

SpaceLab.OBDH2D.2020.06.001 rev C



OBDH 2.0 Documentation

OBDH 2.0 Documentation

SpaceLab, Universidade Federal de Santa Catarina, Florianópolis - Brazil

November 2022

OBDH 2.0 Documentation

November, 2022

Project Chief:

Eduardo Augusto Bezerra

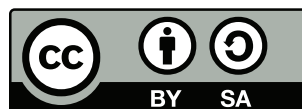
Authors:

Gabriel Mariano Marcelino
André Martins Pio de Mattos
Yan Castro Azeredo
Bruno Benedetti
Rebecca Quintino do Ó

Contributing Authors:

Revision Control:

Version	Author(s)	Changes	Date
0.1	Gabriel M. Marcelino	Document creation	10/2019
0.5	Gabriel M. Marcelino	First stable hardware	08/2020
0.6	A. M. P. Mattos, G. M. Marcelino, Y. A. Azeredo	First hardware integration	04/2021
0.7	A. M. P. Mattos, G. M. Marcelino, Y. A. Azeredo	General firmware and hardware improvements	06/2021
0.8	Gabriel M. Marcelino	Adding the FRAM memory	10/2021
0.9	Gabriel M. Marcelino	General improvements	03/2022
0.10	Gabriel M. Marcelino, Bruno Benedetti, Rebecca Q. do Ó	General improvements	11/2022



List of Figures

1.1	3D view of the OBDH 2.0 PCB.	1
2.1	Product tree of the OBDH 2.0 module.	4
2.2	OBDH 2.0 Block diagram.	5
2.3	System layers.	5
2.4	OBDH 2.0 operation flowchart	7
2.5	OBDH's 2.0 boot sequence.	8
2.6	Antenna deploy routine	9
2.7	Telecommand's processing flowchart	10
2.8	Transmit/Receive flowchart	11
2.9	Data path diagram.	12
2.10	Available status LEDs.	12
3.1	Top side of the PCB.	13
3.2	Bottom side of the PCB.	14
3.3	Side view of the PCB.	14
3.4	Interfaces diagram.	15
3.5	Bottom view of PC-104 and simplified labels	16
3.6	Antenna module connectors.	17
3.7	Programmer (P1 and P2) and jumper (P6) connectors.	18
3.8	Dedicated UART debug connectors (P7).	18
3.9	Samtec FSI-110-03-G-D-AD connector.	19
3.10	Daughterboard connector (P3).	19
3.11	Recommended shape and size of the daughterboard.	20
3.12	Illustrative daughterboard integration.	20
3.13	Microcontroller internal diagram.	21
3.14	Microcontroller pinout positions.	22
3.15	External watchdog timer circuit.	25
3.16	External memory circuit.	25
3.17	FRAM memory circuit.	26
3.18	I2C buffer circuit.	26
3.19	RS-485 transceiver circuit.	27
4.1	Product tree of the firmware of the OBDH 2.0 module.	30
5.1	Additional GPIOs and I2C channel 1.	44
5.2	Additional I2C channel 0.	44
5.3	Additional GPIO and SPI channel.	44
6.1	Firmware initialization on PuTTY.	46

List of Figures

- A.1 Top view of the OBDH 2.0 v0.5 board. 50
- A.2 Bottom view of the OBDH 2.0 v0.5 board. 51
- A.3 Log messages during the first boot. 52
- A.4 I²C port 0 test. 52
- A.5 I²C port 1 test. 53
- A.6 I²C port 2 test. 53
- A.7 54
- A.8 Current sensor fix. 54
- A.9 Log messages with the read values from the current sensor. 55
- A.10 NOR memory SPI test. 56

- B.1 Top view of the OBDH 2.0 v0.7 board. 58
- B.2 Bottom view of the OBDH 2.0 v0.7 board. 58
- B.3 Log messages during the first boot. 59
- B.4 Setup of the I2C port tests. 60
- B.5 Waveform of the I2C port 0. 61
- B.6 Waveform of the I2C port 1. 61
- B.7 Waveform of the I2C port 2. 62
- B.8 Test results of the NOR flash memory. 63
- B.9 Test results of the FRAM memory. 64

List of Tables

3.1	Boards interfaces.	15
3.2	PC-104 connector pinout.	16
3.3	Antenna module connectors pinout.	17
3.4	Programmer header connector pinout.	18
3.5	Programmer picoblade connector pinout.	18
3.6	Daughterboard connector pinout.	20
3.7	Microcontroller features summary.	21
3.8	USCI configuration.	21
3.9	Microcontroller pinout and assignments.	24
4.1	External libraries and dependencies of the firmware.	29
4.2	Firmware tasks.	31
4.3	Variables and parameters of the OBDH 2.0.	33
4.4	Beacon packet.	34
4.5	EDC information packet.	35
4.6	EDC samples packet.	36
4.7	System telecommand.	36
4.8	Enter hibernation telecommand.	36
4.9	Leave hibernation telecommand.	37
4.10	Leave hibernation telecommand.	37
4.11	Ping telecommand.	39
4.12	Ping telecommand answer.	39
4.13	Message broadcast telecommand.	40
4.14	FreeRTOS main configuration parameters.	41
5.1	Additional PC104 interfaces.	44

Nomenclature

EDC	<i>Environmental Data Collector.</i>
PCB	<i>Printed Circuit Board.</i>
PCB	<i>Printed Circuit Board.</i>
ADC	<i>Analog-to-Digital Converter.</i>
FRAM	<i>Ferroelectric Random-Access Memory.</i>
GPIO	<i>General Purpose Input/Output.</i>
HAL	<i>Hardware Abstraction Layer.</i>
I2C	<i>Inter-Integrated Circuit.</i>
IC	<i>Integrated Circuit.</i>
IDE	<i>Integrated Development Environment.</i>
JTAG	<i>Joint Test Action Group.</i>
LED	<i>Light-Emitting Diode.</i>
OBC	<i>On-Board Computer.</i>
OBDH	<i>On-Board Data Handling.</i>
RTOS	<i>Real-Time Operating System.</i>
SPI	<i>Serial Peripheral Interface.</i>
UART	<i>Universal Asynchronous Receiver/Transmitter.</i>
USCI	<i>Universal Serial Communication Interfaces.</i>

Contents

List of Figures	vi
List of Tables	vii
Nomenclature	ix
1 Introduction	1
2 System Overview	3
2.1 Product tree	3
2.2 Block Diagram	3
2.3 System Layers	4
2.4 Operation	5
2.4.1 Execution Flow	6
2.4.2 Data Flow	6
2.4.3 Status LEDs	6
3 Hardware	13
3.1 Interfaces	14
3.2 External Connectors	14
3.2.1 PC-104	15
3.2.2 Antenna Module	17
3.2.3 Programmer and Debug	17
3.2.4 Daughterboard	18
3.3 Microcontroller	19
3.3.1 Interfaces Configuration	21
3.3.2 Clocks Configuration	21
3.3.3 Pinout	22
3.4 External Watchdog	24
3.5 Non-Volatile Memories	25
3.5.1 Flash NOR	25
3.5.2 FRAM	25
3.6 I2C Buffers	26
3.7 RS-485 Transceiver	26
3.8 Voltage and Current Sensors	27
4 Firmware	29
4.1 Product tree	29
4.2 Dependencies	29

4.3	Tasks	29
4.3.1	Antenna deployment	29
4.3.2	Antenna reading	29
4.3.3	Beacon	29
4.3.4	Data log	31
4.3.5	EDC reading	31
4.3.6	EPS reading	31
4.3.7	Heartbeat	31
4.3.8	Housekeeping	31
4.3.9	Read sensors	31
4.3.10	Startup (boot)	32
4.3.11	System reset	32
4.3.12	Telecommand processing	32
4.3.13	Time control	32
4.3.14	TTC reading	32
4.3.15	Watchdog reset	32
4.4	Variables and Parameters	32
4.5	Telemetry	34
4.5.1	Beacon	34
4.5.2	EDC Information	34
4.5.3	EDC Samples	34
4.6	Telecommands	34
4.6.1	Enter hibernation	35
4.6.2	Leave hibernation	35
4.6.3	Activate beacon	37
4.6.4	Deactivate beacon	37
4.6.5	Activate EDC	37
4.6.6	Deactivate EDC	37
4.6.7	Get EDC info	37
4.6.8	Activate Module	37
4.6.9	Deactivate Module	38
4.6.10	Activate Radiation Instrument	38
4.6.11	Deactivate Radiation Instrument	38
4.6.12	Activate Payload	38
4.6.13	Deactivate Payload	38
4.6.14	Erase Memory	38
4.6.15	Force Reset	38
4.6.16	Get Payload Data	38
4.6.17	Set Parameter	39
4.6.18	Get Parameter	39
4.6.19	Set system time	39
4.6.20	Ping	39
4.6.21	Message broadcast	40
4.6.22	Request data	40
4.7	Operating System	40
4.8	Hardware Abstraction Layer (HAL)	40

5	Board Assembly	43
5.1	Development Model	43
5.1.1	Debug and programming connectors	43
5.1.2	Status leds	43
5.2	Flight Model	43
5.3	Custom Configuration	43
6	Usage Instructions	45
6.1	Powering the Board	45
6.2	Log Messages	45
6.3	Daughterboards Installation	45
	References	47
	Appendices	49
A	Test Report of v0.5 Version	49
A.1	Visual Inspection	49
A.2	Firmware Programming	49
A.3	Communication Busses	50
A.4	Sensors	51
A.4.1	Input Voltage	51
A.4.2	Input Current	52
A.5	Peripherals	53
A.5.1	NOR Flash Memory	53
A.6	Conclusion	55
B	Test Report of v0.7 Version	57
B.1	Visual Inspection	57
B.2	Firmware Programming	57
B.3	Communication Busses	59
B.4	Sensors	60
B.4.1	Input Voltage	60
B.4.2	Input Current	61
B.5	Peripherals	62
B.5.1	NOR Flash Memory	62
B.5.2	FRAM Memory	63
B.6	Conclusion	63

CHAPTER 1

Introduction

The OBDH 2.0 is an On-Board Computer (OBC) module designed for nanosatellites. It is one of the service modules developed for the GOLDS-UFSC CubeSat Mission [1]. The module is responsible for synchronizing actions and the data flow between other modules (i.e., power module, communication module, payloads) and the Earth segment. It packs the generated data into data frames and transmits it back to Earth through a communication module or stores it on non-volatile memory for later retrieval. Commands sent from the Earth segment to the CubeSat are received by radio transceivers located in the communication module and redirected to the OBDH 2.0, which takes the appropriate action or forward the commands to the target module.

The module is a direct upgrade from the OBDH of FloripaSat-1 [2], which grants a flight heritage rating. The improvements focus on providing a cleaner and more generic implementation compared to the previous version, along with more reliability in software and hardware implementations and adaptations for the new mission requirements. The whole project, including source and documentation files, is available freely on a GitHub repository [3] under the GPLv3 license.

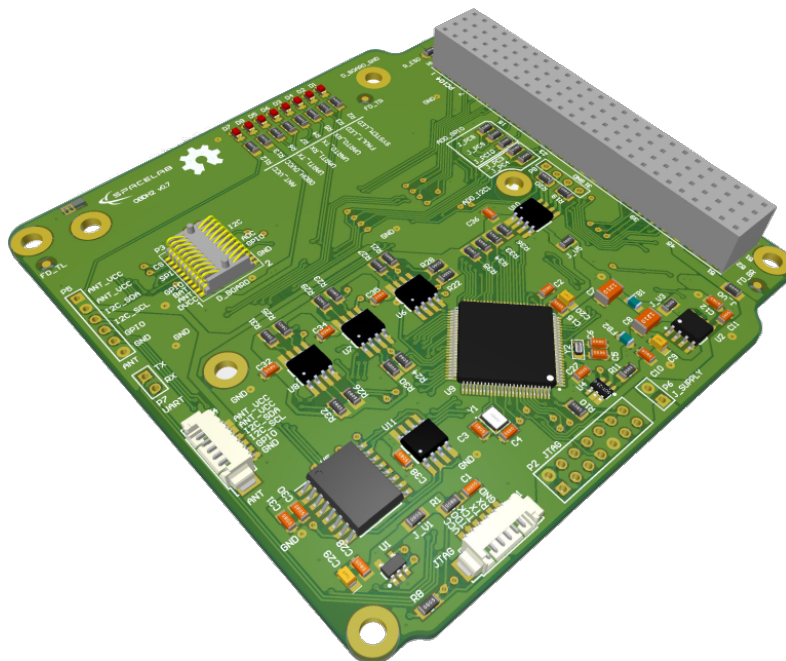


Figure 1.1: 3D view of the OBDH 2.0 PCB.

CHAPTER 2

System Overview

The board has an MSP430 low-power microcontroller that runs the firmware application and several other peripherals for extended operation and physical interfaces (i.e., non-volatile memory, watchdog timer, service modules, payloads interfaces, daughterboard interface, and current monitor). The microcontroller manages the other sub-modules within the board using serial communication buses, synchronizes actions, handles communication with the ground segment, and manages the data flow. The programming language used is C, and the firmware was developed using the Code Composer Studio IDE (a.k.a. CCS) for compiling, programming and testing. The module has many tasks over distinct protocols and time requirements, such as interfacing peripherals and other MCUs. To improve predictability, a Real-Time Operating System (RTOS) is used to ensure that the deadlines are observed, even under a faulty situation in a routine. The RTOS chosen is the FreeRTOS (v10.0.0), since it is designed for embedded systems applications and was already validated in space applications. The firmware architecture follows an abstraction layer scheme to facilitate higher-level implementations and allow more portability across different hardware platforms.

2.1 Product tree

The product tree of the OBDH 2.0 module is available in Figure 2.1.

2.2 Block Diagram

Figure 2.2 presents a simplified view of the module subsystems and interfaces. The microcontroller has a programming JTAG and 6 communication buses, divided into 3 different protocols (I2C, SPI, and UART), that is shared between all the peripherals and external interfaces. Besides these channels, there are GPIO connections for various functions, from control ports to status pins. There is a non-volatile memory device to store the satellite data frames and critical status indicators. Some buffers and transceivers allow secure and proper communication with external modules. A watchdog timer with a voltage monitor and a current sensor is attached to the system to improve the overall reliability and generate essential housekeeping data. There is a generic daughterboard interface for extending the module capabilities with an auxiliary application board. Also, a UART debug interface is directly connected to the microcontroller. More details and descriptions about these components and interfaces are provided in chapter 3.

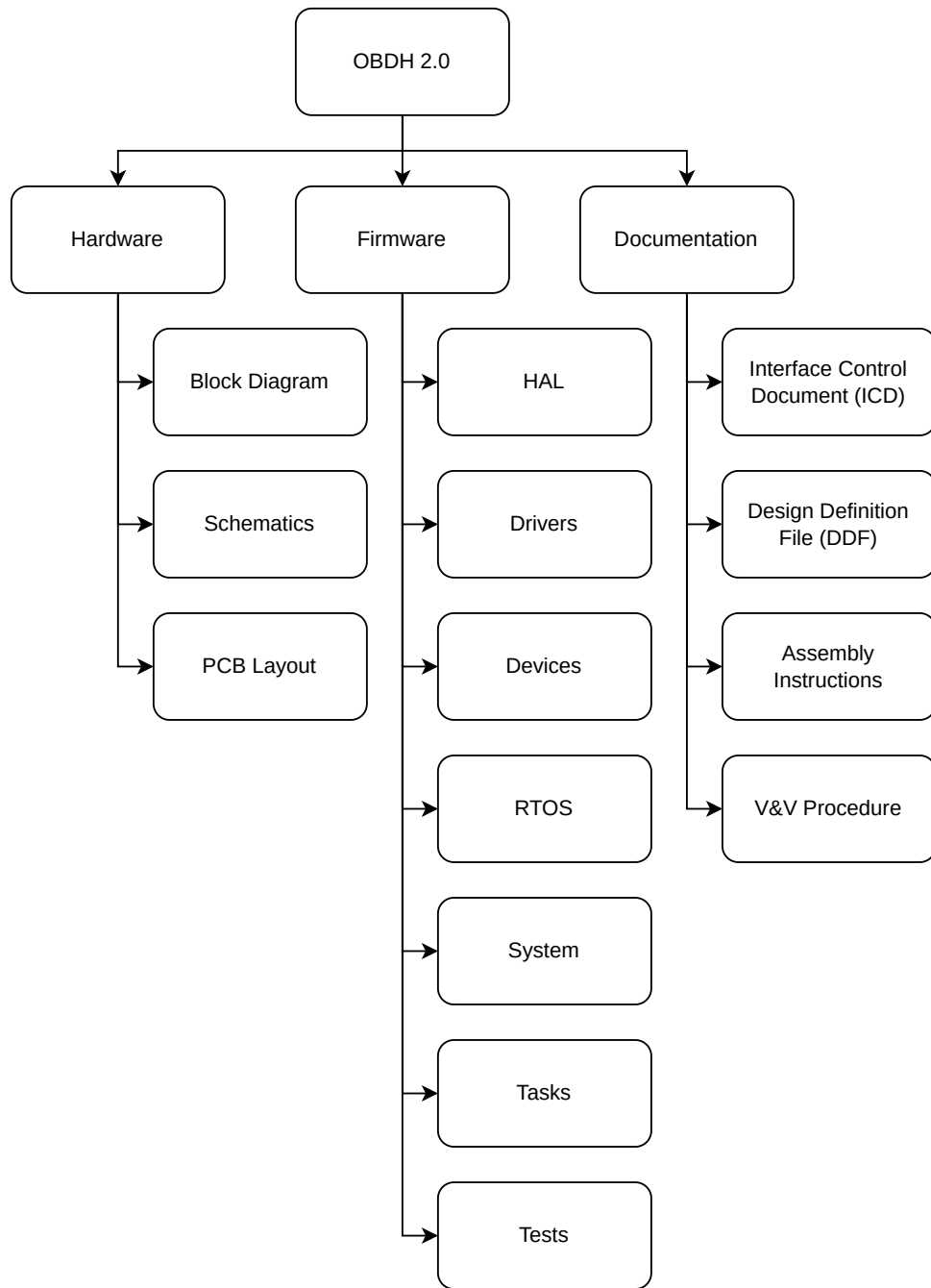


Figure 2.1: Product tree of the OBDH 2.0 module.

2.3 System Layers

As mentioned, the system is divided into abstraction layers to favor high-level firmware implementations. The Figure 2.3 shows this scheme, composed of third-party drivers at the lowest layer above the hardware, the operating system as the base building block of the module, the devices handling implementation, and the application tasks in the highest layer. More details are provided in chapter 4.

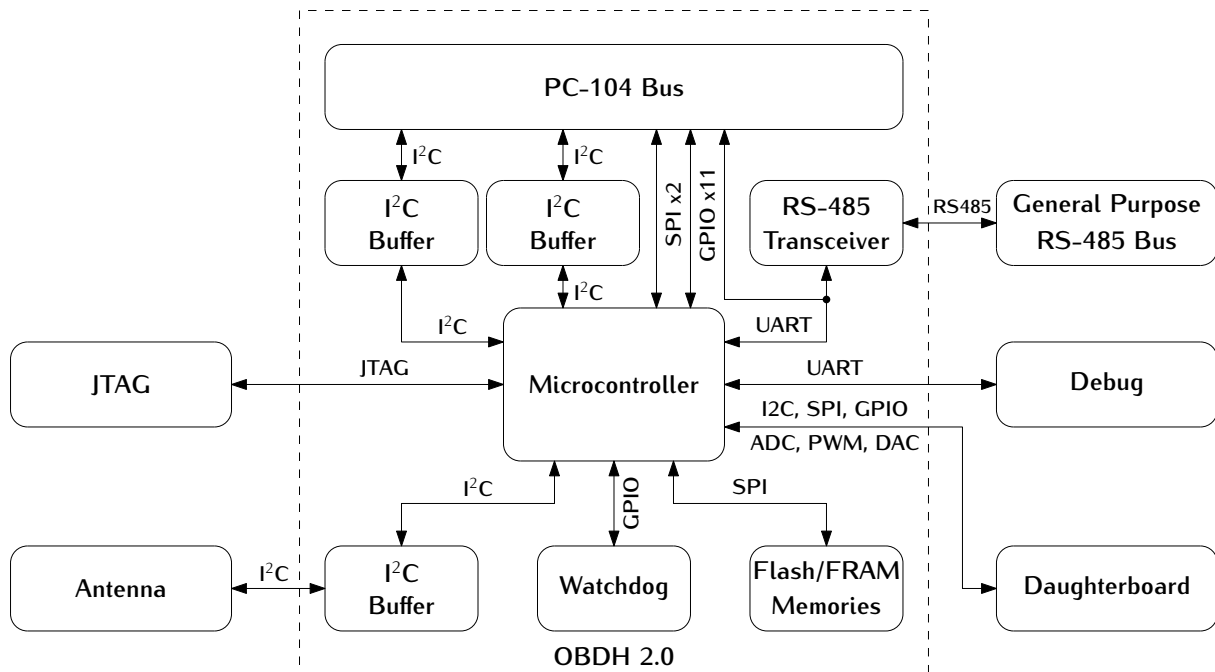


Figure 2.2: OBDH 2.0 Block diagram.

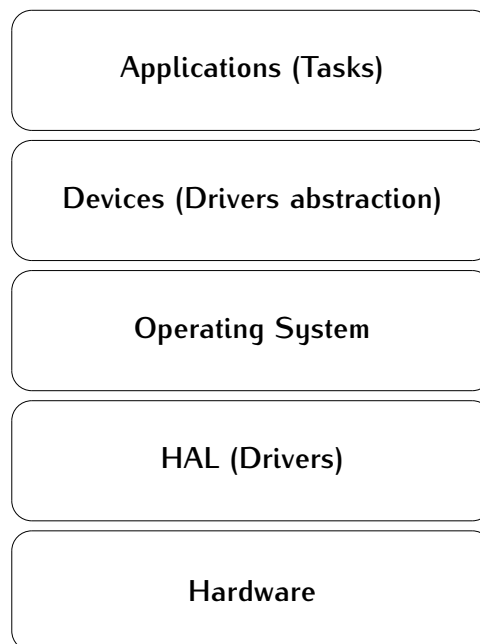


Figure 2.3: System layers.

2.4 Operation

The system operates through the sequential execution of routines (tasks in the context of the operating system) that are scheduled and multiplexed over time. Each routine has a priority and a periodicity, determining the subsequent execution, the set of functionalities currently running, and the memory usage management. Besides this deterministic scheduling system, the routines have communication channels with each other through

the usage of queues, which provides a robust synchronization scheme. In chapter 4 the system operation and the internal nuances are described in detail. Then, this section uses a top-view user perspective to describe the module operation.

2.4.1 Execution Flow

The OBDH 2.0 execution flowchart can be seen on Figure 2.4.

The boot sequence can be seen better on Figure 2.5.

The antenna deploy routine is exemplified with the flowchart on Figure 2.6.

The Telecommand's processing flowchart can be seen on Figure 2.7.

The Transmit/Receive sequence can be seen with the flowchart on Figure 2.8.

2.4.2 Data Flow

The OBDH 2.0 controls most of the CubeSat's data flow, which can be seen on Figure 2.9.

2.4.3 Status LEDs

On the development version of the board, eight LEDs indicate some behaviors of the systems. This set of LEDs can be seen on Figure 2.10.

A description of each of these LEDs are available below:

- **D1 - System LED:** Heartbeat of the system. Blinks at a frequency of 1 Hz when the system is running properly.
- **D2 - Fault LED:** Indicates a critical fault in the system.
- **D3 - UART0 TX:** Blinks when data is being transmitted over the UART0 port.
- **D4 - UART0 RX:** Blinks when data is being received over the UART0 port.
- **D5 - UART1 TX:** Blinks when data is being transmitted over that UART1 port.
- **D6 - UART1 RX:** Blinks when data is being received over the UART1 port.
- **D7 - Antenna VCC:** Indicates that the antenna module board is being power sourced.
- **D8 - OBDH VCC:** Indicates that the OBDH board is being power sourced.

These LEDs are not mounted in the flight version of the module.

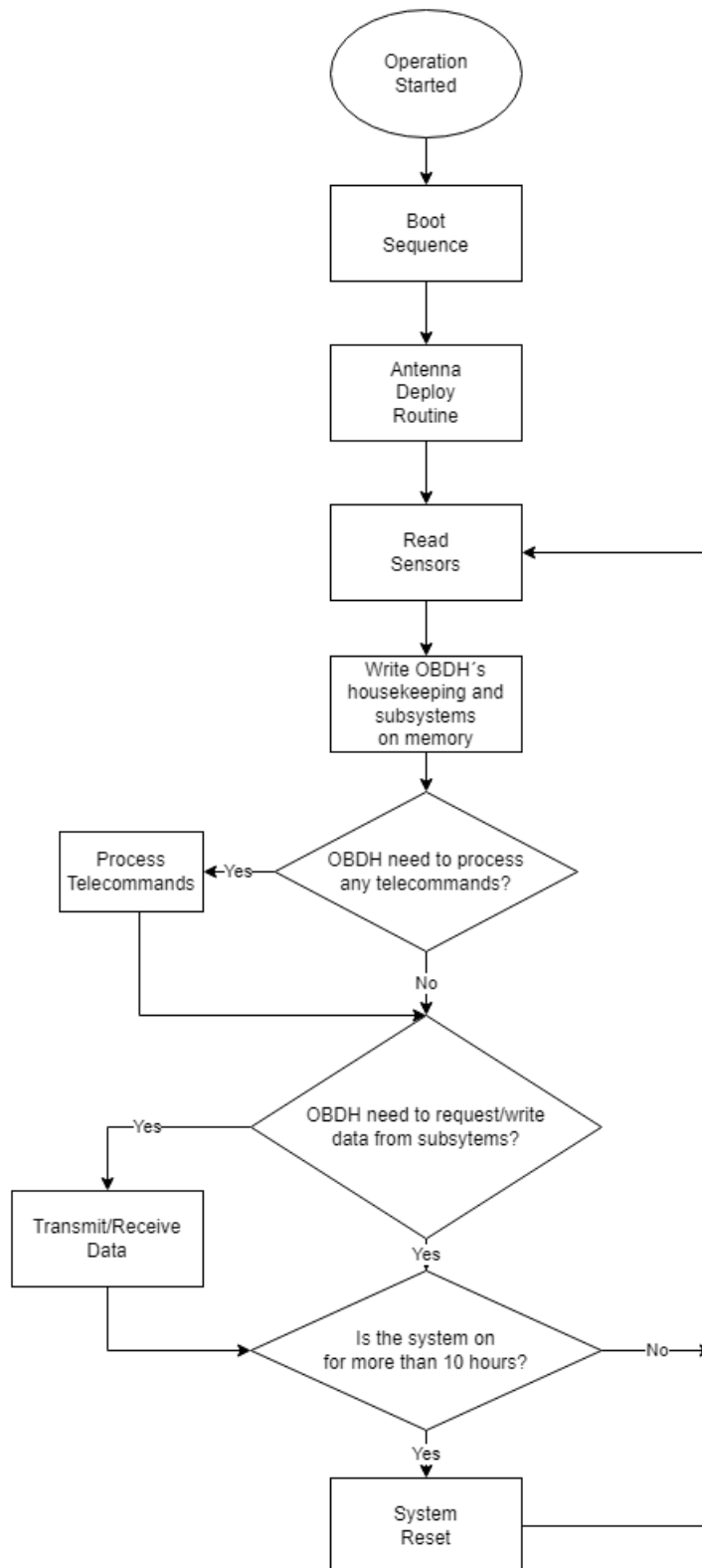


Figure 2.4: OBDH 2.0 operation flowchart

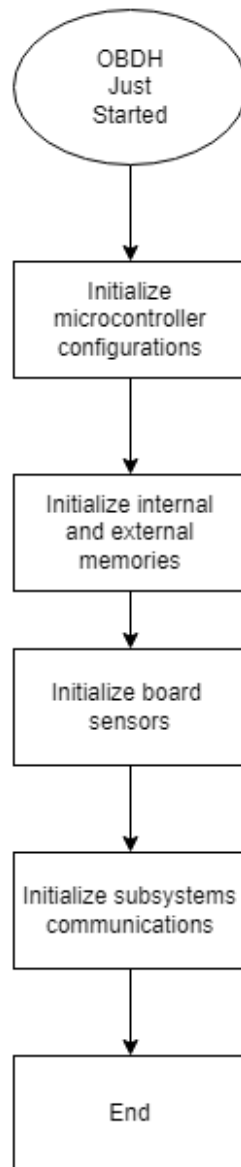


Figure 2.5: OBDH's 2.0 boot sequence.

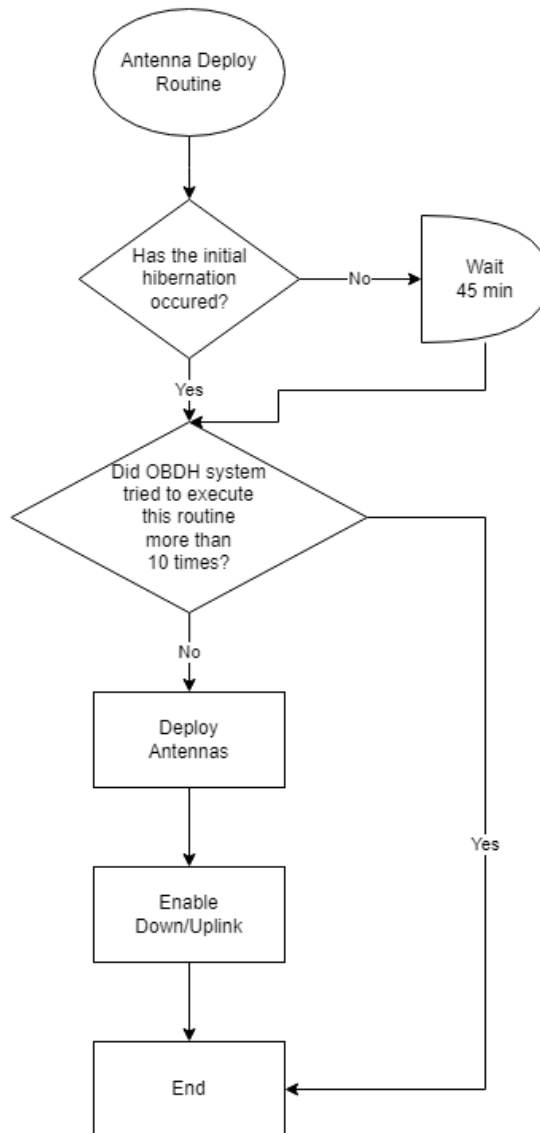


Figure 2.6: Antenna deploy routine

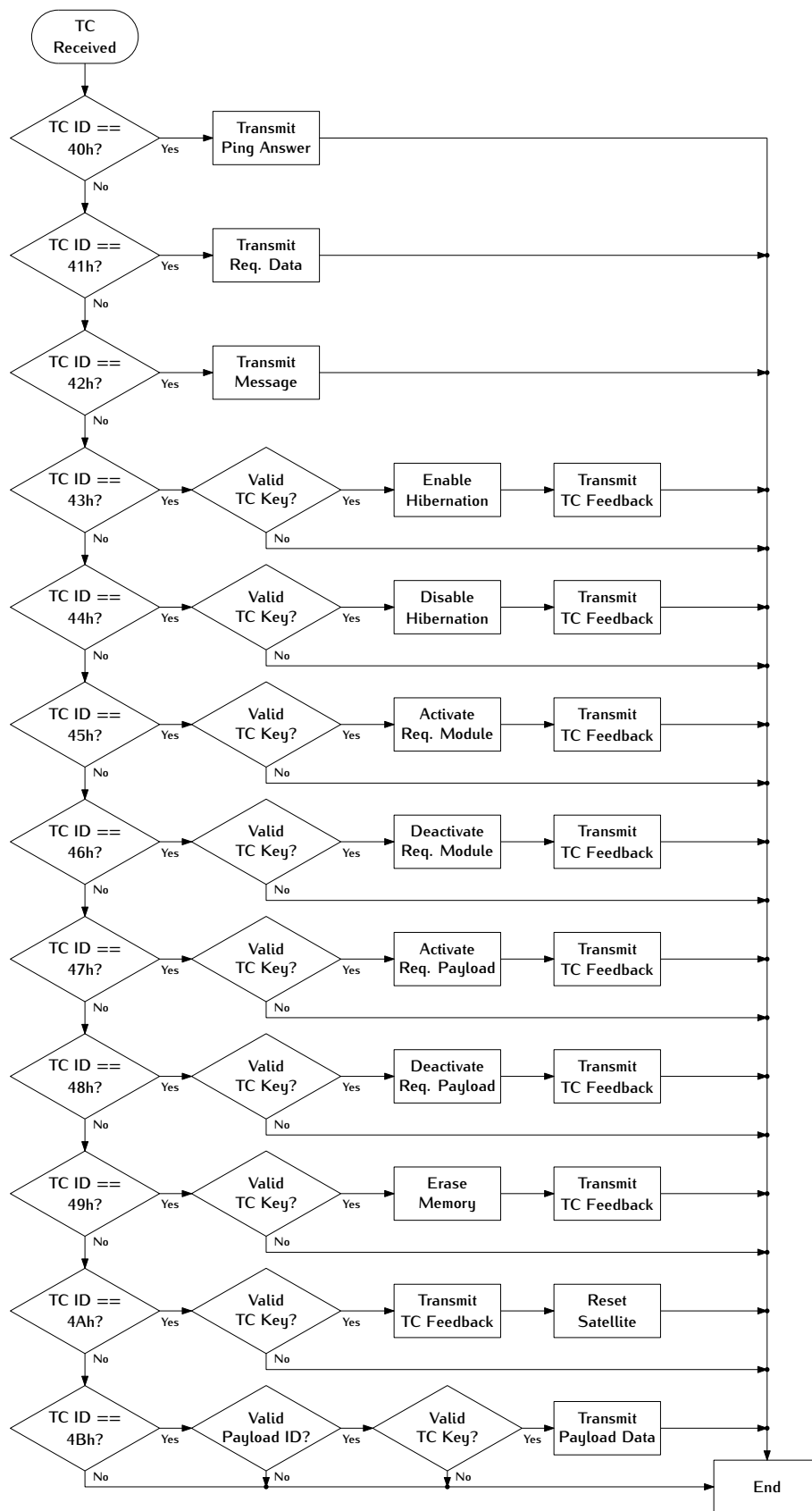


Figure 2.7: Telecommand's processing flowchart

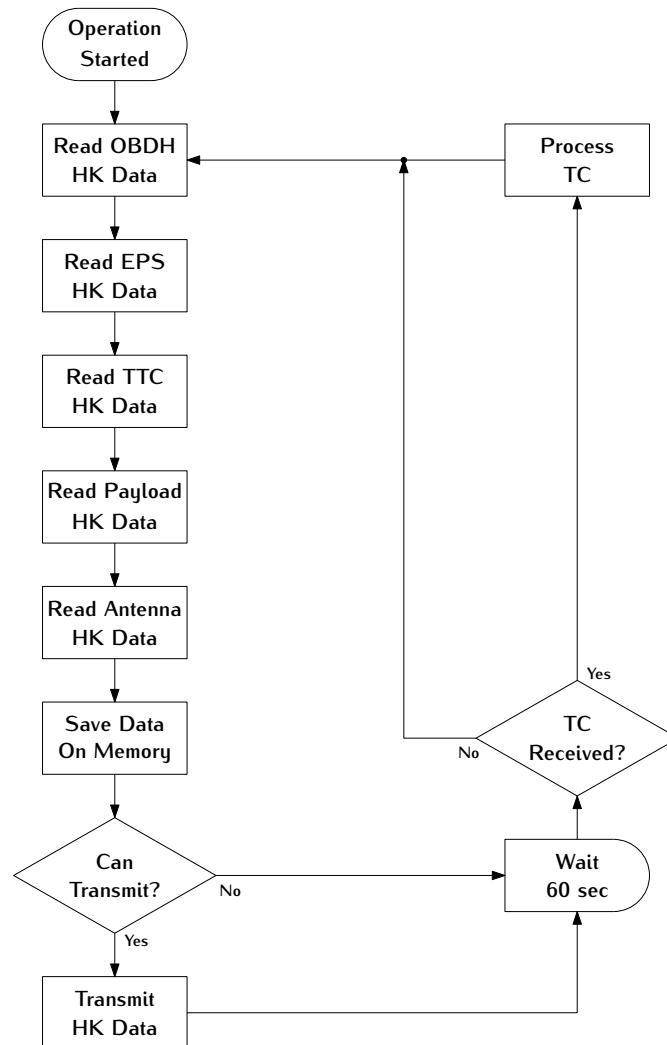


Figure 2.8: Transmit/Receive flowchart

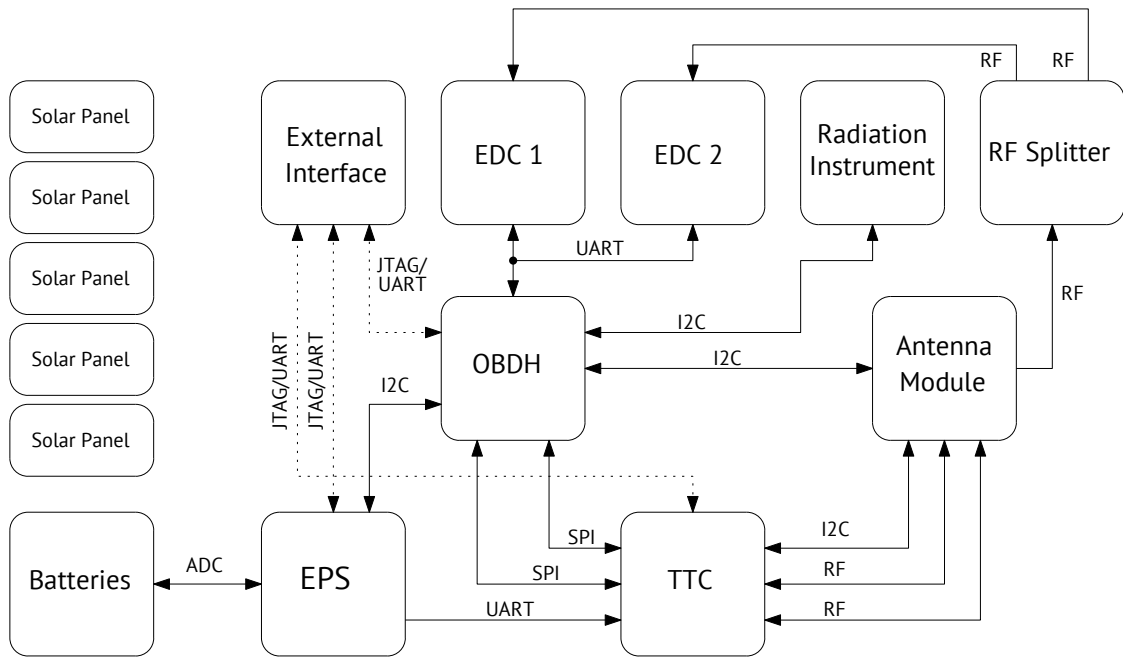


Figure 2.9: Data path diagram.

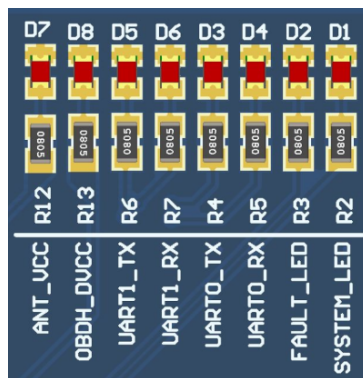


Figure 2.10: Available status LEDs.

CHAPTER 3

Hardware

The OBDH 2.0 architecture focuses on the low-power operation and low-cost production, maintaining performance and proposing different approaches to increase overall reliability. Therefore, the board was developed using these criteria, and the changes from the original design were necessary to improve bottlenecks and achieve the requirements of the further space mission. The Figure 2.2 presents the module architecture from the hardware perspective, including the main PCB components and interfaces: microcontroller, buffers, transceivers, memory, watchdog and voltage monitor, and connectors. The following sections describe the hardware design, interfaces, and standards in detail. The Figures 3.1, 3.2, and 3.3 present 3D-rendered images of the top, bottom, and side views of the board, respectively.

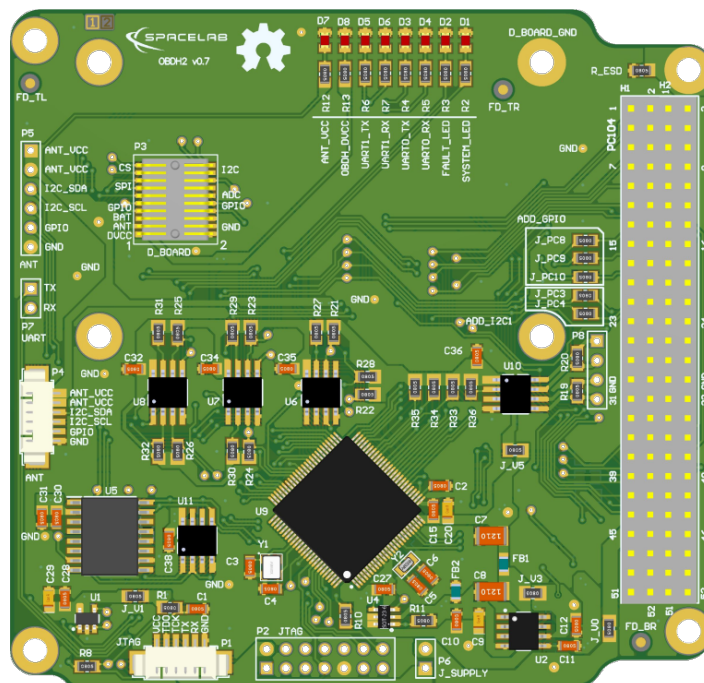


Figure 3.1: Top side of the PCB.

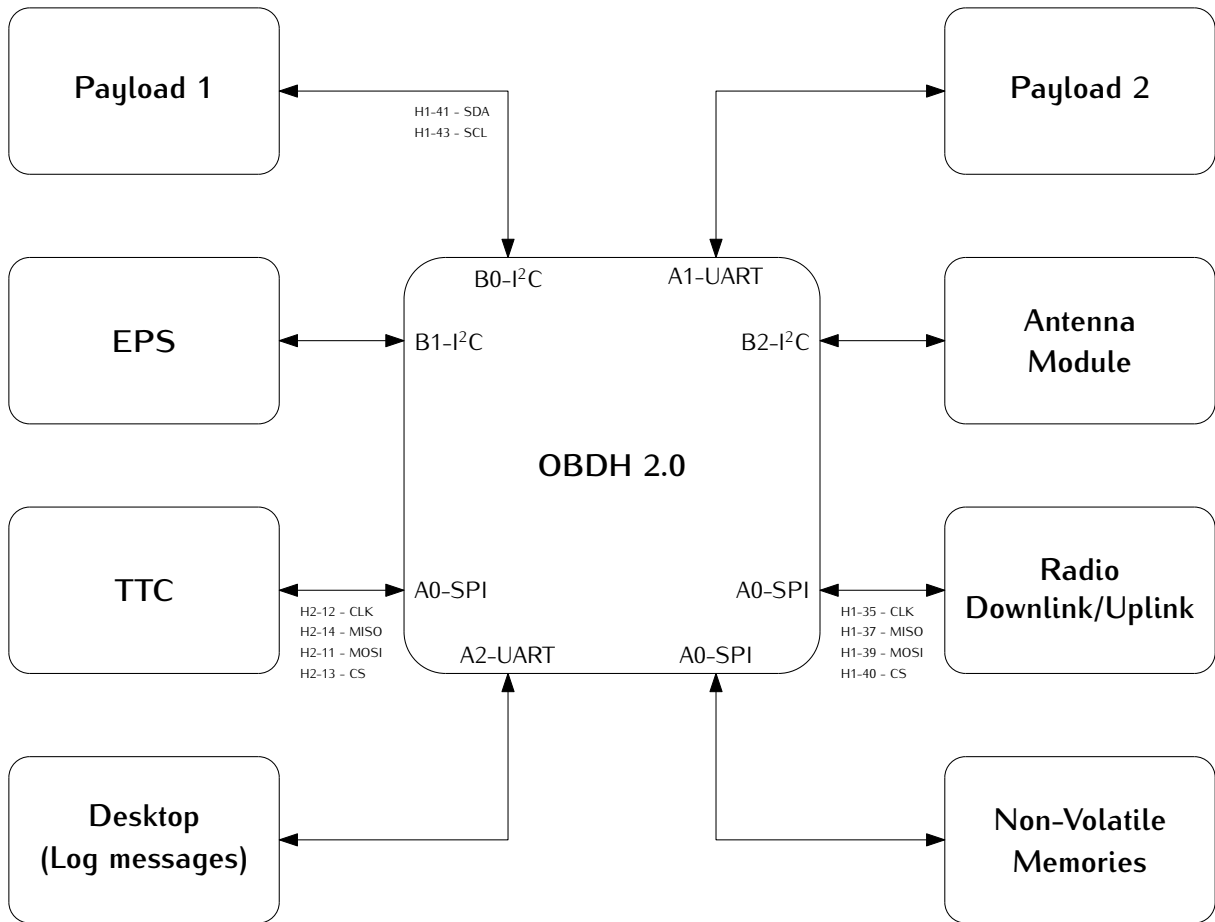


Figure 3.4: Interfaces diagram.

Peripheral	USCI	Protocol	Comm. Protocol
TTC	A0	SPI	Register read/write
Radio (downlink/link)	A0	SPI	Radio config./NGHam
NOR Memory	A0	SPI	-
FRAM Memory	A0	SPI	-
Payload port	A1	UART	¹
PC (log messages)	A2	UART	ANSI messages
Payload port	B0	I ² C	¹
EPS	B1	I ² C	Register read/write
Antenna Module	B2	I ² C	-

Table 3.1: Boards interfaces.

topics describe these interfaces and present the pinout of the connectors.

3.2.1 PC-104

The connector PC-104 is a junction of two double-row 28H headers (*SSW-126-04-G-D*). These connectors create a solid 104-pin interconnection across the different satellite

¹The communication protocol of the payload ports depends of the used payload.

modules. The Figure 3.5 shows the PC-104 interface from the bottom side of the PCB, which allows visualizing the simplified label scheme in the board. Also, the Table 3.2 provides the connector pinout² for the pins that are connected to the module.

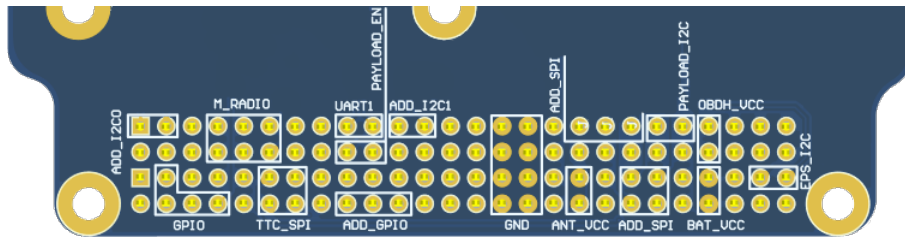


Figure 3.5: Bottom view of PC-104 and simplified labels

Pin [A-B]	H1A	H1B	H2A	H2B
1-2	-	-	-	-
3-4	-	-	GPIO_4	GPIO_5
5-6	-	-	-	-
7-8	GPIO_0	GPIO_1	-	GPIO_6
9-10	GPIO_2	-	-	-
11-12	GPIO_3	GPIO_7	SPI_0_MOSI	SPI_0_CLK
13-14	-	-	SPI_0_CS_1	SPI_0_MISO
15-16	-	-	-	-
17-18	UART_1_RX	GPIO_8	-	-
19-20	UART_1_TX	GPIO_9	-	-
21-22	-	-	-	-
23-24	-	-	-	-
25-26	-	-	-	-
27-28	-	-	-	-
29-30	GND	GND	GND	GND
31-32	GND	GND	GND	GND
33-34	-	-	-	-
35-36	SPI_0_CLK	-	VCC_3V3_ANT	VCC_3V3_ANT
37-38	SPI_0_MISO	-	-	-
39-40	SPI_0_MOSI	SPI_0_CS_0	-	-
41-42	I2C_0_SDA	-	-	-
43-44	I2C_0_SCL	-	-	-
45-46	VCC_3V3	VCC_3V3	VCC_BAT	VCC_BAT
47-48	-	-	-	-
49-50	-	-	I2C_1_SDA	-
51-52	-	-	I2C_1_SCL	-

Table 3.2: PC-104 connector pinout.

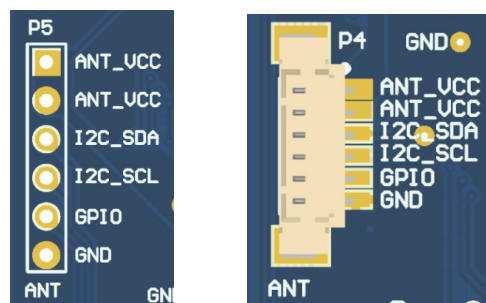
²This pinout is simplified since additional interfaces were omitted. Refer to *option sheet* in chapter 5.

3.2.2 Antenna Module

The communication with the Antenna module is performed through external connectors, which are presented in the Figure 3.6. Both connectors have the same connections, but the 3.6(a) (6H header) is used for development, and the 3.6(b) (6P picoblade) as the connector for the flight model. This interface consists of a dedicated I2C, power supply, and GPIO, described in the Table 3.3.

Pin	Row
1	VCC_3V3_ANT
2	VCC_3V3_ANT
3	I2C_SDA
4	I2C_SCL
5	GPIO
6	GND

Table 3.3: Antenna module connectors pinout.



(a) Debug interface of the antenna module.

(b) Main interface of the antenna module.

Figure 3.6: Antenna module connectors.

3.2.3 Programmer and Debug

The interface with the microcontroller programmer is performed through external connectors, which are presented in the Figure 3.7. Both connectors have the same JTAG and UART interfaces. However, the 14H header is used during development, and the 6P picoblade (provides a more compact and reliable attachment) as the connector for the flight model, which is described in the Table 3.4 and Table 3.5, respectively. This interface consists of a dedicated debug UART, a JTAG, and an external power supply. The debug UART connection has another access point in a dedicated 2H header (P7), as shown in Figure 3.8. Also, to use this external supply, it is necessary to connect both pins of a 2H header jumper (P6).



Figure 3.7: Programmer (P1 and P2) and jumper (P6) connectors.

Pin [A-B]	Row A	Row B
1-2	TDO_TDI	VCC_3V3
3-4	-	-
5-6	-	-
7-8	TCK	-
9-10	GND	-
11-12	-	UART_TX
13-14	-	UART_RX

Table 3.4: Programmer header connector pinout.

Pin	Row
1	VCC_3V3
2	TDO_TDI
3	TCK
4	UART_TX
5	UART_RX
6	GND

Table 3.5: Programmer picoblade connector pinout.

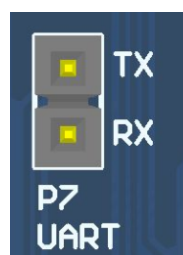


Figure 3.8: Dedicated UART debug connectors (P7).

3.2.4 Daughterboard

The daughterboard interface uses the Samtec FSI-110-D connector [4], which can be seen in the Figure 3.9. This connector has metal contacts in the format of flexible arcs and four polymer guide pins (a pair for the top and bottom). When the daughterboard is attached, there is some pressure on the metal contacts that bend and create a meaningful

pin connection to the daughterboard copper pads³. A picture of this connector on the PCB can be seen in Figure 3.10.

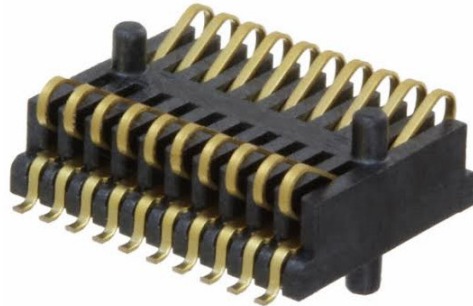


Figure 3.9: Samtec FSI-110-03-G-D-AD connector.

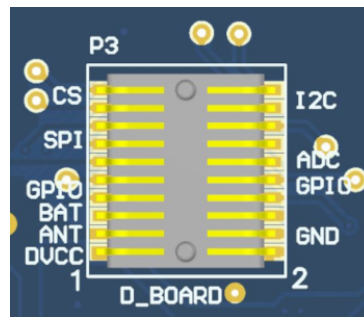


Figure 3.10: Daughterboard connector (P3).

The pinout of the daughterboard interface is available in the Table 3.6. There are different power supply lines (OBDH, Antenna, and battery), communication buses (I2C and SPI), GPIO, and ADC interfaces available. Besides the GPIO and ADC pins, the other interfaces are shared with other modules and peripherals.

Guidelines

The recommended shape and size of the daughterboard can be seen in the Figure 3.11. Besides that, there are mandatory and suggested elements placement: four M3 holes for mechanical attachment, required; contact connector pads (in light gray on the bottom layer), required; two debug headers on the left and bottom sides, suggested; and a general purpose flight model picoblade suggested.

3.3 Microcontroller

The OBDH 2.0 uses a low-power and low-cost microcontroller family from Texas Instruments; the MSP430F6659 [5]. This device provides sufficient performance for low and medium-complexity software and algorithms, allowing the module to execute the required

³These daughterboard pads are similar to the ones used as a footprint in the OBDH, despite a slightly bigger size.

Pin [A-B]	Row A	Row B
1-2	VCC_3V3	GND
3-4	VCC_3V3_ANT	GND
5-6	VCC_BAT	GND
7-8	GPIO_0	GPIO_1
9-10	GPIO_2	GPIO_3
11-12	SPI_0_CLK	ADC_0
13-14	SPI_0_MISO	ADC_1
15-16	SPI_0_MOSI	ADC_2
17-18	SPI_0_CS_0	I2C_2_SDA
19-20	SPI_0_CS_1	I2C_2_SCL

Table 3.6: Daughterboard connector pinout.

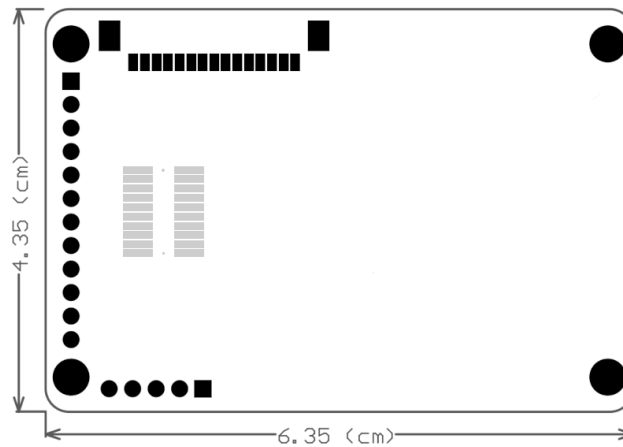


Figure 3.11: Recommended shape and size of the daughterboard.

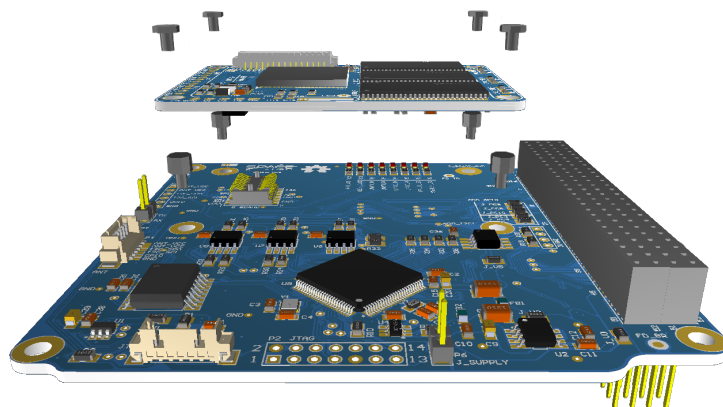


Figure 3.12: Illustrative daughterboard integration.

tasks. The Table 3.7 presents a summary of the main available features and Figure 3.13 shows the internal subsystems, descriptions, and peripherals. The microcontroller interfaces, configurations, and auxiliary components are described in the following topics.

Flash	SRAM	Timers	USCI	ADC	DAC	GPIO
512KB	64KB	2	6 (SPI / I2C / UART)	12	2	74

Table 3.7: Microcontroller features summary.

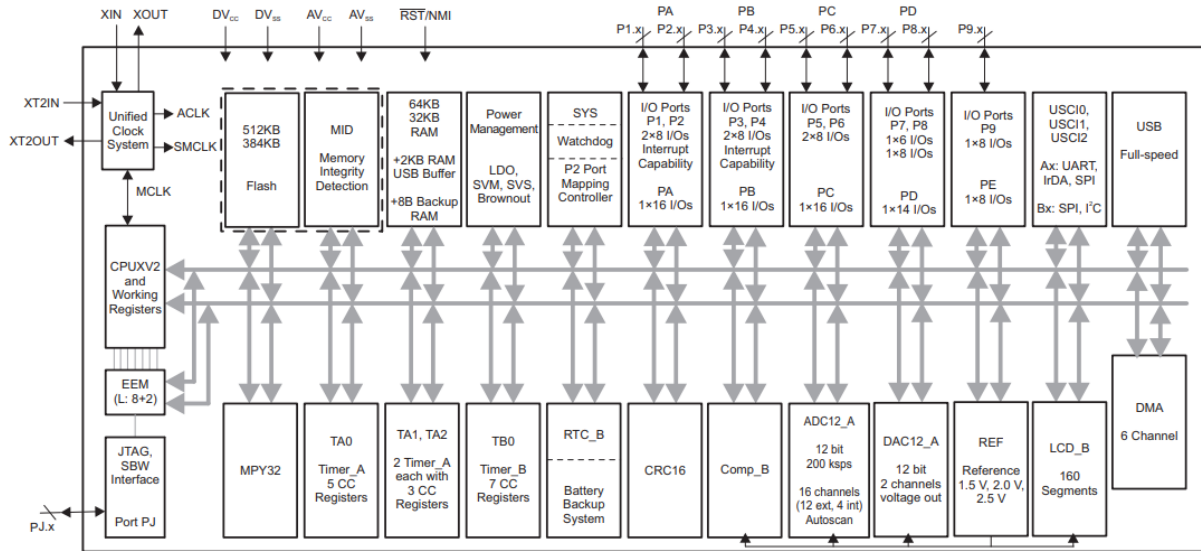


Figure 3.13: Microcontroller internal diagram.

3.3.1 Interfaces Configuration

The microcontroller has 6 Universal Serial Communication Interfaces (USCI) that can be configured to operate with different protocols and parameters. These interfaces are connected to different modules and peripherals, as presented in the Figure 3.4. The Table 3.8 describes each interface configuration.

Interface	Protocol (Index)	Mode	Word Length	Data Rate	Configuration
USCI_A0	SPI	Master	8 bits	1 Mbps	Phase: High Polarity: Low
USCI_A1	UART1	-	8 bits	115200 bps	Stop bits: 1 Parity: None
USCI_A2	UART0	-	8 bits	115200 bps	Stop bits: 1 Parity: None
USCI_B0	I2C0	Master	8 bits	100 kbps	Adr. len: 7 bits
USCI_B1	I2C1	Master	8 bits	100 kbps	Adr. len: 7 bits
USCI_B2	I2C2	Master	8 bits	100 kbps	Adr. len: 7 bits

Table 3.8: USCI configuration.

3.3.2 Clocks Configuration

Besides the internal clock sources, the microcontroller has two dedicated clock inputs for external crystals: the main clock and the auxiliary. A 32 MHz crystal and a 32.769 kHz

are connected to these inputs. The first source is used for generating the Master Clock (MCLK) and the Subsystem Master Clock (SMCLK), which are used by the CPU and the internal peripheral modules. The second source is used for generating the Auxiliary Clock (ACLK) that handles the low-power modes and might be used for peripherals.

3.3.3 Pinout

An illustration of the microcontroller pinout positions can be seen in the Figure 3.14. The Table 3.9 presents the OBDH 2.0 microcontroller pins assignment.

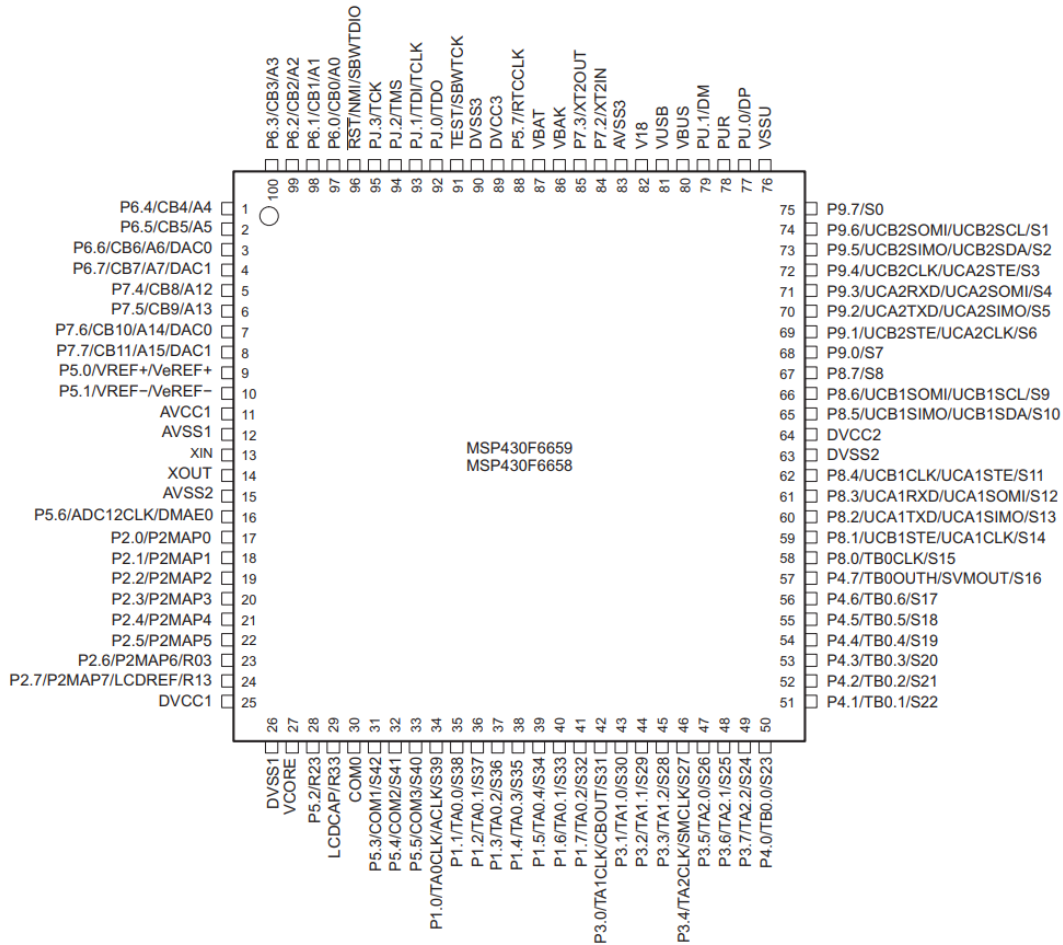


Figure 3.14: Microcontroller pinout positions.

Pin Code	Pin Number	Signal
P1.0	34	MAIN_RADIO_ENABLE
P1.1	35	MAIN_RADIO_GPIO0
P1.2	36	MAIN_RADIO_GPIO1
P1.3	37	MAIN_RADIO_GPIO2
P1.4	38	MAIN_RADIO_RESET
P1.5	39	MAIN_RADIO_SPI_CS
P1.6	40	TTC_MCU_SPI_CS

P1.7	41	-
P2.0	17	SPI_CLK
P2.1	18	I2C0_SDA
P2.2	19	I2C0_SCL
P2.3	20	-
P2.4	21	SPI_MOSI
P2.5	22	SPI_MISO
P2.6	23	VERSION_BIT0
P2.7	24	VERSION_BIT1
P3.0	42	I2C0_EN
P3.1	43	I2C1_EN
P3.2	44	I2C2_EN
P3.3	45	I2C0_READY
P3.4	46	I2C1_READY
P3.5	47	I2C2_READY
P3.6	48	PC104_GPI00
P3.7	49	PC104_GPI01
P4.0	50	PC104_GPI02
P4.1	51	PC104_GPI03
P4.2	52	MEM_HOLD
P4.3	53	MEM_RESET
P4.4	54	MEM_SPI_CS
P4.5	55	PC104_GPI04
P4.6	56	PC104_GPI05
P4.7	57	PC104_GPI06
P5.0	9	VREF
P5.1	10	AGND
P5.2	28	SYSTEM_FAULT_LED
P5.3	31	SYSTEM_LED
P5.4	32	PAYLOAD_0_ENABLE
P5.5	33	PAYLOAD_1_ENABLE
P5.6	16	-
P5.7	88	-
P6.0	97	D_BOARD_ADC0
P6.1	98	D_BOARD_ADC1
P6.2	99	D_BOARD_ADC2
P6.3	100	OBDH_CURRENT_ADC
P6.4	1	OBDH_VOLTAGE_ADC
P6.5	2	D_BOARD_SPI_CS0
P6.6	3	D_BOARD_SPI_CS1
P6.7	4	-
P7.0	-	-
P7.1	-	-
P7.2	84	XT2_N
P7.3	85	XT2_P
P7.4	5	D_BOARD_GPI00

P7.5	6	D_BOARD_GPIO1
P7.6	7	D_BOARD_GPIO2
P7.7	8	D_BOARD_GPIO3
P8.0	58	-
P8.1	59	-
P8.2	60	UART1_TX
P8.3	61	UART1_RX
P8.4	62	-
P8.5	65	I2C1_SDA
P8.6	66	I2C1_SCL
P8.7	67	ANTENNA_GPIO
P9.0	68	FRAM_WP
P9.1	69	FRAM_SPI_CS
P9.2	70	UART0_TX
P9.3	71	UART0_RX
P9.4	72	WDI_EXT
P9.5	73	I2C2_SDA
P9.6	74	I2C2_SCL
P9.7	75	MR_WDOG
PJ.0	92	TP21
PJ.1	93	TP22
PJ.2	94	TP23
PJ.3	95	TP24
-	13	XT1IN
-	14	XT1OUT
-	96	JTAG_TDO_TDI
-	91	JTAG_TCK

Table 3.9: Microcontroller pinout and assignments.

3.4 External Watchdog

In addition to the internal watchdog timer of the microcontroller, to ensure a system reset in case of a software freeze, an external watchdog circuit is being used. For that, the TPS3823 IC from Texas Instruments [6] was chosen. This IC is a voltage monitor with a watchdog timer feature. This circuit can be seen in the Figure 3.15.

This circuit works this way: if the WDI pin remains high or low longer than the timeout period, then reset is triggered. The timer clears when reset is asserted or when WDI sees a rising or falling edge.

The watchdog timer task clears the TPS3823 timer by toggling the WDI pin every 100 *ms*. If the WDI pin state stays unmodified for more than 1600 *ms*, the reset pin is cleared, and the microcontroller is reset.

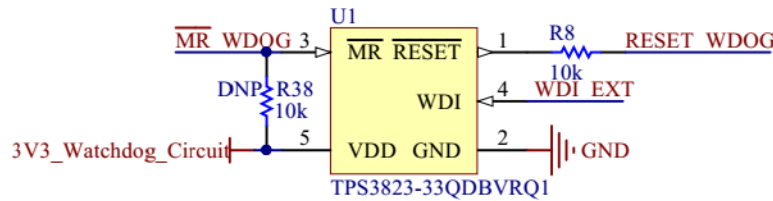


Figure 3.15: External watchdog timer circuit.

3.5 Non-Volatile Memories

There are two non-volatile memories available on the module: one flash NOR memory and one FRAM memory.

3.5.1 Flash NOR

The flash NOR non-volatile memory model is the Micron MT25QL01GBBB, which is composed of a NOR flash architecture with 1 Gb of capacity (or 128 MB) and features extended SPI configurations. As seen in Figure 3.4, an SPI bus is used to communicate with this peripheral, using the Table 3.8 configurations. Also, some control pins are connected to microcontroller GPIOs: HOLD#, RESET#, and W#.

When RESET# is driven LOW, the device is reset, and the outputs are tri-stated. The HOLD# signal pauses serial communications without deselecting or resetting the device; outputs are tri-stated, and inputs are ignored. The W# signal handles as write protection, freezes the status register, turning its non-volatile bits read-only and preventing the write operation from being executed.

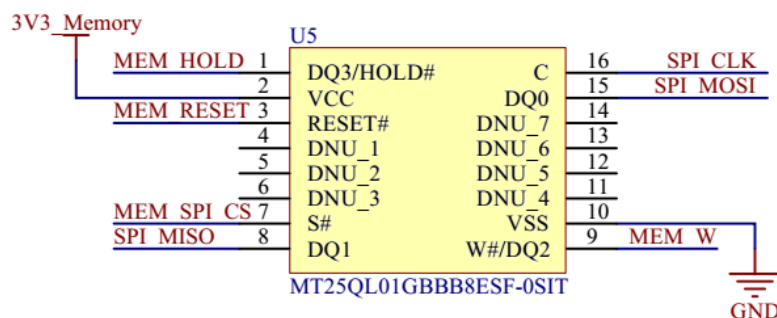


Figure 3.16: External memory circuit.

3.5.2 FRAM

The EXCELON™ Auto CY15X102QN is an automotive grade, 2Mb non-volatile memory employing an advanced ferroelectric process. A ferroelectric random access memory or F-RAM is non-volatile and performs reads and writes similar to RAM. It provides reliable data retention for 121 years. The schematics of the memory can be seen in Figure 3.17, an SPI bus is used to communicate with this peripheral.

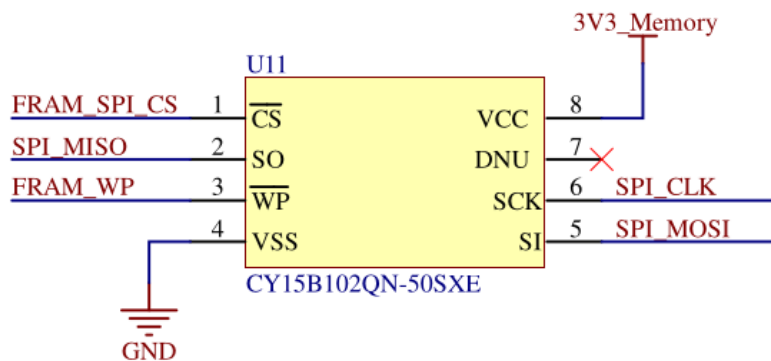


Figure 3.17: FRAM memory circuit.

3.6 I2C Buffers

The microcontroller I2C interfaces have dedicated IC buffers, which improve the signal quality throughout the various connectors and offer reliability enhancements since it protects the bus in case of failures. This measure was adopted in all the satellite modules due to previous failures in I2C buses. Using this scheme, the modules connected through this protocol might have shared connections without losing performance or reliability.

The buffer selected for this function is the Texas Instruments TCA4311 device. Besides the I2C inputs and outputs, it features control and status signals that are connected to GPIOs in the microcontroller: an enable and an operation-ready status. Also, both inputs and outputs in these I2C lines have external pull-up resistors.

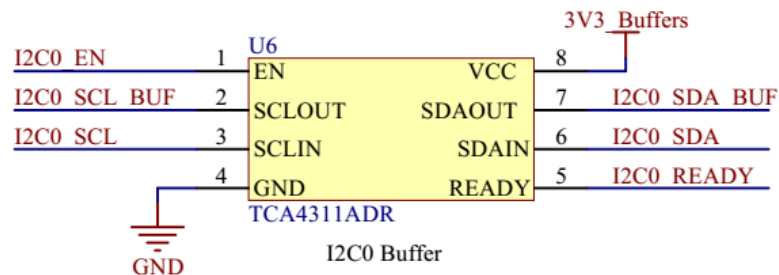


Figure 3.18: I2C buffer circuit.

3.7 RS-485 Transceiver

The module features an RS-485 interface connected to a 4H header (P8). This interface uses a transceiver (THVD1451) to convert the incoming RS-485 signals to UART and vice-versa. The outputs are 120 Ω differential pairs that have termination resistors before connecting to the header pins.

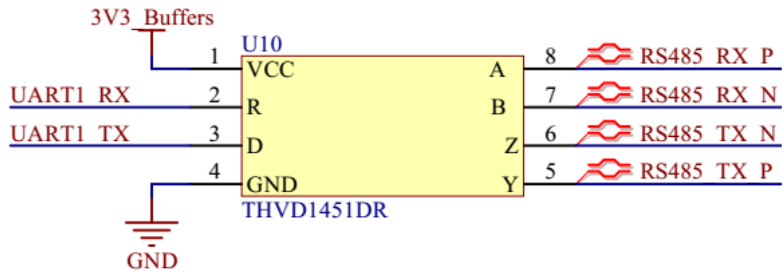


Figure 3.19: RS-485 transceiver circuit.

3.8 Voltage and Current Sensors

To monitor the board's overall current and voltage, the module has a current sensor using a Maxim Integrated IC (MAX9934) and a buffered voltage divider circuit with a Texas Instruments IC (TLV341A). These circuits have direct analog outputs that are connected to ADC inputs. The microcontroller's internal ADC peripheral has a dedicated input for a voltage reference, which is connected to the REF5030A IC. This device generates a precise 3 V output that enhances the measures and conversions performed by the microcontroller.

CHAPTER 4

Firmware

4.1 Product tree

The product tree of the firmware part of the OBDH 2.0 module is available in Figure 4.1.

4.2 Dependencies

The firmware depends on external libraries to access the embedded hardware or to communicate with other modules. A list of these libraries and the used version is available in Table 4.1.

Library	Version
MSP430 DriverLib	v2.91.11.01
FreeRTOS	v10.2.1

Table 4.1: External libraries and dependencies of the firmware.

4.3 Tasks

A list of the firmware tasks can be seen in the Table 4.2.

All these tasks are better described below.

4.3.1 Antenna deployment

This task deploys the Antenna module at the start of the mission.

4.3.2 Antenna reading

Reads the Antenna module status.

4.3.3 Beacon

The Beacon task transmits a data package containing the satellite's basic telemetry data every 60 seconds.

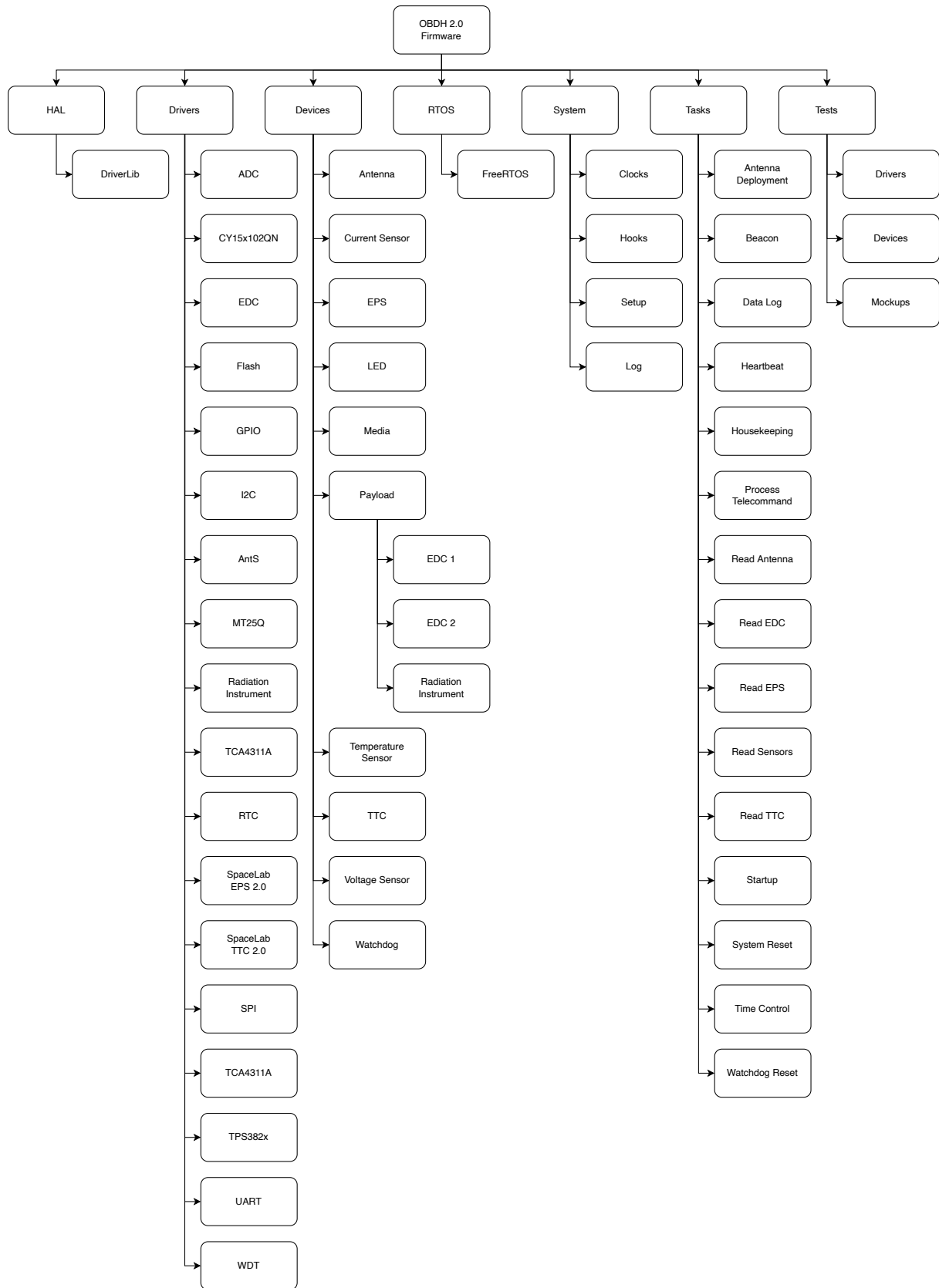


Figure 4.1: Product tree of the firmware of the OBDH 2.0 module.

Name	Priority	Initial delay [ms]	Period [ms]	Stack [bytes]
Antenna deployment	Highest	0	Aperiodic	150
Antenna reading	Medium	2000	60000	150
Beacon	High	10000	60000	1000
Data log	Medium	2000	600000	225
EDC reading	Medium	2000	60000	300
EPS reading	Medium	2000	60000	384
Heartbeat	Lowest	2000	500	160
Housekeeping	Medium	2000	10000	160
Read sensors	Medium	2000	60000	140
Startup (boot)	Highest	0	Aperiodic	350
System reset	Medium	0	36000000	128
Telecommand processing	High	10000	5000	500
Time control	Medium	1000	1000	128
TTC reading	Medium	2000	60000	384
Watchdog reset	Lowest	0	100	150

Table 4.2: Firmware tasks.

4.3.4 Data log

This task saves the housekeeping data of the satellite in flash memory every 10 minutes.

4.3.5 EDC reading

This task reads all the EDC packages and data.

4.3.6 EPS reading

This task reads all the EPS data and status.

4.3.7 Heartbeat

The heartbeat task keeps blinking a LED ("*System LED*" in Figure 2.10) at a rate of 1 Hz during the execution of the system. Its purpose is to give visual feedback on the execution of the scheduler. This task does not have a specific purpose on the flight version of the module (the flight version of the PCB does not have LEDs).

4.3.8 Housekeeping

This task reads all the important OBDH data and status.

4.3.9 Read sensors

This task reads the internal sensors of the OBDH every 60 seconds.

4.3.10 Startup (boot)

This task is the first executed task when the system starts. All devices, libraries, and data structures are initialized in this task. When the execution is done, the remaining tasks of the system are allowed to execute.

4.3.11 System reset

This task resets the microcontroller by software every 10 hours. This can be useful to clean up possible wrong values in variables, repeat the antenna deployment routine (limited to n times), clean up the RAM, etc.

4.3.12 Telecommand processing

This task processes all the telecommands; it needs to have a really short period for a more responsive operation.

4.3.13 Time control

This task is responsible for the time management of the system. At every second, it increments the system time (epoch). Also, it saves the current system time in the non-volatile memory every minute.

4.3.14 TTC reading

This task reads all the TTC data and status.

4.3.15 Watchdog reset

This task resets the internal and external watchdog timer every 100 ms. The internal watchdog has a maximum count time of 500 ms, and the external watchdog has a maximum of 1600 ms (see chapter 3 for more information about the watchdog timers).

To prevent the system to not reset during an anomaly on some task (like an execution time longer than planned), this task has the lowest possible priority: 0.

4.4 Variables and Parameters

The internal variables and parameters of the OBDH firmware can be seen in Table 4.3.

ID	Name/Description	Type
0	Time counter in milliseconds	uint32
1	Temperature of the μC in Kelvin	uint16
2	Input current in mA	uint16
3	Input voltage in mV	uint16
	Last reset cause:	
	- 0x00 = No interrupt pending	
	- 0x02 = Brownout (BOR)	
	- 0x04 = RST/NMI (BOR)	

- 0x06 = PMMSWBOR (BOR)	
- 0x08 = Wakeup from LPMx.5 (BOR)	
- 0x0A = Security violation (BOR)	
- 0x0C = SVSL (POR)	
- 0x0E = SVSH (POR)	
- 0x10 = SVML_OVP (POR)	
- 0x12 = SVMH_OVP (POR)	
- 0x14 = PMMSWPOR (POR)	
- 0x16 = WDT time out (PUC)	
- 0x18 = WDT password violation (PUC)	
- 0x1A = Flash password violation (PUC)	
- 0x1C = Reserved	
- 0x1E = PERF peripheral/configuration area fetch (PUC)	
- 0x20 = PMM password violation (PUC)	
- 0x22 to 0x3E = Reserved	
5 Reset counter	uint16
6 Last valid telecommand (uplink packet ID)	uint8
7 Temperature of the radio in Kelvin	uint16
8 RSSI of the last valid telecommand	uint16
9 Temperature of the antenna in Kelvin	uint16
Antenna status bits:	
- Bit 15: The antenna 1 is deployed (0) or not (1)	
- Bit 14: Cause of the latest activation stop for antenna 1	
- Bit 13: The antenna 1 deployment is active (1) or not (0)	
- Bit 11: The antenna 2 is deployed (0) or not (1)	
- Bit 10: Cause of the latest activation stop for antenna 2	
- Bit 9: The antenna 2 deployment is active (1) or not (0)	
- Bit 8: The antenna is ignoring the deployment switches (1) or not (0)	
10 - Bit 7: The antenna 3 is deployed (0) or not (1)	uint16
- Bit 6: Cause of the latest activation stop for antenna 3	
- Bit 5: The antenna 3 deployment is active (1) or not (0)	
- Bit 4: The antenna system independent burn is active (1) or not (0)	
- Bit 3: The antenna 4 is deployed (0) or not (1)	
- Bit 2: Cause of the latest activation stop for antenna 4	
- Bit 1: The antenna 4 deployment is active (1) or not (0)	
- Bit 0: The antenna system is armed (1) or not (0)	
11 Hardware version	uint8
12 Firmware version (ex.: "v1.2.3" = 0x00010203)	uint32
13 Mode ("Normal" = 0, "Hibernation" = 1)	uint8
14 Timestamp of the last mode change	uint32
15 Mode duration in sec. (valid only in hibernation mode)	uint32
16 Initial hibernation executed	boolean
17 Initial hibernation time counter (minutes)	uint8
18 Antenna deployment executed	boolean
19 Antenna deployment counter	uint8

Table 4.3: Variables and parameters of the OBDH 2.0.

4.5 Telemetry

4.5.1 Beacon

The beacon packet is transmitted every 1 minute and contains basic telemetry data of the satellite. The content of this packet can be seen in Table 4.4.

- Period: 60 seconds
- Band: UHF
- Condition to operate: Always on

Parameter	Content	Length [bytes]
Packet ID	10h	1
Satellite callsign	"0PY0EGU"	7
μ C temperature	Raw μ C temperature	2
μ C voltage	Raw μ C voltage	2
μ C current	Raw μ C current	2
Last reset cause	Last reset cause ID	1
System time	System time in ticks	4
Radio temperature	Raw radio temperature	4
Last TC RSSI	Raw RSSI value	2
Last received TC	Last received TC ID	1
Battery 1 voltage	Raw battery 1 voltage	2
Battery 2 voltage	Raw battery 2 voltage	2
Battery current	Raw battery current	2
Battery charge	Raw battery charge	2
Total	-	34

Table 4.4: Beacon packet.

4.5.2 EDC Information

Telemetry packages from EDC can be seen in Table 4.5.

4.5.3 EDC Samples

The EDC samples are **TBD** bytes long and are transmitted in Y packets with 219 bytes each.

4.6 Telecommands

The system telecommands can be seen in Table 4.7.

Parameter	Content	Len. [bytes]
Packet ID	11h	1
Satellite callsign	"0PY0EGU"	7
PTT Decoder		
Time tag	PTT signal receiving time	4
Error code	Error code	1
Carrier frequency	Carrier frequency	2
Carrier Abs	Carrier amplitude at ADC interface output	2
Message length	User message length in bytes	1
User message	ARGOS-2 PTT-A2 user message	35
HK Info		
Current time	Current time since J2000 epoch	4
Elapsed time	Elapsed time since last reset	4
Current supply	System current supply in mA	2
Voltage supply	System voltage supply in mV	2
Temperature	EDC board temperature	1
PLL sync bit	RF front end LO..	1
ADC RMS	RMS level at front-end output	2
Num of RX PTT	Generated PTT packages since last initialization	1
Max		1
Memory error count		1
System State		
Current time		4
PTT available	Number of PTT Package available for reading	1
PTT is paused	PTT decoder task status	1
Sampler state	ADC sampler state	1
Total	-	79

Table 4.5: EDC information packet.

4.6.1 Enter hibernation

This telecommand activates the hibernation mode in a satellite. During the hibernation mode, no transmissions are made by the satellite, it keeps just listening for new incoming packets (reception). The satellite will stay in hibernation mode for a custom period (1 to 65536 minutes), or until a "Leave Hibernation" mode is received. This is a private telecommand, a key is required to send it. Beyond the packet ID and the source callsign (or address), the number of minutes (2 bytes long) is also transmitted.

4.6.2 Leave hibernation

This telecommand complements the "enter hibernation" telecommand by deactivating the hibernation mode in the satellite. When a satellite receives this telecommand it enables the transmission again immediately. This is also a private telecommand; a specific key is required to send it. There is no additional content to this telecommand packet, just the

Parameter	Content	Length [bytes]
Packet ID	12h	1
Satellite callsign	"0PY0EGU"	7
Time tag	Elapsed time since J2000 epoch	4
Packet counter	ADC sample packet number	1
I sample[n]	First ADC I-sample	2
Q sample[n]	First ADC Q-sample	2
...
I sample[n+102]	First ADC I-sample	2
Q sample[n+102]	First ADC Q-sample	2
Total	-	219

Table 4.6: EDC samples packet.

Name	Parameters	Access
Enter hibernation	Hibernation period in seconds	Private
Leave hibernation	None	Private
Activate beacon	None	Private
Deactivate beacon	None	Private
Activate downlink	None	Private
Deactivate downlink	None	Private
Activate EDC	None	Private
Deactivate EDC	None	Private
Get EDC info	None	Private
Activate Radiation instrument	Experiment period in seconds	Private
Deactivate Radiation instrument	None	Private
Set system time	Time value (epoch)	Private
Ping	None	Public
Message broadcast	ASCII message	Public
Request data	Data flags	Public

Table 4.7: System telecommand.

Parameter	Content	Length [bytes]
Packet ID	20h	1
Ground station callsign	Any callsign (ASCII, filled with "0"s)	7
Hibernation period	Period in minutes (1 to 65535)	2
Key	Telecommand key (ASCII)	10
Total	-	20

Table 4.8: Enter hibernation telecommand.

packet ID and the source callsign (or address).

Parameter	Content	Length [bytes]
Packet ID	21h	1
Ground station callsign	Any callsign (ASCII, filled with "0"s)	7
Key	Telecommand key (ASCII)	10
Total	-	18

Table 4.9: Leave hibernation telecommand.

4.6.3 Activate beacon

This telecommand activates the beacon transmissions.

4.6.4 Deactivate beacon

This telecommand deactivates the beacon transmissions.

4.6.5 Activate EDC

This telecommand activates the EDC payload.

4.6.6 Deactivate EDC

This telecommand deactivates the EDC payload.

4.6.7 Get EDC info

This telecommand request information from the EDC payload. When received, the OBDH transmits the housekeeping and state frames of the EDC module (28 bytes). This telecommand does not require a key.

4.6.8 Activate Module

The "Activate Module" telecommand is a command to activate an internal module of the satellite. Each module has a unique ID that is passed as an argument of this telecommand's packet. The Table 4.10 shows the current used IDs. This is also a private telecommand, and a key is required to transmit it.

Module	ID number
Battery heater	1
Beacon	2
Periodic telemetry	3

Table 4.10: Leave hibernation telecommand.

4.6.9 Deactivate Module

The “Deactivate Module” telecommand complements the telecommand above. It works the same way and with the same parameters, but in this case, it has the purpose of deactivating a given module of the satellite.

4.6.10 Activate Radiation Instrument

This telecommand activates the Radiation Instrument.

4.6.11 Deactivate Radiation Instrument

This telecommand deactivates the Radiation Instrument.

4.6.12 Activate Payload

This telecommand is similar to the telecommand “Activate Module”, but in this case is used for the activate payloads of the satellite. Each satellite will have a list of IDs of the set of payloads.

This is also a private telecommand, and a key is required to transmit it.

4.6.13 Deactivate Payload

Same as the “Deactivate Module” telecommand, but for payloads.

4.6.14 Erase Memory

The telecommand “erase memory” erases all the content presented in the non-volatile memories of the onboard computer of a satellite. This is a private command, and a key is required to send it. No additional content is required in a erase memory telecommand packet, just the packet ID and the source callsign (or address).

4.6.15 Force Reset

This telecommand performs a general reset of the satellite. When received, the satellite reset all subsystems. This is a private telecommand, and a key is required to send this command to a satellite. There is no additional content in this packet, just the packet ID and the source callsign (or address).

4.6.16 Get Payload Data

This telecommand allows a ground station to download data from a specific satellite payload. The required fields are the payload ID, and, optionally, arguments to be passed to the payload. The IDs and arguments vary according to the satellite. This is a private telecommand, and a key is required to send it.

4.6.17 Set Parameter

This telecommand allows the configuration of specific parameters of a given satellite subsystem. The required fields are the ID of the subsystem to set (1 byte), the ID of the parameter to set (1 byte), and the new value of the parameter (4 bytes long). The possible IDs (subsystem and parameter) vary according to the satellite. This is a private telecommand, and a key is required to send it.

4.6.18 Get Parameter

The telecommand “Get Parameter” complements the “Set Parameter” telecommand. It has the purpose of reading specific parameters of a given subsystem. The required fields are the subsystem’s ID (1 byte) and the parameter ID (1 byte). The possible IDs (subsystem and parameter) vary according to the satellite. This is a private telecommand, and a key is required to send it.

4.6.19 Set system time

This telecommand sets the internal system time. This is useful for synchronizing the satellite time with a ground station installed on Earth.

4.6.20 Ping

The ping request telecommand is a simple command to test the communication with the satellite. When the satellite receives a ping packet, it will respond with another ping packet (with another packet ID, as defined in the downlink packets list). There are no additional parameters in the ping packet, just the packet ID and the source callsign (or address). It is also a public telecommand; anyone can send a ping request telecommand to a satellite.

Parameter	Content	Length [bytes]
Packet ID	22h	1
Ground station callsign	Any callsign (ASCII, filled with “0”s)	7
Total	-	8

Table 4.11: Ping telecommand.

Parameter	Content	Length [bytes]
Packet ID	12h	1
Satellite callsign	“PY0EGU”	7
Destination callsign	Requester callsign (ASCII, filled with “0”s)	7
Total	-	15

Table 4.12: Ping telecommand answer.

4.6.21 Message broadcast

The “broadcast message” is another public telecommand; no authentication or key is required to send this telecommand to a satellite. This command has the purpose of making a satellite transmit a custom message back to Earth. This can be useful for communication tasks, like a station sending data to another. There are two parameters in this telecommand: the destination callsign (or address), and the content of the message, which can be any sequence of ASCII characters or any byte value. There is a limit of 38 characters in the message field. The Table 4.13 shows the Telecommand parameters.

Parameter	Content	Length [bytes]
Packet ID	23h	1
Ground station callsign	Any callsign (ASCII, filled with “0”s)	7
Message	Message to broadcast (ASCII)	up to
Total	-	8

Table 4.13: Message broadcast telecommand.

4.6.22 Request data

The data request telecommand is a command to download data from the satellite. This command allows a ground station to get specific parameters from a given period (stored in the satellite’s non-volatile memory of the onboard computer). The list of possible parameters varies according to the satellite. The required fields of this telecommand are the parameter ID (1 byte), the start period in milliseconds (epoch, 4 bytes), and the end period in milliseconds (epoch, 4 bytes). This is a private telecommand, and a key is required to send it.

4.7 Operating System

The FreeRTOS 10 [7] is being used as an operating system. FreeRTOS is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors. Distributed freely under the MIT open-source license, FreeRTOS includes a kernel and a growing set of IoT libraries suitable for use across all industry sectors. FreeRTOS is built with an emphasis on reliability and ease of use.

The main configuration parameters of the operating system in this project are available in Table 4.14.

More details of the used configuration parameters can be seen in the file *firmware/-config/FreeRTOSConfig.h* from [3].

4.8 Hardware Abstraction Layer (HAL)

As the Hardware Abstraction Layer (HAL), the DriverLib [8] from Texas Instruments is begin used. It is the official API to access the registers of the MSP430 microcontrollers.

Parameter	Value	Unit
Version	v10.2.0	-
Tick rate (Hz)	1000	Hz
CPU clock (HZ)	32	MHz
Max. priorities	5	-
Heap size	40960	bytes
Max. length of task name	20	-

Table 4.14: FreeRTOS main configuration parameters.

The DriverLib is meant to provide a “software” layer to the programmer to facilitate a higher programming level than direct register accesses. By using the high-level software APIs provided by DriverLib, users can create powerful and intuitive code that is highly portable between devices within the MSP430 platform and different families in the MSP430/MSP432 platforms.

CHAPTER 5

Board Assembly

The OBDH2 has some DNP components to provide flashing, debugging, testing, or extra interfaces if needed. These components may be optional for the flight model of the board. The draftsman document can be viewed for more detailed information regarding their location and board dimensions [9].

5.1 Development Model

5.1.1 Debug and programming connectors

The P2 and P6 connectors are used for flashing and debugging the OBDH2 board. See again chapter 3 (Hardware) and subsection 3.2.3 (Programmer and Debug) for more information.

5.1.2 Status leds

As already exposed in the document OBDH2 has status LEDs to be used during the development and test phases. See again chapter 2 (System Overview) and subsection 2.3.1 (Status LEDs) for more information.

5.2 Flight Model

The flight model of the OBDH 2.0 boards follows a special assembly, some components are not soldered, like the LEDs and some connectors. The PCB is also fabricated with higher quality, using the Class 3 standard and a core material less sensitive to temperature changes. The silkscreen is also not printed on the board.

5.3 Custom Configuration

On the PC104 connector of OBDH2, there are some jumper resistors to enable extra I2C, SPI, and GPIO interfaces if desired. Note that the I2C0, I2C1, and SPI channels should not be used with shared devices. These components' corresponding tables and locations on the PCB are shown on Table 5.1 and Figures 5.1, 5.2 and 5.3.

Label	Interface
J_PC1	I2C0_SDA
J_PC2	I2C0_SCL
J_PC3	I2C1_SDA
J_PC4	I2C1_SCL
J_PC5	SPI_MOSI
J_PC6	SPI_MISO
J_PC7	SPI_CLK
J_PC8	GPIO
J_PC9	GPIO1
J_PC10	GPIO2
J_PC11	GPIO3

Table 5.1: Additional PC104 interfaces.

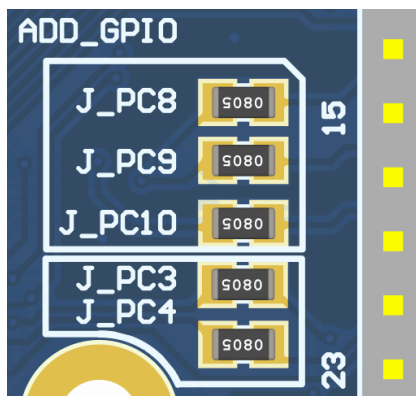


Figure 5.1: Additional GPIOs and I2C channel 1.

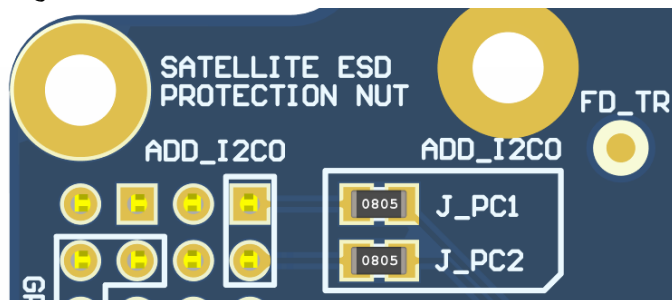


Figure 5.2: Additional I2C channel 0.



Figure 5.3: Additional GPIO and SPI channel.

CHAPTER 6

Usage Instructions

6.1 Powering the Board

Since the OBDH 2.0 is a service module within a satellite bus, to correctly provide its power supply, it requires an external 3.3 ± 0.2 V power input and a current capability of at least 100 mA (might change depending on the daughterboard requirements). As presented in the PC-104 and programming interface sections, some options are given to power the module to improve flexibility during development. The board has two power schemes: the JTAG interface for debugging, and the PC-104, for the flight configuration. The first case uses both P1 or P2 connectors as power input (besides the JTAG and UART interfaces) and requires a jumper connection in the P6 connector. The second uses the PC-104 pins H1-45 and H1-46 to provide the power, and the P6 connector should remain open. For pinout details, refer to the external connectors in the hardware chapter.

6.2 Log Messages

The OBDH 2.0 has a UART interface dedicated to debugging, described in Table 3.8. It follows a log system structure to improve the information provided in each message. The Figure 6.1 shows an example of the logging system, more specifically the initialization sequence. The messages use the following scheme: in green inside brackets, the timestamp; in magenta, the scope (or origin) of the log; and lastly the actual message, which might be white (info or note), yellow (warning), and red (error).

6.3 Daughterboards Installation

The daughterboard requirements might change for each application board attached. Then, it is important to check at least a minimal set of mandatory characteristics. First, it is important to verify mechanical parameters that concern size (recommended 63.5×43.5 mm), maximum height (no higher than 7 mm), screw attachment (refer to mechanical sheet [9]), and contact connector positioning. After this, the electrical interface must be checked (refer to subsection 3.2.4). There are 3 different power supply options, a 3.3 V source shared with the OBDH board itself, another 3.3 V source shared with the antenna deployer, and the main battery bus that ranges from 5.4 to 8.4 V. Lastly, depending on the application board design, it is necessary to check communication interface protocols and parameters, control inputs and outputs, and external interfaces with other modules.



```
 /dev/ttyUSB0 - PuTTY
OBDM 2.0 Copyright (C) 2020, SpaceLab;
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions.

Source code: https://github.com/spacelab-ufsc/obdm2
Documentation: https://github.com/spacelab-ufsc/obdm2/doc

*****
*****
*****  SpaceLab  *****
*****          *****
*****  OBDM 2.0  *****
*****          *****
*****
*****

=====
Version:      [ 0.3.13 ]
Status:      Development
Author:      SpaceLab <spacelab.ufsc@gmail.com>
=====

[ 196 ] Startup: FreeRTOS V10.2.0
[ 202 ] Startup: System clocks: MCLK=31981568 Hz, SMCLK=31981568 Hz, ACLK=32768 Hz
[ 216 ] LEDs: Initializing system LEDs...
[ 223 ] EPS: Initializing EPS device...
[ 230 ] EPS: Error during the initialization! (error 4294967295)
[ 241 ] Radio: Initializing radio device...
[ 2077 ] Startup: libcsp disabled!
[ 2092 ] Payload EDC: Error initializing the device!
[ 2101 ] Startup: Boot completed with ERRORS!
```

Figure 6.1: Firmware initialization on PuTTY.

Bibliography

- [1] SpaceLab. FloripaSat-2 Documentation, 2021. Available at <<https://github.com/spacelab-ufsc/floripasat2-doc>>.
- [2] SpaceLab. On-Board Data Handling, 2019. Available at <<https://github.com/floripasat/obdh>>.
- [3] SpaceLab. On-Board Data Handling 2.0, 2020. Available at <<https://github.com/spacelab-ufsc/obdh2>>.
- [4] Samtec. FSI-110-03-G-D-AD, 2020. Available at <<https://www.samtec.com/products/fsi-110-03-g-d-ad>>.
- [5] Texas Instruments Inc. *MSP430x5xx and MSP430x6xx Family User's Guide*, October 2016.
- [6] Texas Instruments Inc. *TPS328x Voltage Monitor With Watchdog Timer*, July 2020.
- [7] Amazon Web Services, Inc. FreeRTOS - Real-time operating system for microcontrollers, 2020. Available at <<https://www.freertos.org/>>.
- [8] Texas Instruments. MSP Driver Library, 2020. Available at <<https://www.ti.com/tool/MSPDRIVERLIB>>.
- [9] SpaceLab. On-board data handling 2.0 draftsman document, 2020. Available at <https://github.com/spacelab-ufsc/obdh2/tree/master/hardware/outputs/board_draftsman>.

APPENDIX A

Test Report of v0.5 Version

This appendix is a test report of the first manufactured and assembled PCB (version v0.5).

- **PCB manufacturer:** PCBWay (China)
- **PCB assembly:** PCBWay (China)
- **PCB arrival date:** 2021/04/14
- **Execution date:** 2021/04/16 to TBC
- **Tester:** G. M. Marcelino

A.1 Visual Inspection

- **Test description/Objective:** Inspection of the board, visually and with a multimeter, searching for fabrication and assembly failures.
- **Material:**
 - Multimeter UNI-T DT830B
- **Results:** The results of this test can be seen in Figures A.1 (top view of the board) and A.2 (bottom view of the board).
- **Conclusion:** No problems were identified on this test.

A.2 Firmware Programming

- **Test description/Objective:** Inspection of the board, visually and with a multimeter, searching for fabrication and assembly mistakes.
- **Material:**
 - Code Composer Studio v9.3.0
 - MSP-FET Flash Emulation Tool
 - USB-UART converter
 - Screen (Linux software)

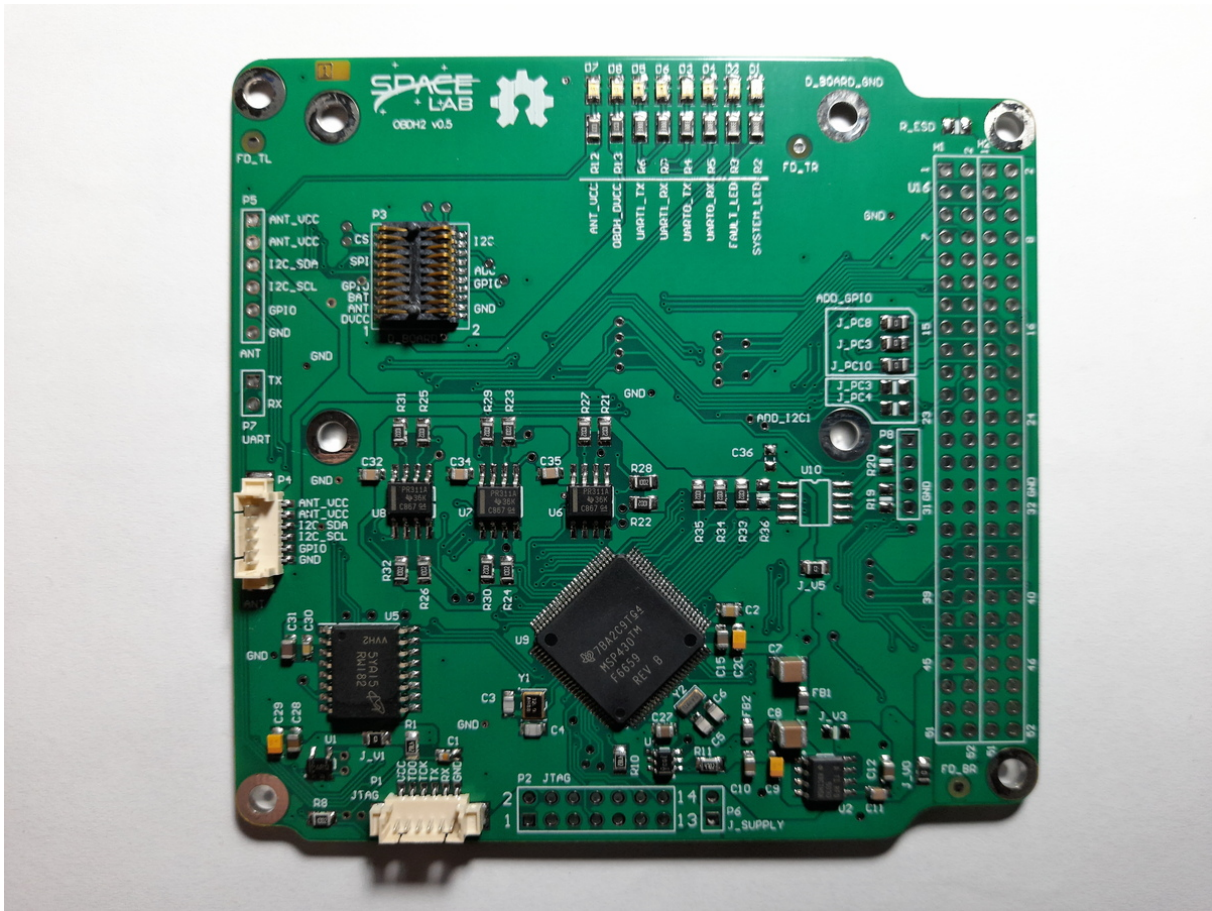


Figure A.1: Top view of the OBDH 2.0 v0.5 board.

- **Results:** The results of this are available in Figure A.3, where the log messages of the first boot of the board can be seen.
- **Conclusion:** No problems were identified on this test.

A.3 Communication Busses

- **Test description/Objective:** Test the communication busses of the board, as listed below:
 - I²C Port 0
 - I²C Port 1
 - I²C Port 2
- **Material:**
 - Saleae Logic Analyzer (24 MHz, 8 channels)
 - Saleae Logic software (v1.2.18)
 - MSP-FET Flash Emulation Tool

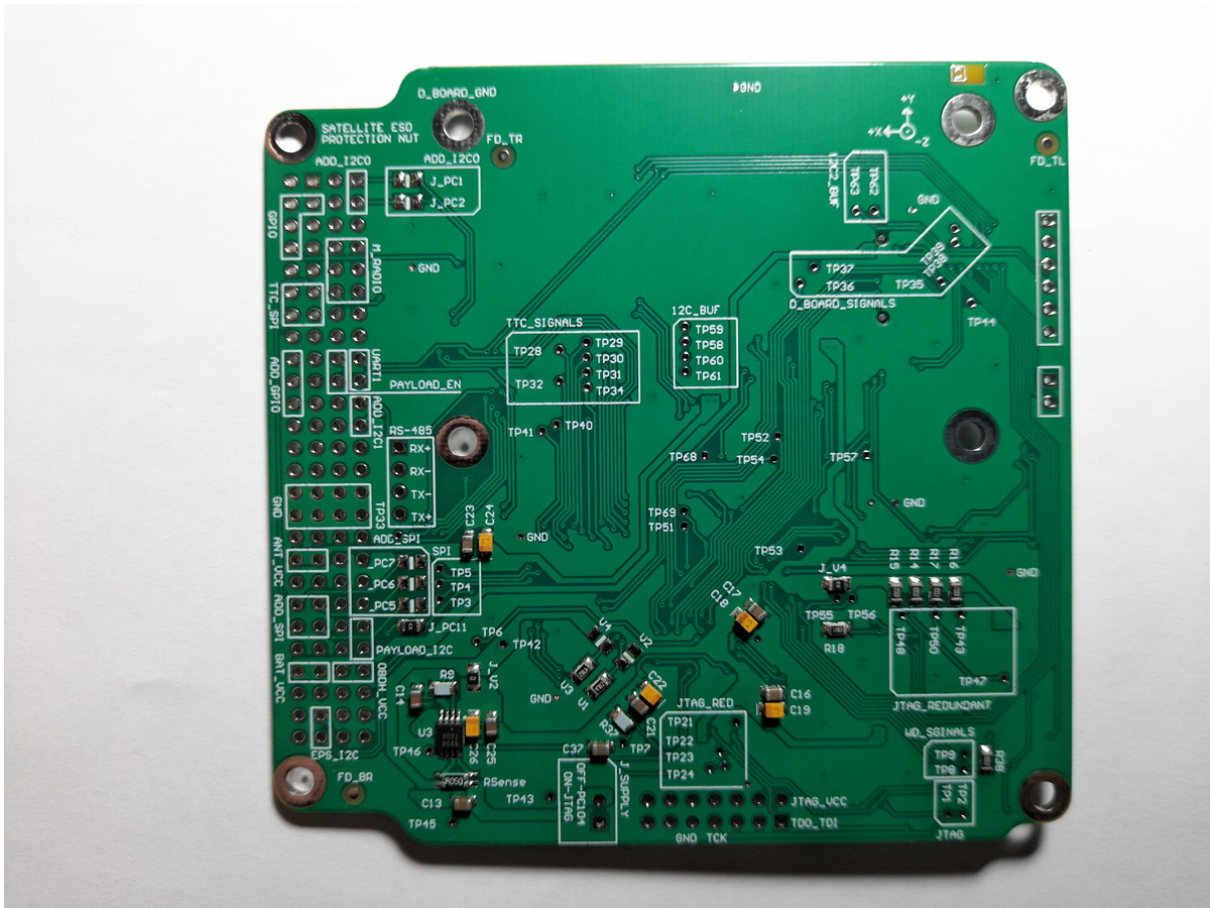


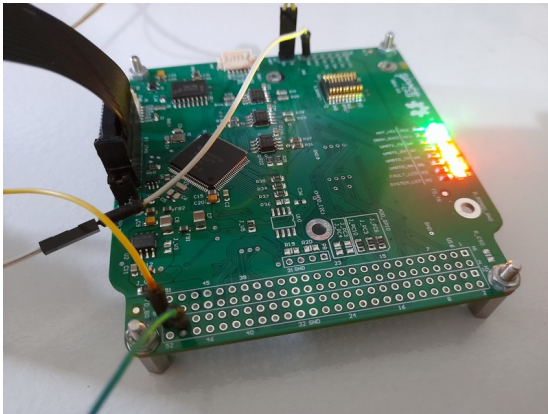
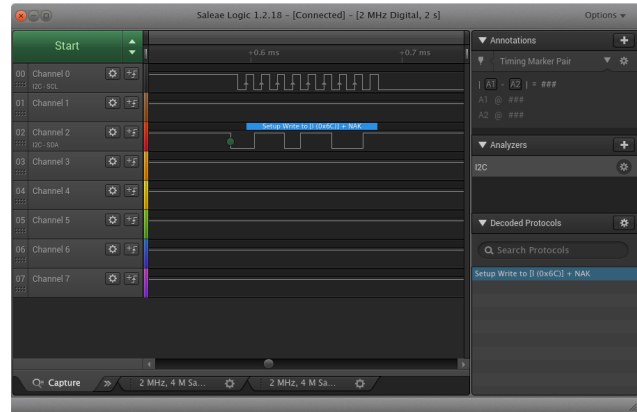
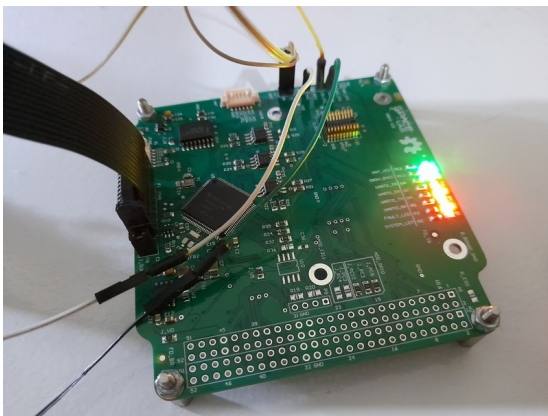
Figure A.2: Bottom view of the OBDH 2.0 v0.5 board.

- **Results:** The results of this test can be seen in Figures A.4, A.5 and A.6.
- **Conclusion:** No problems were identified on this test, all buses are working as expected.

A.4 Sensors

A.4.1 Input Voltage

- **Test description/Objective:** .
- **Material:**
 - Code Composer Studio v9.3.0
 - MSP-FET Flash Emulation Tool
 - USB-UART converter
 - Screen (Linux software)
- **Results:** .
- **Conclusion:** .

(a) Connections of the I²C port 1 test.(b) Waveforms of the I²C port 1 test.Figure A.5: I²C port 1 test.(a) Connections of the I²C port 2 test.(b) Waveforms of the I²C port 2 test.Figure A.6: I²C port 2 test.

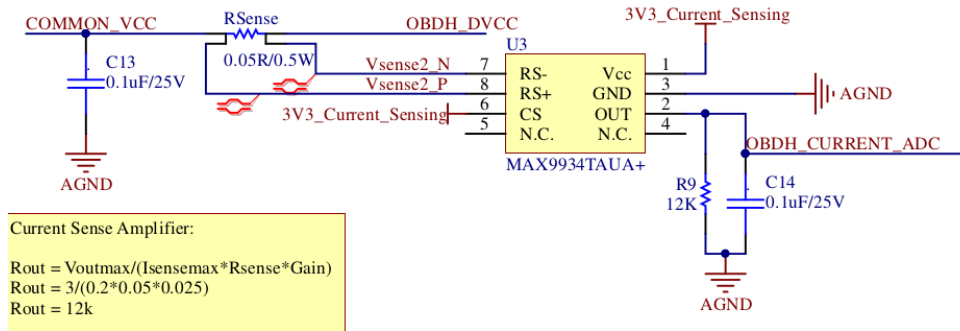
- Code Composer Studio v9.3.0
- MSP-FET Flash Emulation Tool
- USB-UART converter
- Screen (Linux software)

- Results: .
- Conclusion: .

A.5 Peripherals

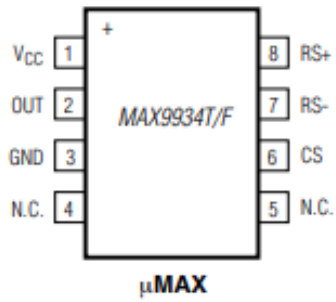
A.5.1 NOR Flash Memory

- **Test description/Objective:** Test the functionality of the NOR flash memory by verifying the device ID register of the IC.

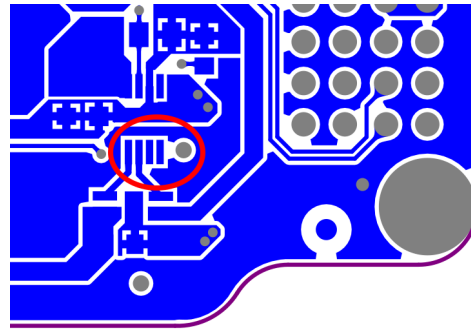


(a) Current sensing circuit.

TOP VIEW



(b) MAX9934 pinout.



(c) Current sensing layout (bottom layer).

Figure A.7: .

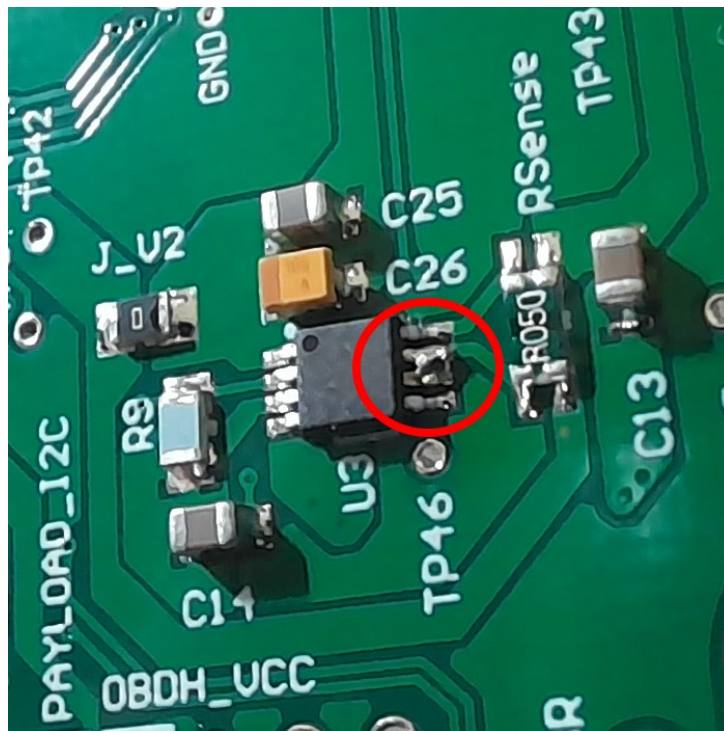


Figure A.8: Current sensor fix.

```

[screen 0: ttyUSB0]
[ 230 ] LEDs: Initializing system LEDs...
[ 237 ] Current Sensor: Initializing the current sensor...
[ 246 ] Time Control: Last saved system time (epoch): 0 sec
[ 256 ] Current Sensor: Raw input current: 101
[ 264 ] Current Sensor: Current input current: 35 mA
[ 273 ] Voltage Sensor: Initializing the voltage sensor...
[ 282 ] Voltage Sensor: Current input voltage: 3391 mV
[ 292 ] Temperature Sensor: Initializing the temperature sensor...
[ 301 ] Temperature Sensor: Current temperature: 44.32 oC
[ 311 ] EPS: Initializing EPS device...
[ 326 ] EPS: Error during the initialization! (error -1)
[ 335 ] Radio: Initializing radio device...
[ 452 ] Radio: Error reading the device ID!
[ 465 ] Startup: libcsf disabled!
[ 480 ] Payload EDC: Error initializing the device!
[ 489 ] Antenna: Initializing...
[ 503 ] Antenna: Error reading the antenna status!
[ 512 ] Startup: Boot completed with 4 ERROR(s)!
[ 520 ] Current Sensor: Raw input current: 99
[ 528 ] Current Sensor: Current input current: 35 mA
[ 1521 ] Radio: Transmitting 58 byte(s)...
[ 1629 ] Radio: Error transmitting a packet!
[ 1646 ] Radio: Error starting reception!
[ 1654 ] Uplink: Error during data reception! Trying again in 10000 ms...
[ 5519 ] Current Sensor: Raw input current: 94
[ 5528 ] Current Sensor: Current input current: 33 mA
[ 10519 ] Current Sensor: Raw input current: 82
[ 10528 ] Current Sensor: Current input current: 29 mA
[ 11765 ] Radio: Error starting reception!
[ 11773 ] Uplink: Error during data reception! Trying again in 10000 ms...
[ 15519 ] Current Sensor: Raw input current: 81
[ 15528 ] Current Sensor: Current input current: 28 mA
[ 20519 ] Current Sensor: Raw input current: 90
[ 20528 ] Current Sensor: Current input current: 31 mA
[ 21885 ] Radio: Error starting reception!
[ 21893 ] Uplink: Error during data reception! Trying again in 10000 ms...
[ 25519 ] Current Sensor: Raw input current: 85
[ 25528 ] Current Sensor: Current input current: 30 mA
[ 30519 ] Current Sensor: Raw input current: 93
[ 30528 ] Current Sensor: Current input current: 33 mA
[ 32005 ] Radio: Error starting reception!
[ 32013 ] Uplink: Error during data reception! Trying again in 10000 ms...
[ 35519 ] Current Sensor: Raw input current: 85
[ 35528 ] Current Sensor: Current input current: 30 mA
[ 40519 ] Current Sensor: Raw input current: 85
[ 40528 ] Current Sensor: Current input current: 30 mA
[ 42125 ] Radio: Error starting reception!
[ 42133 ] Uplink: Error during data reception! Trying again in 10000 ms...
[ 45519 ] Current Sensor: Raw input current: 93
[ 45528 ] Current Sensor: Current input current: 33 mA
[ 50519 ] Current Sensor: Raw input current: 82
[ 50528 ] Current Sensor: Current input current: 29 mA
[ 52245 ] Radio: Error starting reception!
[ 52253 ] Uplink: Error during data reception! Trying again in 10000 ms...

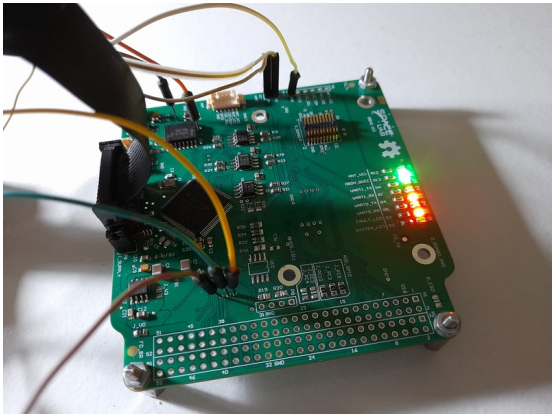
```

Figure A.9: Log messages with the read values from the current sensor.

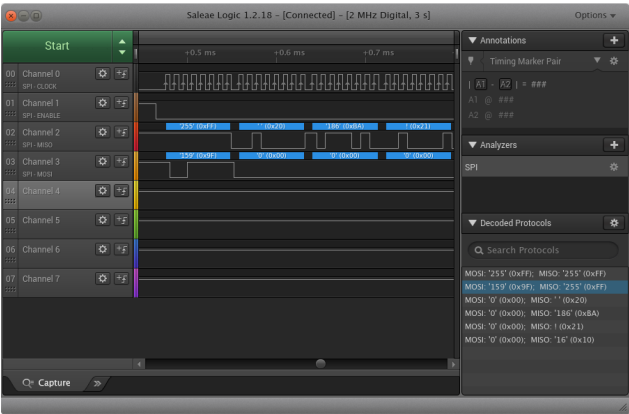
- **Material:**
 - Saleae Logic Analyzer (24 MHz, 8 channels)
 - Saleae Logic software (v1.2.18)
 - MSP-FET Flash Emulation Tool
- **Results:** The results of this test can be seen in Figure A.10.
- **Conclusion:** No problems were identified on this test, as can be seen in Figure A.10(b), the device ID register was read as expected.

A.6 Conclusion

Excluding the current sensor issue, no major problems were identified during the executed tests. For the next fabrication round, the identified mistakes will be corrected.



(a) Connections of the NOR flash memory test.



(b) Waveforms of the NOR memory SPI.

Figure A.10: NOR memory SPI test.

APPENDIX B

Test Report of v0.7 Version

This appendix is a test report of the first manufactured and assembled PCB (version v0.7).

- **PCB manufacturer:** PCBWay (China)
- **PCB assembly:** PCBWay (China)
- **PCB arrival date:** 2022/04/18
- **Execution date:** 2022/04/22 to 2022/08/10
- **Tester:** Gabriel M. Marcelino, Vitória B. Bianchin and Bruno Benedetti
- **DNP components:** P8, P2, P5, P6, P7, D1, D2, D3, D4, D5, D6, D7, D8, U10, R19, R20, R_ESD, J_PC3, J_PC4, R2, R3, R4, R5, R6, R7, R12, R13, J_V3, J_PC5, J_PC6, J_PC7, J_PC1, J_PC2, V1, V4, R36, C36

B.1 Visual Inspection

- **Test description/Objective:** Inspection of the board, visually and with a multimeter, searching for fabrication and assembly failures.
- **Material:**
 - Digital microscope (1000x)
 - Multimeter Fluke 17B+
- **Results:** The results of this test can be seen in Figures B.1 (top view of the board) and B.2 (bottom view of the board).
- **Conclusion:** No problems were identified on this test.

B.2 Firmware Programming

- **Test description/Objective:** Inspection of the board, visually and with a multimeter, searching for fabrication and assembly mistakes.
- **Material:**

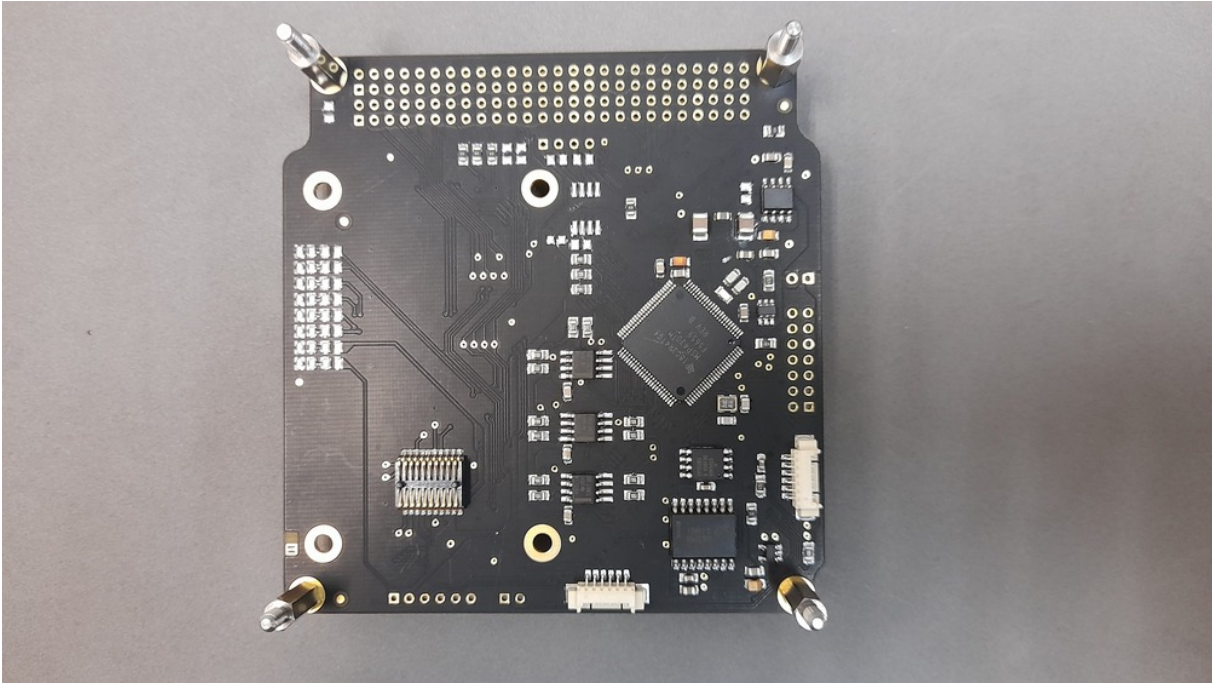


Figure B.1: Top view of the OBDH 2.0 v0.7 board.

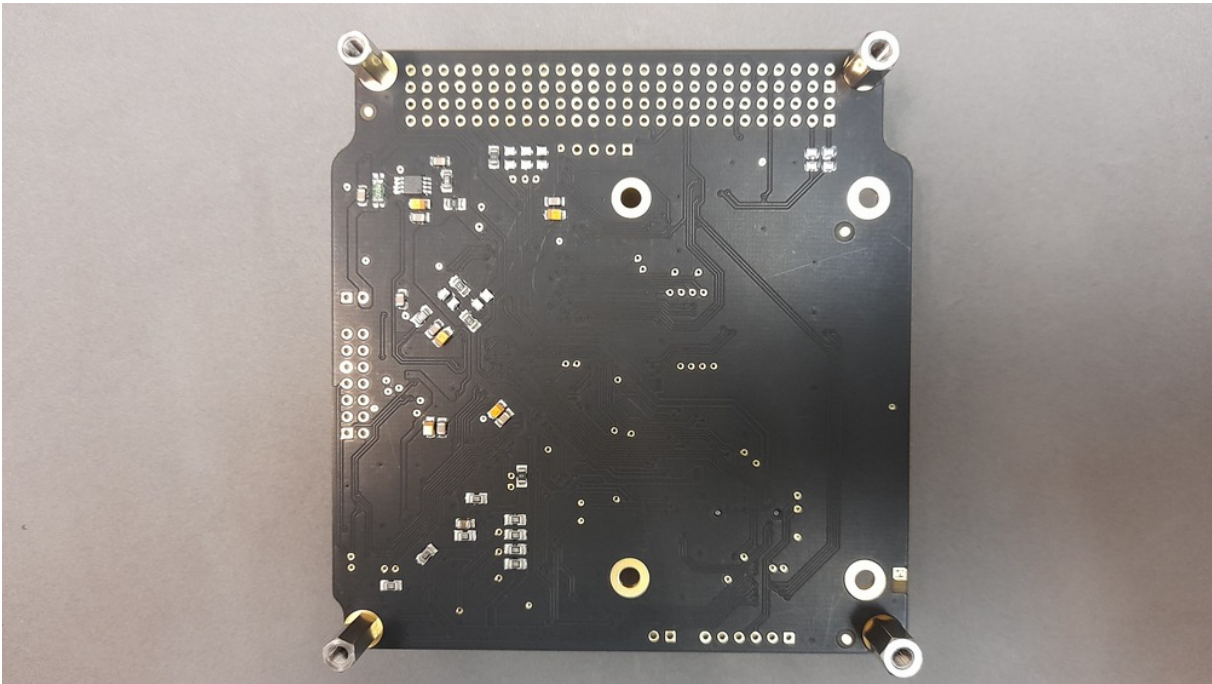


Figure B.2: Bottom view of the OBDH 2.0 v0.7 board.

- Code Composer Studio v11
- MSP-FET Flash Emulation Tool
- USB-UART converter
- PuTTY

- **Results:** The results of this are available in Figure B.3, where the log messages of the first boot of the board can be seen.
- **Conclusion:** No problems were identified on this test.

```

Source code: https://github.com/spacelab-ufsc/obdh2
Documentation: https://github.com/spacelab-ufsc/obdh2/doc

.....
SpaceLab
OB30420
.....

=====
Version:      v0.9.0
Status:       Development
Author:       Gabriel Mariano Marcelino <gabriel.mar@gmail.com>
Compiled:     Apr 29 2022 at 14:40:46
=====

200 | StartUp: FreeRTOS V10.2.0
206 | StartUp: Hardware revision is 1
214 | StartUp: System clocks: MCLK=31981568 Hz, SCLK=31981568 Hz, HCLK=32768 Hz
228 | StartUp: Last reset cause: 0x0
236 | Media: Initializing M0 memory...
258 | Media: MT250 with 512 Mb capacity detected!
266 | LEDs: Initializing system LEDs...
274 | Current Sensor: Initializing the current sensor...
283 | Current Sensor: Current input current: 158 mA
303 | Voltage Sensor: Initializing the voltage sensor...
312 | Voltage Sensor: Current input voltage: 3487 mV
322 | Temperature Sensor: Initializing the temperature sensor...
332 | Temperature Sensor: Current temperature: 24 oC
342 | EPS: Initializing EPS device...
487 | EPS: Error during the initialization! (error -1)
501 | StartUp: libexp disabled!
516 | Payload: EDC 1: Error during the initialization!
533 | Payload: EDC 0: Error during the initialization!
542 | Antenna: Initializing...
549 | Antenna: Error during the initialization!
558 | StartUp: Boot completed with 4 ERROR(S)!
569 | Time Control: Error reading the system time from the non-volatile memory!
587 | Time Control: The last saved system time is not available!
597 | TTC: Initializing TTC device 0...
605 | Payload: EDC 0: Error reading housekeeping data!
614 | EPS: Initializing EPS device...
623 | TTC: Error reading the hardware version of the TTC device 0!
642 | Read TTC: Error initializing the TTC device!
658 | TTC: Initializing TTC device 1...
677 | Antenna: Initializing...
694 | TTC: Error reading the hardware version of the TTC device 1!
713 | Read TTC: Error initializing the TTC device!
761 | TTC: Error reading the data from the TTC device 0!
771 | Read TTC: Error reading data from the TTC 0 device!
781 | Read Antenna: Error initializing the Antenna device!
807 | TTC: Error reading the data from the TTC device 1!
820 | Read TTC: Error reading data from the TTC 1 device!
830 | EPS: Error during the initialization! (error -1)
833 | Read EPS: Error initializing the EPS device!
848 | EPS: Error reading the data! No initialized driver!
857 | Read EPS: Error reading data from the EPS device!
937 | Antenna: Error reading the antenna data!
946 | Read Antenna: Error reading data from the Antenna device!

```

Figure B.3: Log messages during the first boot.

B.3 Communication Busses

- **Test description/Objective:** Test the communication busses of the board, as listed below:
 - I²C Port 0
 - I²C Port 1
 - I²C Port 2
- **Material:**
 - Saleae Logic Analyzer (24 MHz, 8 channels)
 - Saleae Logic software (v2)

– MSP-FET Flash Emulation Tool

- **Results:** The results of this test can be seen in Figures B.5, B.6 and B.7.
- **Conclusion:** No problems were identified on this test, all buses are working as expected.

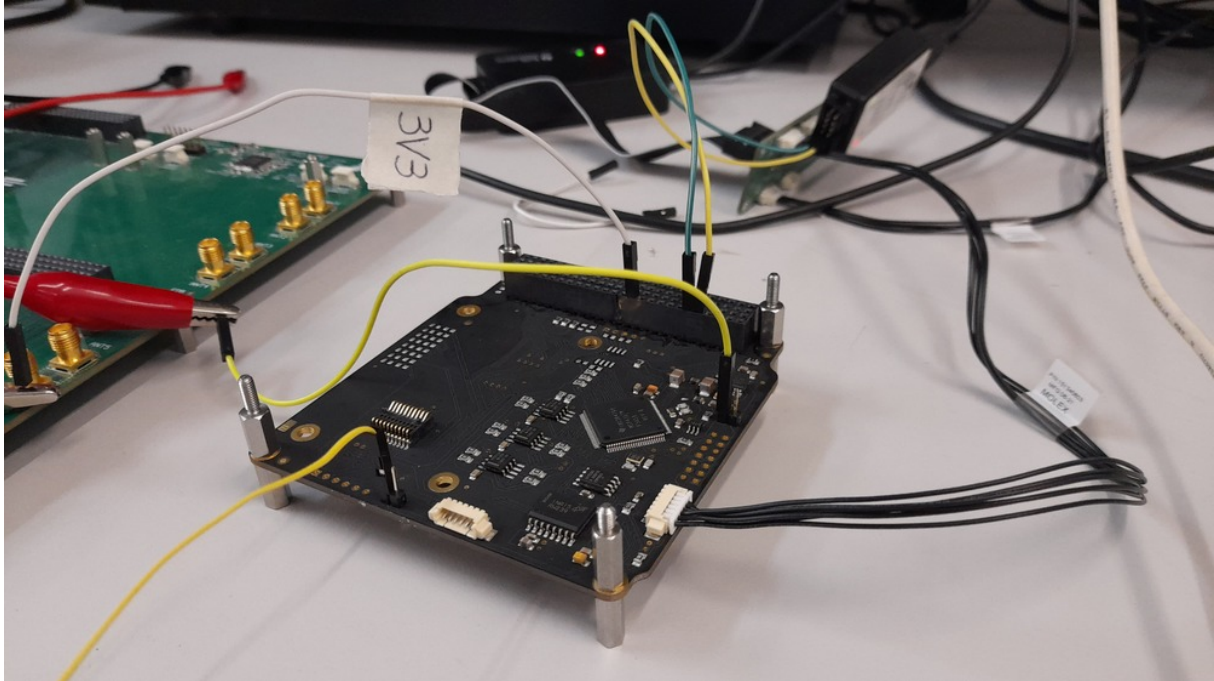


Figure B.4: Setup of the I2C port tests.

B.4 Sensors

B.4.1 Input Voltage

- **Test description/Objective:** Verify the input voltage measurements of the board.
- **Material:**
 - Code Composer Studio v11
 - MSP-FET Flash Emulation Tool
 - Programmable power supply
 - USB-UART converter
 - Screen (Linux software)
- **Results:** TBC.
- **Conclusion:** The input voltage was measured correctly by the sensor.

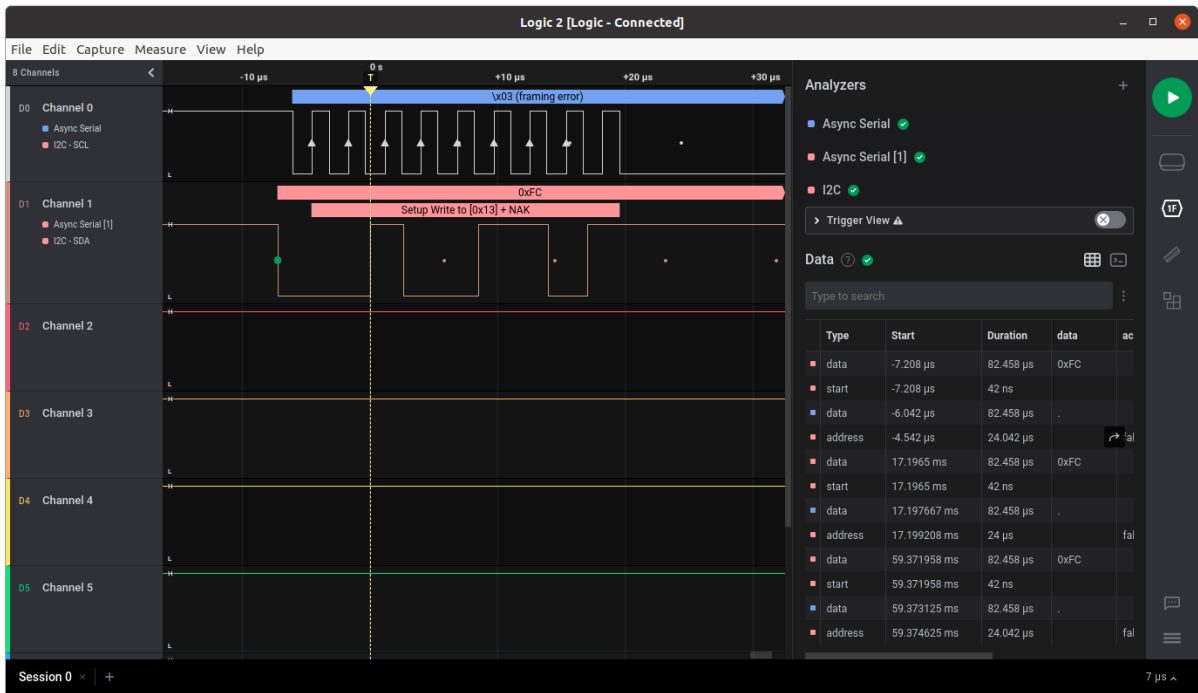


Figure B.5: Waveform of the I2C port 0.

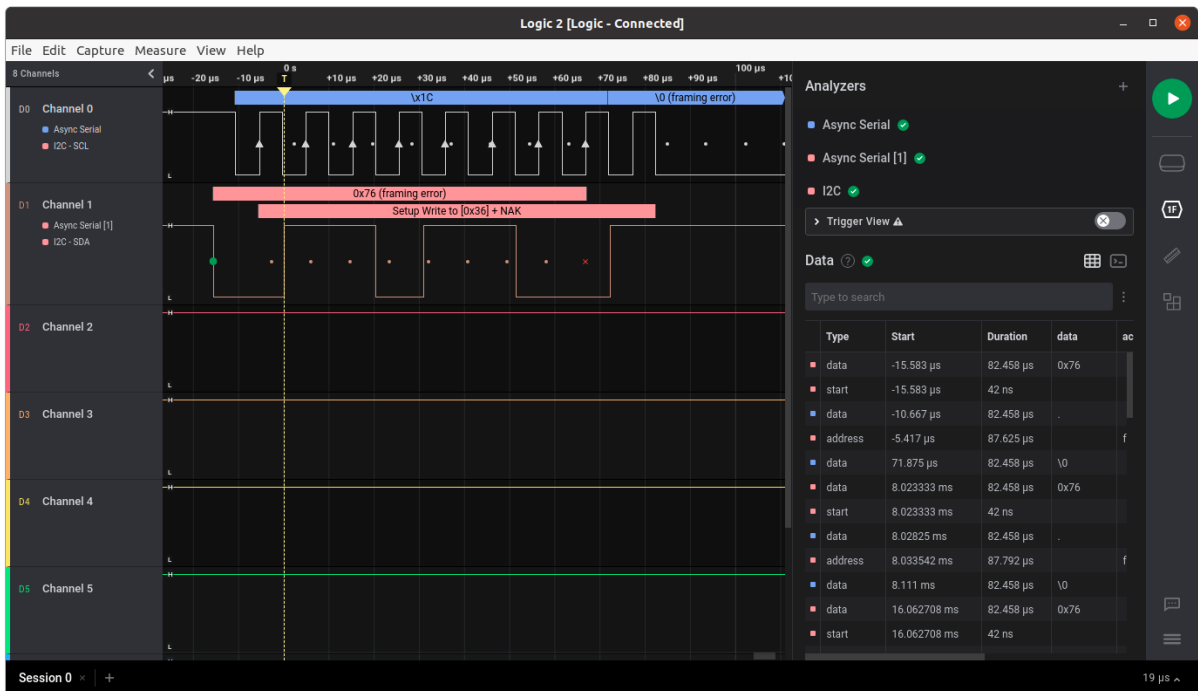


Figure B.6: Waveform of the I2C port 1.

B.4.2 Input Current

- Test description/Objective: Verify the input current measurements of the board.
- Material:

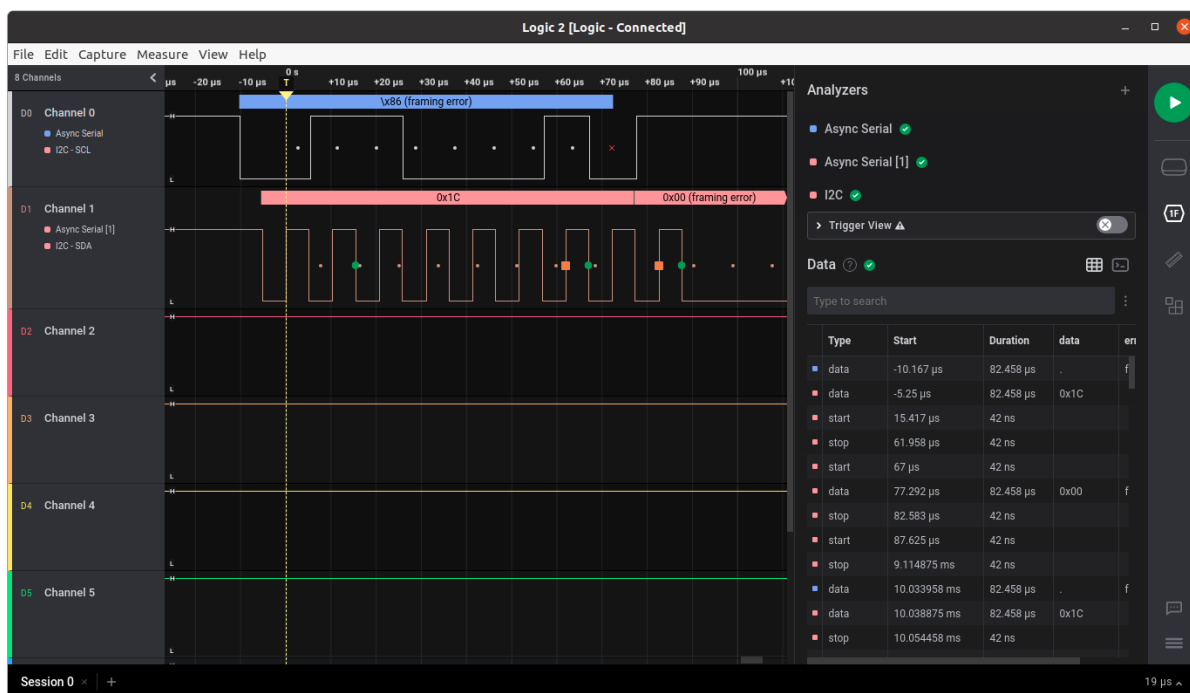


Figure B.7: Waveform of the I2C port 2.

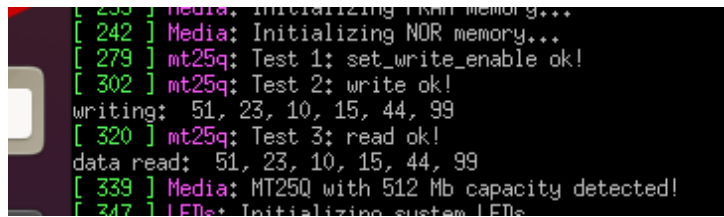
- Code Composer Studio v11
- MSP-FET Flash Emulation Tool
- Programmable power supply
- USB-UART converter
- Screen (Linux software)
- **Results:** TBC.
- **Conclusion:** The input current was measured correctly by the sensor.

B.5 Peripherals

B.5.1 NOR Flash Memory

- **Test description/Objective:** Test the functionality of the NOR flash memory by verifying the device ID register of the IC and performing writing/reading operations.
- **Material:**
 - Code Composer Studio v11
 - MSP-FET Flash Emulation Tool
 - USB-UART converter
 - Screen (Linux software)
- **Results:** The results of this test can be seen in Figure B.8.

- **Conclusion:** No problems were identified on this test, as can be seen in Figure B.8, an writing/reading operation were executed with success.

A terminal window showing the output of a test for NOR flash memory. The text is as follows:

```
[ 235 ] Media: Initializing FRAM memory...  
[ 242 ] Media: Initializing NOR memory...  
[ 279 ] mt25q: Test 1: set_write_enable ok!  
[ 302 ] mt25q: Test 2: write ok!  
writing: 51, 23, 10, 15, 44, 99  
[ 320 ] mt25q: Test 3: read ok!  
data read: 51, 23, 10, 15, 44, 99  
[ 339 ] Media: MT25Q with 512 Mb capacity detected!  
[ 347 ] LEDs: Initializing system LEDs
```

Figure B.8: Test results of the NOR flash memory.

B.5.2 FRAM Memory

- **Test description/Objective:** Test the functionality of the FRAM memory by verifying the device ID register of the IC and performing writing/reading operations.
- **Material:**
 - Code Composer Studio v11
 - MSP-FET Flash Emulation Tool
 - USB-UART converter
 - Screen (Linux software)
- **Results:** The results of this test can be seen in Figure B.9.
- **Conclusion:** No problems were identified on this test.

B.6 Conclusion

No major problems were identified during the executed tests, all peripherals all working as expected.

