only STM board. That will lead us to develop a new driver that will be able to operate the camera on the MCU.

We are also trying to develop a GCS to be able to monitor the system based on LabVIEW Software; this GCS will be able to send telecommands in a visual interface form that will facilitate the operation for the operator on the GCS, More over the operator will see the performance of the CubeSat and will send telecommands for the region of interest to be captured by payload (camera), By sending to latitudes and longitudes, and the CubeSat will take this information and start taking snaps according to user desire.

References

[1] Poly, Cal. "Kahua Ranch Hosts June 23 Reception for Dean, Professor, Alumni and Students from Cal Poly College of Agriculture." (2002), Retrieved (2021).
[2] Gildeh, D. 2003. Final Report: Design and development of OBC for Pico-Sat PALMSAT. University of Surrey, Guildford.
[3] Wells, G. J., Stars, L.. & Jeans, T. 2003. Canada's Smallest Satellite; The Canadian Advanced Nanospace eXperiment (Can X-1). University of Toronto Institute for Aerospace Studies, pp. 4-7. Toronto, Canada.
[4] Hardy, J. 2009. Implementation du Protocol AX.25 a Board du Nano satellite OUFTI-1.Master's thesis, University of Liege, Liege.
[5] Hales, J.H. & Pedersen, M. 2002. Two-Axis MOEMS Sun Sensor for Pico Satellites. Proceedings of The 16th Annual AIAA/USU Conference on small satellites. 2002 Utah, USA (SSC02-VI-6).
[6] Perdomo, César A., Julián R. Camargo, and Albeiro Cortes Cabezas. "On Board Computer Module for Cubesat Compatible with QB50." *Mod. Appl. Sci.* 12.12 (2018): 206, Retrieved (2021).
[7] Jacko, Patrick, et al. "The parallel data processing by nucleon board with STM32 microcontrollers." 2017 International Conference on Modern Electrical and Energy Systems (MEES). IEEE, 2017.
[8] STMicroelectronics," ARM® Cortex®-M4 32b MCU+FPU, 225DMIPS, up to 512kB Flash/128+4KB RAM, USB OTG HS/FS, 17 TIMs, 3 ADCs, 20 comm. Interfaces," DocID027107 Rev 6, September 2016, retrieved online from: https://www.alldatasheet.com/datasheet-pdf/pdf/882029/STMICROELECTRONICS/STM32F446RE.html
[9] STMicroelectronics," STM32F446xx advanced ARM®-based 32-bit MCUs," DocID026976 Rev 3 , July 2017, retrieved online from:https://www.st.com/resource/en/reference_manual/dm00135183-stm32f446xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf.
[10] STMicroelectronics,"UM1724 User manual," DocID025833 Rev 12, September 2016, retrieved online from: https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf
[11] NXP Semiconductors, "I2C-bus specification and user manual," UM10204, Rev. 6. Retrieved online from: https://www.nxp.com/docs/en/user-guide/UM10204.pdf