ENDUROSAT

USER MANUAL

S-Band Transmitter

Last Revision: 15/10/2020

ENDUROSAT

CONTENTS

ENDUROSAT

ACRONYM LIST

BUC	Block Up Convertor
CCSDS	Consultative Committee for Space Data Systems
DVB-S2	Digital Video Broadcasting – Second Generation
EPS	Electrical Power System
ESUP	EnduroSat User Protocol
ETSI	European Telecommunications Standards Institute
FW	Firmware
GND	Ground
GS	Ground Station
LVDS	Low Voltage Differential Signaling
MPEG	Moving Picture Experts Group
OBC	On Board Computer
OSI	Open Systems Interconnection
RF	Radio Frequency
SMA	Sub-Miniature version A
TS	Transport Stream
UART	Universal Asynchronous Receiver/Transmitter

ENDUROSAT

1 INTRODUCTION

1.1 Overview

This User Manual describes the relationship between the HOST (OBC, Payload) and the S-Band Transmitter on one hand and between the S-Band Transmitter and the S-Band Antenna on the other hand. It specifies the interface requirements that the participating subsystems must meet so that a successful communication is achieved. This User Manual points out the concept of operation of the interfaces, defines the message structure and protocols that govern the interchange of data and commands. The interfaces are discussed in the context of the OSI model layers.

A typical application of the EnduroSat's S-Band Transmitter is shown on Figure 1. Two separate connectivity sides can be distinguished – HOST (OBC) side and Antenna side. Hence, two separate interfaces – HOST (OBC) interface and Downlink interface.

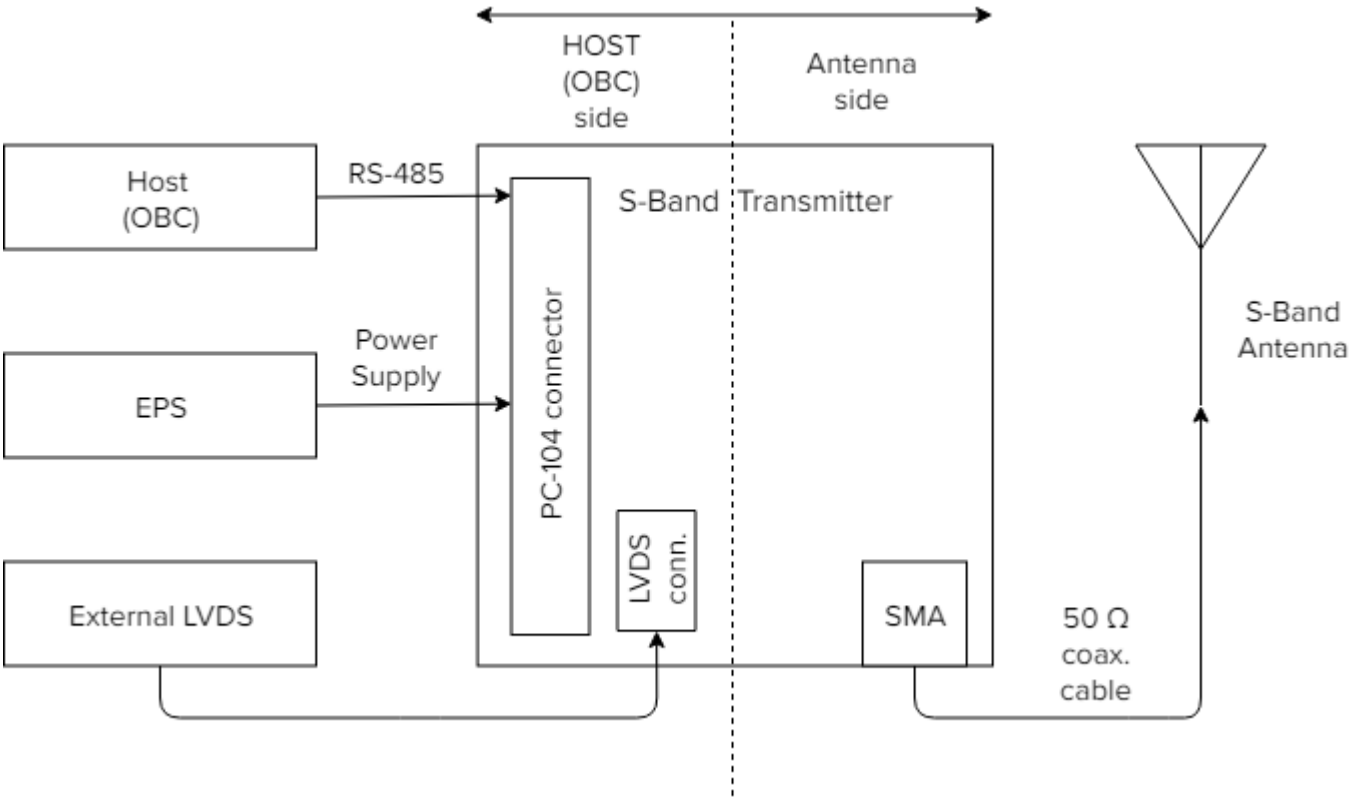

Figure 1: Endurosat's S-Band Transmitter Typical Application

The S-Band Transmitter is built around the PC-104 connector which is the most common for CubeSat systems. Although two different interfaces are available on this connector (UART, RS-485), only one of them is operational depending on the internal configuration of the module. The default interface is RS-485.

Also, this module is configured with DVB-S2 (ETSI EN 302 307) for the standard configuration. That is why this User Manual will refer only to it.

The HOST (OBC) interface connects the HOST (OBC or Payload) to the S-Band Transmitter. The physical connection is accomplished through the PC-104 connector. Figure 2 shows the position and naming of the PC-104 container connector and also the pin numbering. Figure 3 shows all the signals allocated on that connector, whereas Table 1 and Table 2 give further details for each of the used pins.

The physical interface used in the standard configuration of the EnduroSat's S-Band Transmitter module is the RS-485 allocated on pins 37 and 38 of the H1 header. This is a standard RS-485 (TIA485) driver/receiver operating in half-duplex mode. The default UART settings are as follows:

- Baud Rate: 3 Mbps
- Data: 8 bits
- Parity: None
- Stop bits: 1 bit

The RS-485 interface of the S-Band Transmitter can be either terminated or not with a 120 Ω resistor depending on client's preferences.

The module can also be delivered with either 12V or 5V power supply according to client's needs. Please, specify the desired option before the order.

Figure 2: Container Connectors

ENDUROSAT

1.2 Pinout Diagrams

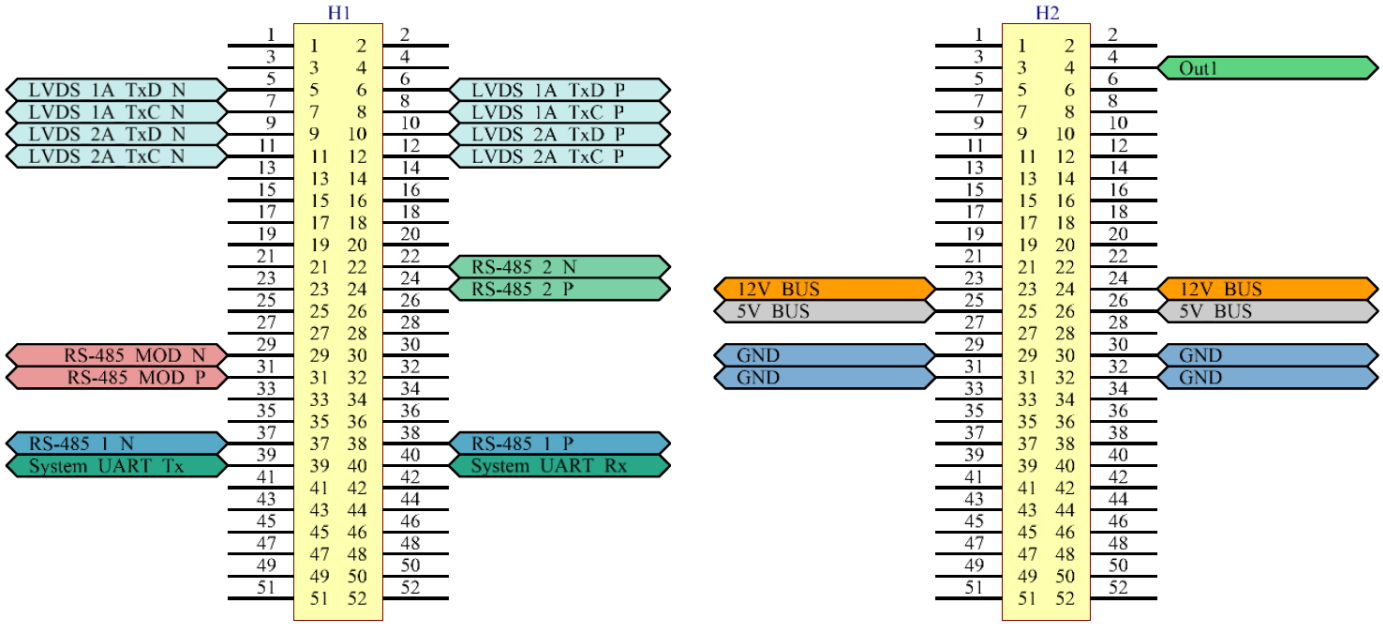

Figure 3: Pinout of the Stack Connectors

Table 1: H1 Container Connector Pinout

Pin	Mnemonic	Description
H1-5	LVDS_1A_TxD_N	LVDS Channel 1A Data Negative
H1-6	LVDS_1A_TxD_P	LVDS Channel 1A Data Positive
H1-7	LVDS_1A_TxC_N	LVDS Channel 1A Clock Negative
H1-8	LVDS_1A_TxC_P	LVDS Channel 1A Clock Positive
H1-9	LVDS_2A_TxD_N	LVDS Channel 2A Data Negative
H1-10	LVDS_2A_TxD_P	LVDS Channel 2A Data Positive
H1-11	LVDS_2A_TxC_N	LVDS Channel 2A Clock Negative
H1-12	LVDS_2A_TxC_P	LVDS Channel 2A Clock Positive
H1-22	RS-485_2_N	Secondary RS-485 Negative
H1-24	RS-485_2_P	Secondary RS-485 Positive
H1-29	RS-485 MOD N	RS-485 Modulator Negative
H1-31	RS-485 MOD P	RS-485 Modulator Positive
H1-37	RS-485_1_N	Primary RS-485 Negative
H1-38	RS-485_1_P	Primary RS-485 Positive
H1-39	System_UART_Tx	UART Transmit Data for System Usage
H1-40	System_UART_Rx	UART Receive Data for System Usage

ENDUROSAT

Table 2: H2 Container Connector Pinout

Pin	Mnemonic	Description
H2-4	Out1	S-Band Transmitter Enable
H2-23	12V_BUS	+12V BUS Power Supply
H2-24	12V_BUS	+12V BUS Power Supply
H2-25	5V_BUS	+ 5V BUS Power Supply
H2-26	5V_BUS	+ 5V BUS Power Supply
H2-29	GND	Ground
H2-30	GND	Ground
H2-31	GND	Ground
H2-32	GND	Ground

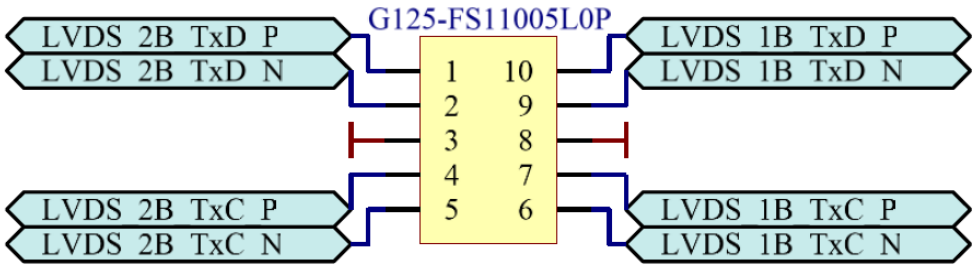

Figure 4: Pinout of the LVDS Connector

Table 3: LVDS Container Connector Pinout

Pin	Mnemonic	Description
G125-FS11005L0P-1	LVDS_2B_TxD_P	LVDS Channel 2B Data Positive
G125-FS11005L0P-2	LVDS_2B_TxD_N	LVDS Channel 2B Data Negative
G125-FS11005L0P-3	GND	Ground
G125-FS11005L0P-4	LVDS_2B_TxC_P	LVDS Channel 2B Clock Positive
G125-FS11005L0P-5	LVDS_2B_TxC_N	LVDS Channel 2B Clock Negative
G125-FS11005L0P-6	LVDS_1B_TxC_N	LVDS Channel 1B Clock Negative
G125-FS11005L0P-7	LVDS_1B_TxC_P	LVDS Channel 1B Clock Positive
G125-FS11005L0P-8	GND	Ground
G125-FS11005L0P-9	LVDS_1B_TxD_N	LVDS Channel 1B Data Negative
G125-FS11005L0P-10	LVDS_1B_TxD_P	LVDS Channel 1B Data Positive

1.3 Recommended Operating Conditions

Table 4 contains information for the recommended operating conditions. The product is intended for proper operation at the conditions listed below.

Table 4: Recommended Operating Conditions

Parameter	Unit	Min	Typ	Max	Comment
12V_BUS	V	9	12	15	
5V_BUS	V	4.8	5	5.2	
Out1 High Level	V	2	3.3	5	Enable - leave open or pull up
Out1 Low Level	V	0	0	0.8	Disable - pull down
RS-485 Receiver Terminating Resistor	Ω		120		Can be switched on/off
System UART Tx High Level	V		3.3		
System UART Tx Low Level	V		0		
System UART Rx High Level	V	2	3.3	3.6	
System UART Rx Low Level	V	0		0.9	
Operating Temperature Range	°C	-30		+70	

1.4 Absolute Maximum Ratings

Table 5 contains information for the Absolute Maximum Ratings. The product is not intended for proper operation at any conditions except for the ones listed in 1.3 Recommended Operating Conditions. Stresses at or above the ones listed below may lead to defects and operational malfunctions.

Table 5: Absolute Maximum Ratings

Parameter	Unit	Min	Max	Comment
12V BUS Voltage	V	8	16	
5V_BUS	V	4.7	5.3	
Out1 Voltage	V	-0.3	5.5	
RS-485 Nodes Voltage	V	-10	6	
UART Voltage	V	-0.5	4.6	
LVDS Nodes Input Voltage	V	-0.3	3.6	
Storage Temperature Range	°C	-40	+80	

2 OPERATION SCENARIOS

The module can operate in two modes:

- Idle Mode;
- Transmit Mode.

The module has an SD card with integrated file system (FAT32) which allows external management from a host system through commands. The commands are the following: Open_File, Create_File, Read_File, Write_File, Delete_File, Delete_All_Files, DIR, DIR_Next, Send_File_s. The commands are applicable in both operation modes. However, bear in mind that **only one file command can be executed at a time.**

The module contains a set of RF parameters for RF part tuning. All the RF parameters can be configured or altered in both operation modes. Moreover, the module has two sets of status data (full and short) which are also accessible through commands.

The module has an internal logical container intended for pseudo-simultaneous execution of three types of commands: a file command, a status command, a configuration command.

The following scenarios are possible **at a time**:

- one file command and one/two configuration command(s);
- one file command and one/two status command(s);
- one file command, one configuration command and one status command;
- one/two configuration command(s);
- one/two status command(s);
- one configuration command and one status command;
- one single command of the three (file or status or configuration).

The diagram in Figure 5 illustrates the Logical Container and the three types of commands.

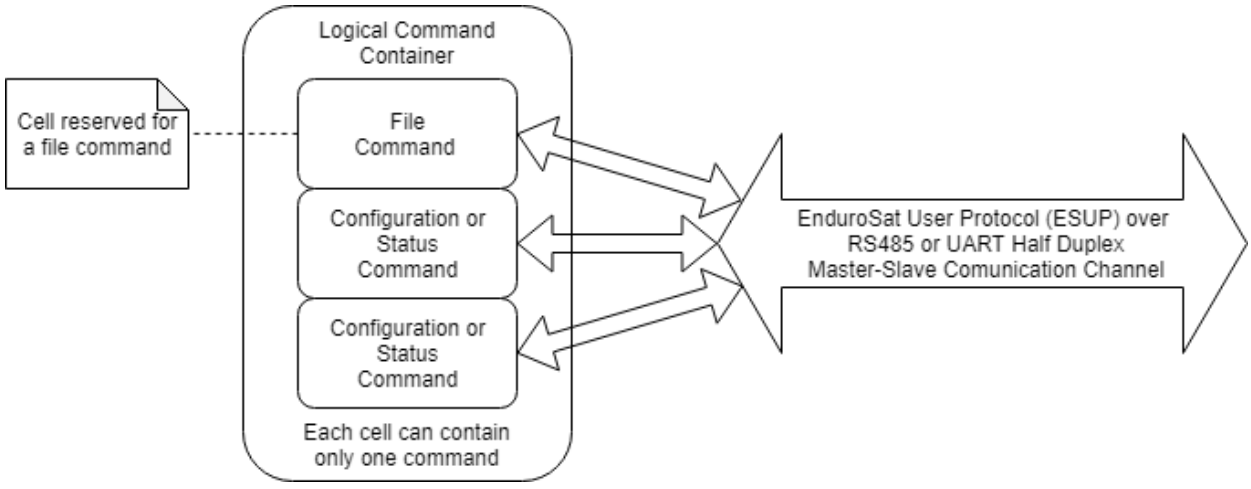

Figure 5: Logical Command Container

ENDUROSAT

Summary:

1. Each cell can contain only one command.
2. The first cell is reserved for a file command, while the second and the third cell can contain either a configuration or a status command.
3. Only one file command can be executed at a time.

2.1 Master-Slave Communication Channel Description

The module logic diagram is represented on the scheme below (Figure 6).

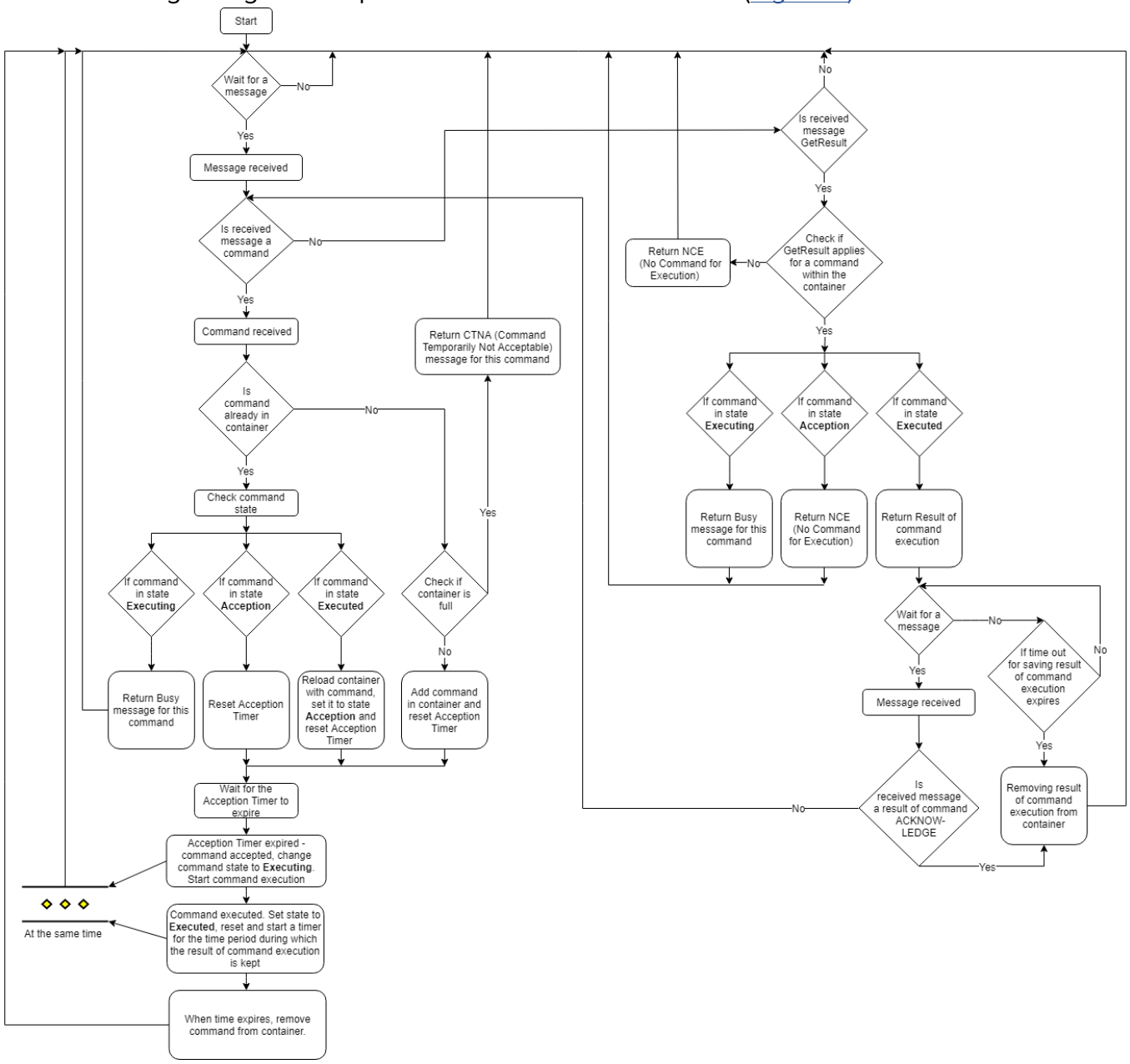

Figure 6: Protocol Processing Logic Diagram

Due to the specifics of the master-slave communication, the host system is responsible for the management of messages sent to the S-Band module. The communication is organized as a process of sending commands for execution and receiving the results of their execution. The sending of commands and the request for the result of their execution is initiated by the host system. The description below gives in detail one complete cycle of sending a command and getting the result of its execution. This is accomplished by triple pseudo-parallelism in order to utilize the entire capacity of the module.

The host sends a command message to the module and then waits for a response from the module maximum 1 ms. The response contains information whether the command has been accepted or if an error has occured. The response from the module is associated only with the received command. The module can respond in several different ways upon receiving a command:

- The module does not respond - because it has not received the message or the message has been damaged due to bad communication;
- The module responds with NOT ACKNOWLEDGE - the reason for this is most often wrong parameters in the message;
- The module responds with BUSY - this means that this command is currently being executed by the module;
- The module responds with COMMAND TEMPORARILY NOT ACCEPTABLE - which means that the module command container is currently full;
- The module responds with ACKNOWLEDGE - this means that the command has been received correctly.

When the module responds only with ACKNOWLEDGE, it starts a timer with duration of 2 ms. During this period of time it anticipates to receive the same command again. If the module receives the same command indeed, it responds once again with an ACKNOWLEDGE message and restarts the corresponding timer. If it does not receive the same command within the aforementioned 2 ms period, it assumes that the the ACKNOWLEDGE message has been correctly received by the host and starts executing the command. Having executed the command, the module stores the result of its execution from 3 to 5 sec. Depending on the type of command the time for execution varies from 100 μs to 40 s.

After the period of 2 ms from returning the ACK message to the host expires, the host may intermittently (at intervals between 10 and 100 ms or more) initiate requesting the result from command execution by Get Result Message.

Upon sending a GetResult message, the module responds in one of the following manners:

- The module does not respond - because it has not received the message or the message has been damaged due to bad communication;
- The module responds with NO COMMAND FOR EXECUTION – when the container does not contain a command for which a result is requested. This usually happens when the time

for storing the result of the command execution has expired and the command has been automatically deleted from the container or when the message contains incorrectly set parameters. This also happens when requesting a result for execution of a command being at acception state;

- The module responds with BUSY – when requesting a result for execution of a command being executed;
- The module responds with the result of command execution.

Having returned a message with the result of command execution, the module anticipates to receive an ACK message from the host as a confirmation for the received result of command execution. If the module receives such a message, it deletes the result of command execution from the container. Moreover, once the module has returned the result of command execution to the host, it may not receive an ACK message but a direct command instead. This happens in case of a file command or if the container is currently full. If the module receives a subsequent command, it deletes the current result from command execution and proceeds with executing the next command in the same manner. If neither an ACK message nor a new command has been sent by the time the storage period for the result of command execution expires, the module deletes the result from the container.

To ensure optimal operation of the module when working simultaneously with two or more commands in the container the pattern described below should be observed. As the communication is half duplex, to prevent collisions it is necessary that a waiting period of at least 5 ms follows every command sent by the host or GetResult message before sending a new command or GetResult message in case no response by the host is received. The sequence of filling the container with commands is determined by the host system based on the duration of commands execution in the module and the corresponding logical planning.

Example: Upon sending a command for transferring a file via the RF channel, the time period for execution is relatively indefinite as it depends on the data transfer rate and the file size. That is why it is appropriate to send this command first and to request the result of its execution at relatively long intervals (0.5 – 1 s). Meantime, it is appropriate to send commands registering the status of the module as they are executed immediately and their result is ready literally in a few seconds.

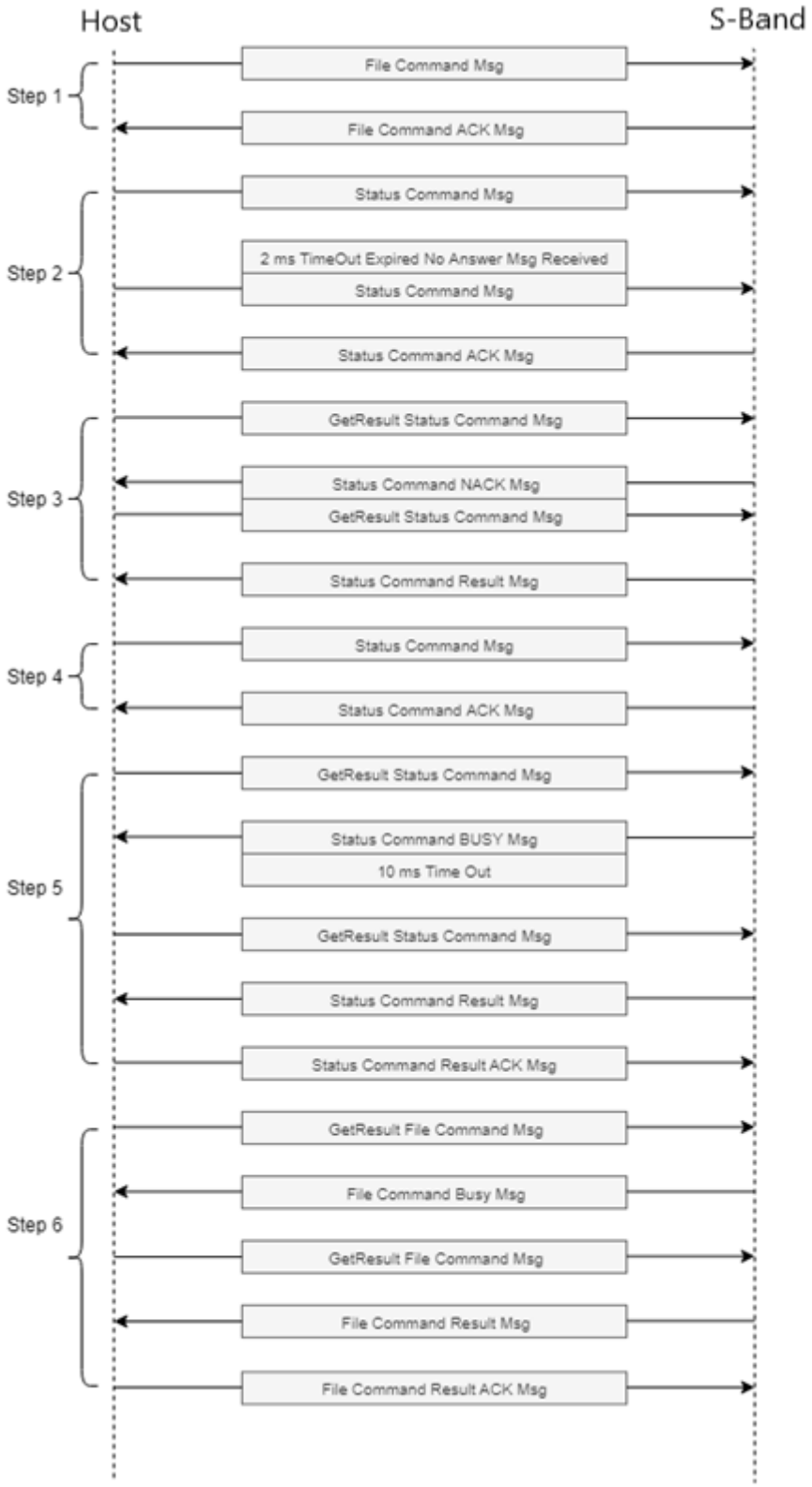

Figure 7: Example Diagram of Container Usage

2.2 System Description

The module is designed to transmit user data via its radio channels in compliance with the DVB-S2 standard. User data is derived from two sources: pre-recorded files from the SD card or directly submitted data via the LVDS interface. Data is transmitted through the radio channels by a set of modulations (QPSK, 8-PSK, 16-APSK) and a set of frequencies (2.0 - 2.5 GHz).

The module is composed of a radio part and a digital part. The radio part consists of a power amplifier and a software-defined digital radio (SDR). The digital part consists of a microcontroller and external memory - SD Card. SDR is composed of an FPGA modulator and a frequency synthesizer. The microcontroller configures, manages and monitors the operation and status of all the modules in the device. That is why it is necessary that data and parameters exchange takes place within the user interface.

The parameters are related to setting the digital radio. The data that is received back shows the state of the system. Also, exchange of user data files within the same interface occurs. The parameters to set the digital radio are as follows:

- Symbol Rate;
- Modulation Code;
- Central Frequency;
- TX Power;
- Roll Off;
- Pilot Signal;
- FEC Frame Size;
- Pretransmission Stuffing Delay.

Each of these parameters can be set and read separately with a corresponding command. All the parameters can be set and read as a whole with a corresponding command. Setting these parameters is possible in both operation modes of the module.

The status parameters of the module can only be read from the module during both operation modes. These parameters show the current mode (Idle or Transmit) of the module, faults information, module temperature and firmware version. The messages for module status are short and extended.

Working with files via the communication interface is very close to the standard way of working with a file system. The available commands are intended for creating a file, opening a file, writing in a file, reading from a file, and retrieving information for a list of existing files.

Working with file commands is possible in both operation modes of the module, except for the case when a file is sent by the module via the RF channel. When a file is being sent via the RF channel, other file commands cannot be processed until the sending process is accomplished.

The modes of the module are managed with the following service commands:

- Switch to Idle Mode;
- Switch to Transmit Mode;
- Firmware Update;
- Safe Shut Down;
- Change Baud Rate of RS-485 or UART interface.

After Power Up the module always starts in Idle Mode. In this mode, power consumption is minimal as the RF part is completely turned off and the only working component is the microcontroller. It maintains the communication with the outer world by receiving parameters, exchanging file information and saving energy. When switching from Transmit Mode to Idle Mode, the module turns off the RF systems. **If information is being transmitted by the SD card or via the LVDS channel at this time, the transmission is interrupted immediately.**

The LVDS interface works according to the EN 50083-9: 2002 standard and maintains data rates up to 130 Mbps. The inteface comprises two LVDS pairs: DATA and CLOCK. For detailed description, please refer to Chapter 5 LVDS Interface.

When switching to Transmit Mode, the entire RF system is switched on with its default or preconfigured parameters. In this mode, the information is transmitted via the RF channel by the SD card or LVDS interface, depending on the selected transmission interface applied for the configuration of the module beforehand. **Switching between interfaces is allowed if no file is being sent by the SD card at this moment. In case the RF configuration parameters are changed while sending a file by the SD card, the module automatically stops the data transmission, changes the RF settings and resumes the data transmission from the interrupting point. When data transmission via the LVDS interface takes place, it is the host system that is responsible for stopping the data transmission via the LVDS interface, changing the RF settings and then resuming the data transmission via the LVDS interface. The reason for this is the following: when changing the RF settings, the signal is temporarily interrupted and its quality is diminished until the new settings are established.**

The system automatically monitors the temperature. When the temperature reaches 70°C**, the system automatically switches to Idle Mode to keep the module from overheating. Transmit Mode can be resumed only when the system cools down by at least 15**°C **- 70**°C **i.e. temperature hysteresis has occurred.**

When the module is in Transmit Mode and no information is being transmitted, the module constantly broadcasts DVBS-2 zero frames in order to keep the opposite modem in lock state. A firmware update of the module is performed by a command provided that a file with the firmware update is available on the SD card. Upon initiating a command for firmware update, the execution of all actions currently running within the module is stopped, the RF part is turned off, an update is carried out and the module is automatically restarted.

In order to save the information in the file system and the internal service information, it is recommended that a Safe ShutDown command is used before turning off the electrical power supply of the module. The command for changing the speed of the RS-485 / UART interface has the following feature: the result of the operation works at the newly set speed. In other words, the command

itself and the ACKNOWLEDGE message work at the current speed, while GetResult and the result work at the new speed.

2.3 Characteristics of Working with File Commands

When creating a new file or opening an already existing one, the corresponding commands return a handle (an integer digit) to the host system. This handle is later used with the read and write commands, as the maximum information in a message is 1472 bytes and these commands are used multiple times until working with the corresponding file is completed. If commands other than the read and write commands are used after creating or opening a file, the corresponding file is automatically closed and the handle is reset. The handle is also reset after the file has been read or written and automatically closed.

When using a command for retrieving a list of files, if the number of these files is greater than the number of files that can be transferred by the list command (DIR), the next command to be used in order to obtain the missing part of the list is DIR_Next. It is used until the corresponding flag as a result of the command is active or until the host system is willing to read (see 3.1.3.1 DIR Command and 3.1.3.2 DIR Next Command). The Delete_File command deletes only the file specified in the command. The Delete_All_Files command deletes all the files except for the Error_Log.txt and Index_Log.txt service files.

3 THE ENDUROSAT USER PROTOCOL (ESUP)

3.1 Packet Structure

The EnduroSat User Protocol (ESUP) uses a communication packet for exchanging data and commands among the modules within the satellite. ESUP's length is between 18 and 1490 bytes and consists of eight functional fields. A size difference between the minimum and the maximum length of the ESUP packet exists because the number of bytes in the data field is a variable. Message size varies in the aforementioned range because data size is a variable. Data size depends on the type of message. Data field is in the interval from 0 to 1500 bytes, hence the efficiency of data transmission is optimized.

An important detail concerning physical communication is that the overall length of all the messages sent to the module must be a multiple of 16. The module also responds with messages multiple of 16. This is achieved by adding 1 to 15 zeros after the last byte of the CRC if necessary, so that the total length of the message is multiple of 16. An example is given in Table 8.

Table 6: Structure of the ESUP packet – Functional Fields

ESUP Packet - Functional Fields							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
4 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	0 - 1472 bytes	4 bytes

All data in the ESUP packet is stored in little-endian format.

Table 7: Description of the functional fields within the ESUP packet

Header	Send "ESUP" in ASCII character format 0x50555345
Module ID	Unique identification number for the S-Band Transmitter module. This number is sticked on the module or is additionally specified.
Data Length	Length of the data in bytes functional field
Command Status	Status of the current command: ESAP_ACK (0x005), ESAP_NOT ACK (0x006), ESAP_BUSY (0x007), ESAP_NCE (0x008), ESAP_CMD_STACK_FULL (0x009), ESAP_CMD_TEMPORARILY_NOT_ACCEPTED (0x0010) or 0x000 when the field is insignificant
Command	Command to be executed. See Command List (Table 9)
Type	Commands may or may not have Type associated with them
Data	Commands may or may not have Data associated with them
CRC32	The CRC calculating from the first byte of header to the last byte of data. The algorithm of calculating the CRC is the following: http://www.opensource.apple.com/source/xnu/xnu1456.1.26/bsd/libkern/crc32.c First, the polynomial itself and its table of feedback terms. The polynomial is: X^32+X^26+X^23+X^22+X^16+X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X^1+X^0

Table 8: Example for making a message multiple of 16

ESUP packet								
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32	Pading zeros
0x45535550	0xXXXX	0x0000	0x0005	0x0101	0x0040	-	0xXXX XXXXX	14 pading zeros to make a message multiple of 16

3.1.1 Command Types and Description

Table 9: Command List

Class of Command	Command		Type		Data
	Name	Code (Hex)	Type Name	Type Code (Hex)	Data Limits
Configuration	GET	0x0100	Symbol Rate	0x0040	1, 2, 3, 4, 5 Msp/s (1-byte unsigned char value)
			Transmit Power	0x0041	27-33 dBm, step 1 dB (1-byte unsigned char value)
			Center Frequency	0x0042	2000.000 - 2500.000 MHz step 1 kHz (4-byte float value)
			MODCOD	0x0043	1-23 (See Table 10) (1-byte unsigned char value)
			Roll-Off	0x0044	0.35 / 0.25 / 0.2 (0 / 1 / 2: 1-byte unsigned char value)
			Pilot Signal	0x0045	On / Off (1 / 0) (1-byte unsigned char value)
			FEC Frame Size	0x0046	Short / Normal (1 / 0) (1-byte unsigned char value)
			Pretransmission Stuff Delay	0x0047	2000 – 10000 ms (2-byte unsigned short value)
			All parameters	0x0048	All Transmission Parameters See Table 11
			Modulator Data Interface	0x004D	See Table 12

	SET	0x0101	Symbol Rate	0x0040	1, 2, 3, 4, 5 Msp/s (1-byte unsigned char value)
			Transmit Power	0x0041	27-33 dBm, step 1 dB (1-byte unsigned char value)
			Center Frequency	0x0042	2000.000 - 2500.000 MHz step 1 kHz (4-byte float value)
			MODCOD	0x0043	1-23 (1-byte unsigned char value) (See Table 10)
			Roll-Off	0x0044	0.35 / 0.25 / 0.2 (0 / 1 / 2: 1-byte unsigned char value)
			Pilot Signal	0x0045	On / Off (1 / 0) (1-byte unsigned char value)
			FEC Frame Size	0x0046	Short / Normal (1 / 0) (1-byte unsigned char value)
			Pretransmission Delay	0x0047	2000 – 10000 ms (2-byte unsigned short value)
			All Parameters	0x0048	All Transmission Parameters (See Table 11)
			RS-485 Baud Rate	0x004B	0 - 11 (See Table 12) (1-byte unsigned char value)
			Modulator Data Interface	0x004D	See Table 13
Status	GET	0x0100	Report	0x0049	See Table 14

File System	DIR	0x0102	N/A	0x0000	See Table 16
	DIR Next	0x0103	N/A	0x0000	See Table 31
	Delete File	0x0104	N/A	0x0000	NULL terminated string with length between 3-30 characters See 3.1.3.3 Delete File Command
	Delete All Files	0x0105	N/A	0x0000	-
	Create File	0x0106	N/A	0x0000	See Table 62, Table 64
	Write File	0x0107	N/A	0x0000	See Table 73
	Open File	0x0108	N/A	0x0000	See Table 82, Table 84
	Read File	0x0109	N/A	0x0000	See Table 93, Table 95
	Send File/s	0x010A	Send File(s)	0x0050	See Table 105
			Send File(s) with RF Tract issue	0x0052	
System Configuration	Transmit Mode	0x0110	N/A	0xXXXX	-
	Idle Mode	0x0111	N/A	0xXXXX	-
	Update FW	0x0112	N/A	0xXXXX	See Table 40
	Safe Shut-down	0x0113	N/A	0xXXXX	See Table 43
Get Results	Get Results	0x0114	0xXXXX	0xXXXX	See all examples of commands described below

Table 10: MODCOD to Modulations

MODCOD	Modulation
1	1/4 QPSK
2	1/3 QPSK
3	2/5 QPSK
4	1/2 QPSK
5	3/5 QPSK
6	2/3 QPSK
7	3/4 QPSK
8	4/5 QPSK
9	5/6 QPSK
10	8/9 QPSK
11	9/10 QPSK
12	3/5 8PSK
13	2/3 8PSK
14	3/4 8PSK
15	5/6 8PSK
16	8/9 8PSK
17	9/10 8PSK
18	2/3 16APSK
19	3/4 16APSK
20	4/5 16APSK
21	5/6 16APSK
22	8/9 16APSK
23	9/10 16APSK

Table 11: All Transmission Parameters

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	9th byte
Symbol Rate	Transmit Power	MOD-COD	Roll-Off	Pilot Signal	FEC Frame Size	Pretrans-mission Delay	Center Frequency
1-byte unsigned char value	1-byte unsigned char value	1-byte unsigned char value	1-byte unsigned char value	1-byte un-signed char value	1-byte unsigned char value	2-byte unsigned short value	4-byte float value
These are the parameters described in Table 9 which are grouped in an array and this is their sequence.							

Table 12: RS-485 Baud Rate

RS-485 Baud Rate Setting	Communication Baud Rate in Kbits/s
0	250
1	500
2	1000
3	2000
4	3000
5	5000

Table 13: Modulator Data Interface

Modulator Data Interface Type	LVDS Input/Output Type
1st byte	2nd byte
1-byte unsigned char value	1-byte unsigned char value
0- Internal Data Interface - Data coming from SD card files 1-LVDS interface	0-PC104 Connector 1- LVDS Connector
For Internal Data Interface the second parameter (LVDS Input/Output Type) is insignificant.	

Table 14: Data field structure for the Simple Report

			Value	Description
1st byte	System State	1-byte unsigned char value	1	Sys After Reset
			2	Sys Idle Mode
			3	Sys Transmission Mode
			4	Sys Going to Shutdown
2nd byte	Status Flags	1-byte unsigned char value	Bit Field Warning and Error Flags	
			bit 0	Reserved
			bit 1	Reserved
			bit 2	SDR Not Initialized
			bit 3	Reserved
			bit 4	System Over Temperature Stop
			bit 5	Reserved
			bit 6	Reserved
			bit 7	Reserved
3rd byte	Reserved	2-byte unsigned short value	-	-
5th byte	CPU Temperature	4-byte float value	CPU temperature in range: -40 ÷ 125°C	
9th byte	Firmware version	4-byte unsigned int value	10200 (0x000027D8) default start firmware version	

Upon reading the report for the first time, the System State field has Sys After Reset value after resetting the module. This value is then automatically set to Sys Idle Mode. For this reason the host system detects when the module has been restarted and takes the appropriate actions.

3.1.2 Commands Syntax

The syntax of every command now follows. Even though all the functional fields in the ESUP packet are in Little-endian format the examples shown below are in Big-endian format for the sake of simplicity.

In all returned results of commands execution, the first byte in the information data field indicates the status of command execution. This should not be confused with the status of the protocol. The meaning of this byte is described for each command.

3.1.2.1 SET Command

Table 15 gives further details of the SET command. Tables from 16 to 22 show an example for setting two different parameters with two commands: symbol rate to 5 Msym/s and pretransmission delay to 3000 ms of the S-Band Transmitter.

Table 15: SET command syntax

Command	One of the Following Types	Data	Data Length
SET 0x0101	0x0040 = Symbol Rate 0x0041 = Tx Power 0x0042 = Centrer Frequency 0x0043 = MODCOD 0x0044 = Roll-Off 0x0045 = Pilot Signal On/Off 0x0046 = FEC Frame size 0x0047 = Pretransmission Delay 0x0048 = All Parameters 0x004B = RS-485 Baud Rate	Data depends on the type of command. See Table 9	Data length depends on the type of command. It varies in the range: 1 - 12 bytes

Table 16: Example – SET Symbol Rate, Set pretransmission delay command from Host

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0101	0x0040	0x05 (5 Msp/s)	0xXXXXXXXX
0x45535550	0xXXXX	0x0002	0x0000	0x0101	0x0047	0x0BB8 (3000mSec)	0xXXXXXXXX

Table 17: Example – SET Symbol Rate, Set pretransmission delay reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0101	0x0040	-	0xXXXXXXXX
0x45535550	0xXXXX	0x0000	0x0005	0x0101	0x0047	-	0xXXXXXXXX

Table 18: Example - GetResult request for: SET Symbol Rate and Set pretransmission delay commands from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0002	0x0000	0x0114	0x0101	0x0040	0xXXXXXXXX
0x45535550	0xXXXX	0x0002	0x0000	0x0114	0x0101	0x0047	0xXXXXXXXX

Table 19: Example - Returned results for GetResult requests from Module when BUSY is returned

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0007	0x0101	0x0040	-	0xXXXXXXXX
0x45535550	0xXXXX	0x0000	0x0007	0x0101	0x0047	-	0xXXXXXXXX

Table 20: Example - Returned results for GetResult requests from Module when Result is returned

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0101	0x 0040	0x00 (status of command execution)	0xXXX XXXXX
0x45535550	0xXXXX	0x0001	0x0000	0x0101	0x 0047	0x00 (status of command execution)	0xXXX XXXXX

Table 21: Meaning of the status byte for "*Set*" of all RF parameters

Value	Value Name
0x00	OK
0x01	System Error
0x02	Busy
0x03	Wrong parameter

Table 22: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0101	0x0040	-	0xXXX XXXXX
0x45535550	0xXXXX	0x0000	0x0005	0x0101	0x0047	-	0xXXX XXXXX

3.1.2.2 GET Command

Table 23 gives further details of the GET command. Tables from 24 to 30 show an example for getting two different parameters with two commands: symbol rate and pretransmission delay of the S-Band Transmitter.

Table 23: GET command syntax

Command	One of the Following Types	Data	Data Length
GET 0x0100	0x0040 = Symbol Rate 0x0041 = Tx Power 0x0042 = Centrer Frequency 0x0043 = MODCOD 0x0044 = Roll-Off 0x0045 = Pilot Signal On/Off 0x0046 = FEC Frame size 0x0047 = Pretransmission Delay 0x0048 = All Parameters 0x0049 = Status Report	Data depends on the type of command. See Table 9	Data length depends on the type of command. It varies in the range: 2-12 bytes

Table 24: Example – GET Symbol Rate, Set pretransmission delay command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0100	0x0040	-	0xXXX XXXXX
0x45535550	0xXXXX	0x0000	0x0000	0x0100	0x0047	-	0xXXX XXXXX

Table 25: Example - GET Symbol Rate, Set pretransmission delay reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0100	0x0040	-	0xXXX XXXXX
0x45535550	0xXXXX	0x0000	0x0005	0x0100	0x0047	-	0xXXX XXXXX

Approximate time for command execution – from 200 µs to 500 µs.

Table 26: ExampleGetResult request for: GET Symbol Rate and Get pretransmission delay commands from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0002	0x0000	0x0114	0x0100	0x0040	0xXXX XXXXX
0x45535550	0xXXXX	0x0002	0x0000	0x0114	0x0100	0x0047	0xXXX XXXXX

Table 27: Example - Returned results for GetResult requests from Module when BUSY is returned

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0007	0x0100	0x0040	-	0xXXXXXXXX
0x45535550	0xXXXX	0x0000	0x0007	0x0100	0x0047	-	0xXXXXXXXX

Table 28: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0100	0x0040	0x01 (status of command execution - sys error)	0xXXX XXXXX
0x45535550	0xXXXX	0x0001	0x0000	0x0100	0x0047	0x01 (status of command execution - sys error)	0xXXX XXXXX

Table 29: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is **OK**

ESUP packet								
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data		CRC32
0x 45535550	0x XXXX	0x 0002	0x 0000	0x 0100	0x 0040	Status of command execution	Data	0xXXX XXXXX
						0	0x05 (5)	
0x 45535550	0x XXXX	0x 0003	0x 0000	0x 0100	0x 0047	Status of command execution	Data	0xXXX XXXXX
						0	0x0BB8 (3000 mSec)	

Table 30: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0100	0x0040	-	0xXXX XXXXX
0x45535550	0xXXXX	0x0000	0x0005	0x0100	0x0047	-	0xXXX XXXXX

3.1.3 FILE COMMANDS

Short additional description

The module does not work with subdirectories of the file system. All files are located in the root directory. Thus faster access and saving space on the SD card are achieved. This fact should be taken into consideration during development and testing of the host system software and when saving files from a computer on the SD card. The module uses the following file name convention: file names can be up to 30 characters in total length and no shorter than 3 characters. File names can contain uppercase and lowercase letters, numbers, underscores, and only one period ("A..Z", "a..z", "0..9", "_ ,."). For any symbols other than the aforementioned, the module returns an error. The only exception to this rule is the Send_File_s command, whose name parameter can contain the star (*) symbol. A more detailed description of this is available in Chapter 3.1.3.9 Send File/s command.

To save a file in the module, the file must be created and then the contained information must be saved. That is why it is necessary that both the Create_File and Write_File commands are used. It is not allowed to add additional data in a file. If an addition is needed, the file must be deleted and a new one with the addition must be created. To read a file from the module, both the Open_File and Read_File commands are used. Additional description of this functionality is available in the chapters describing the following commands: Create_File, Write_File, Open_File and Read_File.

3.1.3.1 DIR Command

The DIR command is executed in the same manner as the already described commands by using the ESUP protocol. The following Table 31 provides details for the content of the data field.

Table 31: DIR command Data field details

1st byte	2nd byte	3rd byte	5th byte to Nth byte	
Status of Command Execution	Flag More Files Avalible	File Name Count	List of file names sequence	
1-byte unsigned char value	1-byte unsigned char value	2-byte unsigned short value	Structure of one part of sequence	
			Null terminated string with length between 3 and 30 bytes	4 bytes unsigned int value show-ing file length in bytes

Table 32: DIR command Data field example

1st byte	2nd byte	3rd byte	5th byte	16th byte	20th byte	32nd byte
0x00	0x00	0x02	54 65 73 74 5f 31 2e 74 78 74 00 (Test_1.txt\0)	0x0000 0064	45 78 61 6d 70 6c 65 2e 74 78 74 00 (Example.txt\0)	0x00000096

- The first byte of the data field, as described in 3.1.2 Commands Syntax, shows the status of command execution. If it is different from 0 (OK) and there are subsequent bytes, then they are insignificant (usually the message ends only with the status);
- The second byte is a flag value. When this flag is active (0x01), it indicates that in addition to the files currently available in the list, other supplementary files are present on the SD card. This flag also shows that you can use the DIR_Next command to retrieve the remaining (not shown) files. If this flag is inactive (0x00), then the current list contains all the files that the SD card accomodates in its main folder;
- The third byte indicates how many file names are present in the list;
- After the fourth byte, the list of file names follows. What comes next is their size in bytes until the volume of the data field is filled or the number of files on the SD card is over (see Table 31). Each file name in the list ends with a terminating NULL. The next four bytes are an unsigned int number, showing the size of the corresponding file in bytes. In the example below, the Test_1.txt file is 100 bytes long and the Example.txt file is 150 bytes long.

An example for DIR command usage

Table 33: Example – DIR command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0102	0x0000	-	0xXXX XXXXX

Table 34: Example – DIR reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0102	0x0000	-	0xXXX XXXXX
Approximate time for command execution – from 500 µs to 10 ms. It depends on the number and length of the files in the SD card.							

Table 35: Example - GetResult request for: DIR command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0102	-	0xXXXXXXXX

Table 36: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0102	0x0000	0x03 (status of command execution: Card Error)	0xXXX XXXXX

Table 37: Meaning of the status byte for the DIR command

Value	Value Name
0x00	OK
0x03	File Card Error

Table 38: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0023	0x0000	0x0102	0x0000	Table 32	0xXXX XXXXX

Table 39: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0102	0x0000	-	0xXXXXXXXX

3.1.3.2 DIR Next Command

The DIR_Next command is executable provided that the DIR command has already been executed and has returned 0x01 in its 2nd byte of the ESUP Data functional field (described in the DIR command section). The DIR_Next command can be used once or multiple times consequtively until all the files from the S-Band Transmitter's SD card internal memory are read. The syntax of the command is the same as the one of the DIR command.

An example for DIR_Next command usage

Table 40: Example – DIR_Next command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Comand	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0103	0x0000	-	0xXXXXXXXX

Table 41: Example – DIR_Next reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Comand	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0103	0x0000	-	0xXXXXXXXX

Approximate time for command execution – from 500 µs to 10 ms. It depends on the number and length of the files in the SD card.

Table 42: Example - GetResult request for DIR_Next command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0103	-	0xXXXXXXXX

Table 43: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0103	0x0000	0x03 (status of command execution – Card Error)	0xXXX XXXXX

Table 44: Meaning of the status byte for the DIR_Next command

Value	Value Name
0x00	OK
0x03	File Card Error

Table 45: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32

0x45535550	0xXXXX	0x0023	0x0000	0x0103	0x0000	Table 33	0xXXX XXXXX

Table 46: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0103	0x0000	-	0xXXX XXXXX

3.1.3.3 Delete File Command

The Delete_File command works with file names as described above. It can delete only one file from the SD card. Depending on the command execution status that is generated upon getting the result, the file is either successfully deleted or an additional check is needed. This check is done with the DIR and DIR_Next commands to validate whether the file still exists on the SD card or has already been deleted. If it still exists, another attempt to delete it can be made. When you try to delete a non-existing file, a corresponding status is returned as a result of command execution.

An example how to use the Delete_File command in order to delete the „Example.txt“ file

Table 47: Example – Delete_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x000C	0x0000	0x0104	0x0000	45 78 61 6D 70 6C 65 2E 74 78 74 00 Example.txt\0	0xXXX XXXXX

Table 48: Example – Delete_File reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0104	0x0000	-	0xXXX XXXXX

Table 49: Example - GetResult request for: Delete_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0104	-	0xXXX XXXXX

Approximate time for command execution – from 200 µs to 600 µs.

Table 50: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0104	0x0000	0x04 (status of command execution: Communi-cation Error)	0xXXX XXXXX

Table 51: Meaning of the status byte for the Delete_File command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	File Not Found	File is not found on the SD card or does not exist
0x03	File Card Error	Internal SD card error
0x04	File Communication Error	Wrong file name

Table 52: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0104	0x0000	0x00 (status of command execution-OK)	0xXXX XXXXX

Table 53: Example - Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0104	0x0000	-	0xXXX XXXXX

3.1.3.4 Delete All Files Command

The Delete_All_Files command deletes all files from the root folder of the SD card except for the service files Error_Log.txt, Index_Log.txt. Depending on the command execution status that is generated upon getting the result, the files are either successfully deleted or an additional check is needed. This check is done with the DIR and DIR_Next commands to validate whether the files still exist on the SD card or have already been deleted. If they still exist, another attempt to delete them can be made.

An example for **Delete_All_Files command usage**

Table 54: Example – Delete_All_Files command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0105	0x0000	-	0xXXX XXXXX

Table 55: Example – Delete_ All_Files reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0105	0x0000	-	0xXXX XXXXX

Approximate time for command execution – from 600 µs to 100 ms. It depends on the number of files written on the SD card.

Table 56: Example - GetResult request for: Delete_ All_Files command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0105	-	0xXXX XXXXX

Table 57: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0105	0x0000	0x03 (status of command execution - Card Error)	0xXXX XXXXX

Table 58: Meaning of the status byte for the Delete_All_Files command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	File Not Found	File is not found on the SD card or does not exist
0x03	File Card Error	Internal SD card error

Table 59: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0105	0x0000	0x00 (status of command execution - OK)	0xXXX XXXXX

Table 60: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0105	0x0000	-	0xXXXXXXXX

3.1.3.5 Create File Command

The Create_File command is executed in the same manner as the already described commands by using the ESUP protocol. The following Table 61 provides details for the content of the data field.

Table 61: Create_File command data field details

Create File command data field	
Null terminated string with length between 3 and 30 bytes	4 bytes unsigned int value showing file length in bytes

Table 62: Create_File command Data field example

Create File command data field example	
4E 65 77 5F 46 69 6C 65 2E 74 78 74 00 (New_File.txt\0)	0x00EA82FC (15368956 bytes)

Table 63: Create_File command ANSWER Data field details

Create File command ANSWER data field	
Status byte	4 bytes signed int value - handle

Table 64: Create_File command ANSWER Data field example

Create File command ANSWER data field example	
0x00	0x6F38AC5F

The file name ends with a terminating NULL. The next four bytes are an unsigned int number, showing the size of the corresponding file in bytes. In the example below, the New_File.txt file is 15368956 bytes long. File names are generated according to the convention described in Chapter 3.1.3 File Commands.

Upon successful execution of the Create_File command, the returned result for command execution contains a 4-byte signed int number in addition to the status. It is a handle to the already created file for the host system. When the file is successfully created, the value of this handle is different from -1. The handle is used later when the Write_File command is applied.

The command returns an error if you try to create a file with a name of an already existing file. The parameters used to call the Create_File command ensure that in case a file with the same name exists, it will not be overwritten.

If another file command different from Write_File is called, then after the Create_File command has been successfully executed, this handle is deleted and the newly created file is closed with a zero length. The handle is valid until a different file command is called or until the module is restarted.

An example for **Create_File command usage**

Table 65: Example – Create_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0011	0x0000	0x0106	0x0000	Table 62	0xXXXXXXXX

Table 66: Example – Create_File reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0106	0x0000	-	0xXXXXXXXX

Approximate time for command execution – from 200 µs to 600 µs.

Table 67: Example - GetResult request for: Create_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0106	-	0xXXXXXXXX

Table 68: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0106	0x0000	0x01 (status of command execution - Cannot Be Opened)	0xXXX XXXXX

Table 69: Meaning of the status byte for the Create_File command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	File Cannot Be Opened	File cannot be opened due to internal reasons or file name error

Table 70: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0005	0x0000	0x0106	0x0000	Table 64	0xXXX XXXXX

Table 71: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0106	0x0000	-	0xXXX XXXXX

3.1.3.6 Write File Command

The Write_File command is executed in the same manner as the already described commands by using the ESUP protocol. The Write_File command must only be called after the Create_File command has already been initiated and successfully finished - the host system has a valid handle which is different from -1.

As the maximum data length of a Write_File command is 1472 bytes, when saving a longer file the Write_File command is called multiple times. The data length in the data field is determined by the host system. It is recommended that the maximum length is used when transferring large files. The host system is responsible for dividing the output file into separate consecutive pieces of up to 1472 bytes and for putting each piece into a separate Write_File command with corresponding packet numbers starting from zero. This sequence must be maintained until the entire file is transferred to the module. It is vitally important that data length must be multiple of 4. The only exception to this rule can be the last data piece of a file.

The following information preceeds the file data in the data field of the command:

- length of file data in the data field of the current packet;
- file handle - returned by the Create_File command;
- packet number of the current packet.

While the corresponding file is being transferred, the same file handle is used. The module automatically compares the length of the transferred information with the file length, which is specified in the Create_File command. After all the file data has been transferred, the module automatically closes the file within its own file system. If another file command is called before the file transfer has been accomplished, the file is closed with the heretofore transferred data. If the module is restarted during the file transfer, the heretofore transferred data is ignored and the file remains with zero length.

Table 72: Write_File command data field details

1st byte	3rd byte	7th byte	11th byte to Nth byte
Data Packet Length	File Handle	Packet Number	Current Packet File Data
2-byte unsigned short value	4-byte signed int value	4-byte unsigned int value	1 - 1472 bytes

Table 73: Write_File command data field example

1st byte	3rd byte	7th byte	11th byte to Nth byte
0x0010	0x6F38AC5F	0x00000000	0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
2-byte unsigned short value	4-byte signed int value	4-byte unsigned int value	1 - 1472 bytes

In the example clarifying the data field of the Write_File command, you convey information for recording 16 bytes of data (the numbers from 0 to 15) in the first zero count packet with data to record in a file.

An example for **Write_File** command usage

Table 74: Example – Write_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x001A	0x0000	0x0107	0x0000	Table 73	0xXXX XXXXX

Table 75: Example – Write_File reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0107	0x0000	-	0xXXX XXXXX
Approximate time for command execution – from 1 ms to 3 ms.							

Table 76: Example - GetResult request for: Write_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0107	-	0xXXX XXXXX

Table 77: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0107	0x0000	0x01 (status of com-mand execu-tion – Invalid Handle)	0xXXX XXXXX

Table 78: Meaning of the status byte for the Write_File command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	File Invalid Handle	Invalid File Handle – because it is diffrenet from the value returned by Create_File command or File Handle closed in module's system
0x02	Not Enough Space	Card is full
0x03	Card Error	Internal card error
0x04	File Communication Error	Different values for length in protocol header and data field
0x05	Invalid Packet Number	The module expects a different packet number from the one written in the current data field

Table 79: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0107	0x0000	Status from file operation – 0 (OK)	0xXXX XXXXX

Table 80: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0107	0x0000	-	0xXXX XXXXX

3.1.3.7 Open File Command

The Open_File command is executed in the same manner as the already described commands by using the ESUP protocol. The following Table 81 provides details for the content of the data field.

Table 81: Open_File command data field details

Open File command data field
NULL terminated string with length between 3 and 30 bytes

Table 82: Open_File command Data field example

Create File command data field example
45 78 69 73 74 69 6E 67 5F 46 69 6C 65 2E 74 78 74 00 (Existing_File.txt\0)

Table 83: Open_File command ANSWER Data field details

Create File command ANSWER data field		
Status byte	4 bytes signed int value - handle	4 bytes unsigned int value – file length

Table 84: Open_File command ANSWER Data field example

Create File command ANSWER data field example		
0x00	0x6F38AC5F	0x00000010

The file name ends with a terminating NULL. In the example below, the Existing_File.txt file is 16 bytes long. File names are generated according to the convention described in Chapter 3.1.3 File Commands.

Upon successful execution of the Open_File command, the returned result for command execution contains two 4-byte numbers in addition to the status. The first one is a 4-byte signed int. It is a handle to the already opened file for the host system. When the file is successfully opened, the value of this handle is different from -1. The handle is used later when the Read_File command is applied. The second number is a 4-byte unsigned int. It shows the size of the opened file in bytes.

The command returns an error if you try to open a non-existing file or if you submit a file name containing symbols which are different from the aforementioned (see *Chapter 3.1.3 File Commands*).

If another file command different from Read_File is called, then after the successful execution of the Open_File command, this handle is deleted and the file is closed. The handle is valid until the whole file is read (through Read commands), until another file command is called or until the module is restarted.

An example for **Open_File** command usage

Table 85: Example – Open_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0012	0x0000	0x0108	0x0000	Table 82	0xXXX XXXXX

Table 86: Example – Open_File reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0108	0x0000	-	0xXXX XXXXX

Approximate time for command execution – from 200 µs to 1 ms.

Table 87: Example - GetResult request for: Open_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0108	-	0xXXX XXXXX

Table 88: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0108	0x0000	0x01 (status of command execution - Cannot Be Opened)	0xXXX XXXXX

Table 89: Meaning of the status byte for the Open_File command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	File Cannot Be Opened	File cannot be opened due to internal reasons or file name error
0x03	Card error	Internal card error

Table 90: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x000A	0x0000	0x0108	0x0000	Table 84	0xXXX XXXXX

Table 91: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0108	0x0000	-	0xXXX XXXXX

3.1.3.8 Read File Command

The Read_File command is executed in the same manner as the already described commands by using the ESUP protocol. The Read_File command must only be called after the Open_File command has already been initiated and successfully finished - the host system has a valid handle which is different from -1.

As the maximum data length of a Read_File command is 1472 bytes, when reading a longer file the Read_File command is called multiple times. The data length in the data field of the reply is the maximum possible except for the last packet. The module starts transferring data with a packet number 0 and increments this number automatically at every subsequent call of the Read_File command.

The following information preceeds the file data in the data field of the command reply:

- status from command execution;
- length of file data in the data field of the current packet;
- packet number of the current packet.

The module system is responsible for dividing the output file into separate consecutive pieces of up to 1472 bytes and for putting each piece into a separate Read_File command with corresponding packet numbers starting from zero. This sequence is maintained until the entire file is transferred to the host.

While the corresponding file is being transferred, the same file handle is used. After all the file data has been transferred, the module automatically closes the file within its own file system. If another file command is called before the file transfer has been accomplished, the file automatically closes and the handle receives the value -1.

Table 92: Read_File command data field details

Read File command data field
File Handle (4-byte signed int) returned by Open_File command

Table 93: Read_File command Data field example

Read File command data field example
0x6F38AC5F

Table 94: Read_File command ANSWER data field details

1st byte	2nd byte	4th byte	8th byte to Nth byte
Read_File command status	Current Data Packet Length	Packet Number	Current Packet File Data
1-byte unsigned char value	2-byte unsigned short value	4-byte unsigned int value	1 - 1472 bytes

Table 95: Read_File command ANSWER data field example

1st byte	2nd byte	4th byte	8th byte to Nth byte
0x00	0x00000010	0x00000100	0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
1-byte unsigned char value	2-byte unsigned short value	4-byte unsigned int value	1 - 1472 bytes

In the example for data field of the Read_File command we submit information for 16 bytes of data read (the numbers from 0 to 15) in the 256th count packet with file data.
An example for Read_File command usage

Table 96: Example – Read_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0004	0x0000	0x0109	0x0000	Table 93	0xXXXXXXXX

Table 97: Example – Read_File reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0109	0x0000	-	0xXXXXXXXX

Table 98: Example - GetResult request for Read_File command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0109	-	0xXXX XXXXX
Approximate time for command execution – from 500 µs to 2 ms.							

Table 99: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0109	0x0000	0x01 (status of command execution – Invalid Handle)	0xXXX XXXXX

Table 100: Meaning of the status byte for the Read_File command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	File Invalid Handle	Invalid File Handle – because it is diffrenet from the value returned by Open_File command or File Handle closed in module's system
0x03	Card Error	Internal card error

Table 101: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0017	0x0000	0x0109	0x0000	Table 95	0xXXX XXXXX

Table 102: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0109	0x0000	-	0xXXX XXXXX

3.1.3.9 Send File/s Command

Additional description

The Send_File_s command is used to send a file or a series of files from the SD card via the RF channel of the module. The command can only be executed in Transmit Mode. If the command is called in Idle Mode, it returns an error. Once initiated, the command cannot be interrupted unless the module is set to Idle Mode. This feature must be taken into account by the host system logic when planning to send files via the RF channel. Moreover, the input interface of the modulator must be set in Internal Data Interface mode (see Table 12).

The command can be executed at two different module states. It is the host system that chooses whether/when to execute the command. The two states are as follows:

- The module works normally and everything is OK (type: Send File/s without issue);
- Incorrect or incomplete SDR initialization or disturbed communication with BUC (type: Send file with RF tract issue).

The host system is given the opportunity to try to send the necessary information from the module despite the problems with the module. Practically, these are emergency situations, but this opportunity exists as an extreme option to send information since there is no other way to influence the module. When an RF tract issue is present, it is not clear whether the information can be sent, hence an explicit opportunity exists.

The Sent_File command serves for sending a single file or a set of files with partially identical names. A functionality for manipulating a set of files (similar to DOS commands) is implemented. This functionality is applied for partial replacement of characters in a file name by using the * symbol. There are a number of possible combinations for using the symbol *, which are described in Table 103.

Table 103: Possible combinations of using the * symbol in the file name parameter of a command

Combination	Description
*S.S	The command Send_File_s read from the SD card and sends all the files which have an identical part in their names. * - denotes a char or a string which is different in all the file names S - denotes either a character or a string which is the same in all the file names
S.	
S.S	
S.*	
.	
S*.*	
S*.S	

The Send_File_s command is executed in the same manner as the already described commands by using the ESUP protocol.

Table 104: Send_File_s command data field details

Send_File_s command data field
NULL terminated string with length between 3 and 30 bytes

Table 105: Send_File_s command Data field example for sending one file

Send_File_s command data field example
45 78 69 73 74 69 6E 67 5F 46 69 6C 65 2E 74 78 74 00 (Existing_File.txt\0)

Table 106: Send_File_s command Data field example for sending a couple of files

Send_File_s command data field example
2A 5F 46 69 6C 65 2E 74 78 74 00 (*_File.txt\0)

The command reads and sends all files which have the following identical part in their names: _File.txt

An example for **Send_File_s** command usage

Table 107: Example – Send_File_s command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0012	0x0000	0x010A	0x0049	Table 105	0xXXX XXXXX

Table 108: Example – Send_File_s reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x010A	0x0049	-	0xXXX XXXXX

Approximate time for command execution – it depends on the length of trasmitted file/s divided by the effective data bit transmission rate of the modulator. Effective transmission rate depends on tuned symbol rate and modulation.

Table 109: Example - GetResult request for: Send_File_s command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0002	0x0000	0x0114	0x010A	0x0049	0xXXX XXXXX

Table 110: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x010A	0x0049	0x02 (status of com-mand execu-tion – Not All Files Sent)	0xXXX XXXXX

Table 111: Meaning of the status byte for the Send_File_s command

Value	Value Name	Description
0x00	OK	Operation successful
0x02	Not All Files Sent	
0x03	Card error	
0x04	File Communication Error	
0x05	File not ready SYS	

Table 112: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32

0x45535550	0xXXXX	0x0001	0x0000	0x010A	0x0049	0 (OK - file sent sucessfully)	0xXXX XXXXX

Table 113: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x010A	0x0049	-	0xXXX XXXXX

3.1.3.10 Transmit Mode Command

Additional Description

When switching from Idle Mode to Trasmit Mode, the module turns on all RF systems (modulator, SDR, RF power amplifier) with their default or previously modified parameters. An important detail is that no other commands related to reading or changing the RF configuration parameters are executed while the command is being executed. Only file commands except for Sent_File_s commands and commands for reading a status can be executed in parallel.

After the module switches to Transmit Mode, it starts filling information in the data fields of the status messages, which have zero values in Idle Mode. The Transmit_Mode command is executed in the same manner as the already described commands by using the ESUP protocol.

An example for Transmit_Mode command usage

Table 114: Example – Transmit_Mode command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0110	0x0000	-	0xXXX XXXXX

Table 115: Example – Transmit_Mode reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0110	0x0000	-	0xXXX XXXXX

Approximate time for command execution – from 19 s to 40 s.

Table 116: Example - GetResult request for Transmit_Mode command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0110	-	0xXXX XXXXX

Table 117: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0110	0x0000	0x01 (status of command execution - System Error)	0xXXX XXXXX

Table 118: Meaning of the status byte for the Transmit_Mode command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	System Error	Configuration Not Completed Successfully

If an error is returned during command execution, additional information can be obtained by reading the extended status and paying attention to the flag field.

Table 119: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0110	0x0000	0x00 (OK)	0xXXX XXXXX

Table 120: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0110	0x0000	-	0xXXX XXXXX

3.1.3.11 Idle Mode Command

When switching to Idle Mode, the module turns off all RF systems (modulator, SDR, RF power amplifier). If a file is being transferred from the SD card at the moment when a Idle Mode command is initiated, the file/s transmission is interrupted immediately.

Switching to Idle Mode can also happen automatically when the CPU temperature reaches 70°C. If this happens, the System Over Temperature Stop flag remains raised among all the status flags, which is an indication for the host system that the event has occurred.

Due to the temperature hysteresis realized, the module can be switched to Transmit Mode again, only after the temperature decreases with at least 15°C.

The Load_Mode command is executed in the same manner as the already described commands by using the ESUP protocol.

An example for Load_Mode command usage

Table 121: Example – Load_Mode command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0111	0x0000	-	0xXXX XXXXX

Table 122: Example – Load_Mode reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0111	0x0000	-	0xXXX XXXXX

Approximate time for command execution – from 200 µs to 1 ms.

Table 123: Example - GetResult request for: Load_Mode command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0111	-	0xXXX XXXXX

This command returns as a result only OK.

Table 124: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0111	0x0000	0x00 (OK)	0xXXX XXXXX

Table 125: Example – Sent ACKNOWLEDGE to Module for Result from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0111	0x0000	-	0xXXX XXXXX

3.1.3.12 Update Firmware Command

The firmware update of the module is performed as described below. Firstly, you need to save a file with the update provided by EnduroSat on the SD card. The module must be set in Idle Mode. A command for firmware update is sent. It has as a parameter the name of the file containing the update. Before the module update is initiated, the module stops all currently running tasks except for the tasks related to the communication with the host system.

If any error related to the update procedure (incorrect file data, damaged file, old firmware version, inappropriate update, etc.) occurs while the firmware update command is being executed, the module restarts automatically. Also, upon successful accomplishment of the update, the module restarts automatically. Due to this fact, the host system may not be able to retrieve the result of command execution. Therefore, the host system must incorporate a logic which is organized so that during a firmware update it starts checking the status reported by the module after a definite timeout (2s) expires. The state after reset must also be checked (see Table 14) and the firmware

version after the update must be higher than the one before the update. A difference in the versions is a guarantee for a successful update. If the host system manages to get the result of the update no matter what it is, it must start working with the module exactly in the same manner as in a state after reset.

Table 126: Update_FW command data field details

Open File command data field
NULL terminated string with length between 3 and 30 bytes

Table 127: Update_FW command Data field example

Create File command data field example
55 70 64 61 74 65 2E 62 69 6E 00 (Update.bin\0)

The file name ends with a terminating NULL. In the example below, the Update.bin file is 11 bytes long. File names are generated according to the convention described in Chapter 3.1.3 File Commands.

The Update_FW command is executed in the same manner as the already described commands by using the ESUP protocol.

An example for **Update_FW** command usage

Table 128: Example – Update_FW command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0112	0x0000	-	0xXXX XXXXX

Table 129: Example – Update_FW reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0112	-	0xXXX XXXXX

ENDUROSAT

Approximate time for command execution – from 500 ms to 2 s.

Table 130: Example - GetResult request for: Update_FW command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x000B	0x0000	0x0112	0x0000	Table 127	0xXXX XXXXX

Table 131: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Com-mand Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0112	0x0000	0x01 (status of command execution – Cannot Be Opened)	0xXXX XXXXX

Table 132: Meaning of the status byte for the Update_FW command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	-	FW File Corrupted
0x02	-	Earlier or the same FW version installed
0x03	Card error	Internal card error
0x04	-	Internal Error
0x05	-	Internal Error
0x06	-	Internal Error
0x07	-	Internal Error
0x08	-	Internal Error
0x09	-	Internal Error

Table 133: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0112	0x0000	0x00 (OK)	0xXXX XXXXX

3.1.3.13 Safe Shutdown Command

The Safe_Shutdown command shuts down everything except for the communication task of the module, records service information onto the SD card and halts the processor. It is a good practice to use the Safe_Shutdown command as it ensures eventual extraction of a history log for the work of the module from the service information, if necessary. As the host system may not read the response from command execution, after about 0.6 s the processor stops responding to any messages. This can be used as a reference for successful command execution. Then the host system must turn off the power of the module.

The Safe_Shutdown command is executed in the same manner as the already described commands by using the ESUP protocol.

An example for Safe_Shutdown command usage

Table 134: Example – Safe_Shutdown command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0113	0x0000	-	0xXXX XXXXX

Table 135: Example – Safe_Shutdown reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0113	0x0000	-	0xXXXX XXXX

Table 136: Example - GetResult request for: Safe_Shutdown command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0000	0x0114	0x0113	-	0xXXX XXXXX

The command returns only status OK.

Table 137: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0113	0x0000	0x00 (OK)	0xXXX XXXXX

3.1.3.14 Change Baud Rate Command

The Change_Baud_Rate command changes the communication speed between the host and the module. The speeds at which the module can operate are given in Table 12. An important detail about this command is that the command and the ACKNOWLEDGE of the command are transferred at the current speed, while the GetResult and its response are transferred at a new speed, provided that the command is executed successfully. This must be kept in mind upon developing the host system software.

The Change_Baud_Rate command is executed in the same manner as the already described commands by using the ESUP protocol.

Approximate time for command execution – from 500 µs to 600 ms.

An example for Change_Baud_Rate command usage

Table 138: Example – Change_Baud_Rate command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0101	0x004B	0x05	0xXXX XXXXX

Table 139: Example – Change_Baud_Rate reply of command from Module

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0000	0x0005	0x0101	0x004B	-	0xXXX XXXXX

Approximate time for command execution – from 100 µs to 200 µs.

Table 140: Example - GetResult request for: Change_Baud_Rate command from Host

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0002	0x0000	0x0114	0x0101	0x004B	0xXXX XXXXX

Table 141: Example – Possible returned results for GetResult requests from Module when Result is returned and status of command execution contains an error

ESUP packet							
Header	Module ID	Data Length	Command Status	Command	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0101	0x004B	0x01 (status of command execution – Cannot Be Opened)	0xXXX XXXXX

Table 142: Meaning of the status byte for the Change_Baud_Rate command

Value	Value Name	Description
0x00	OK	Operation successful
0x01	-	Wrong parameters

Table 143: Example - Returned results for GetResult requests from Module when Result is returned and status of command execution is OK

ESUP packet							
Header	Module ID	Data Length	Command Status	Com-mand	Type	Data	CRC32
0x45535550	0xXXXX	0x0001	0x0000	0x0101	0x00 4B	0x00 (OK)	0xXXXXXXXX

Note:

Protocol Example

The archive file attached to this documentation contains real sniffed data from a logic analyzer for several commands. By using the free Saleae Logic software available in Internet for a logic analyzer, these files can be revised and used as a reference example of how the communication between the host and the module should proceed. An Internet address for downloading this software is stored in a read_me file in the same archive.

ENDUROSAT

4 EXAMPLE USAGE OF CONTAINER WITH COMMANDS

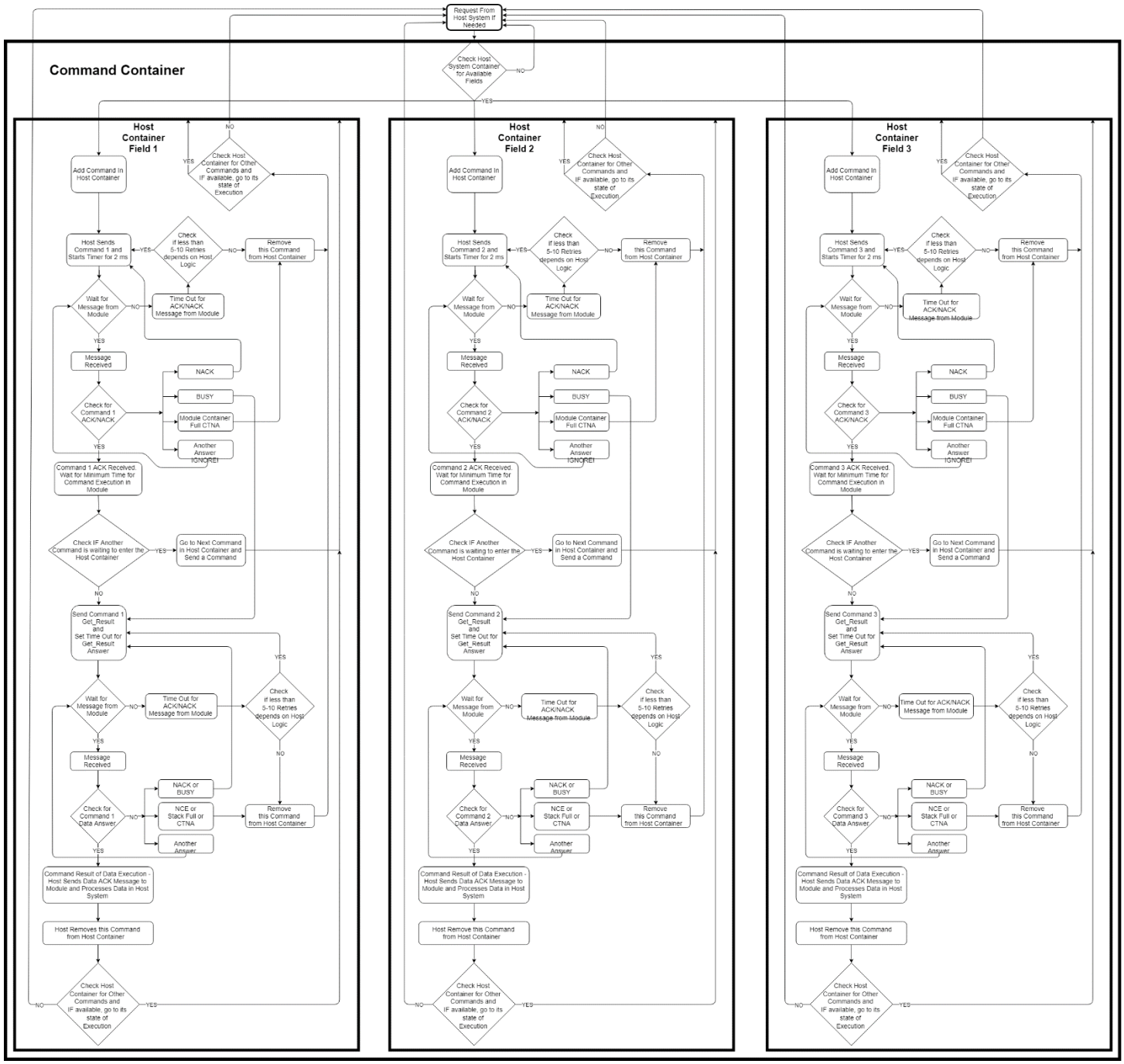

Figure 8: Example for Detailed Message Flow

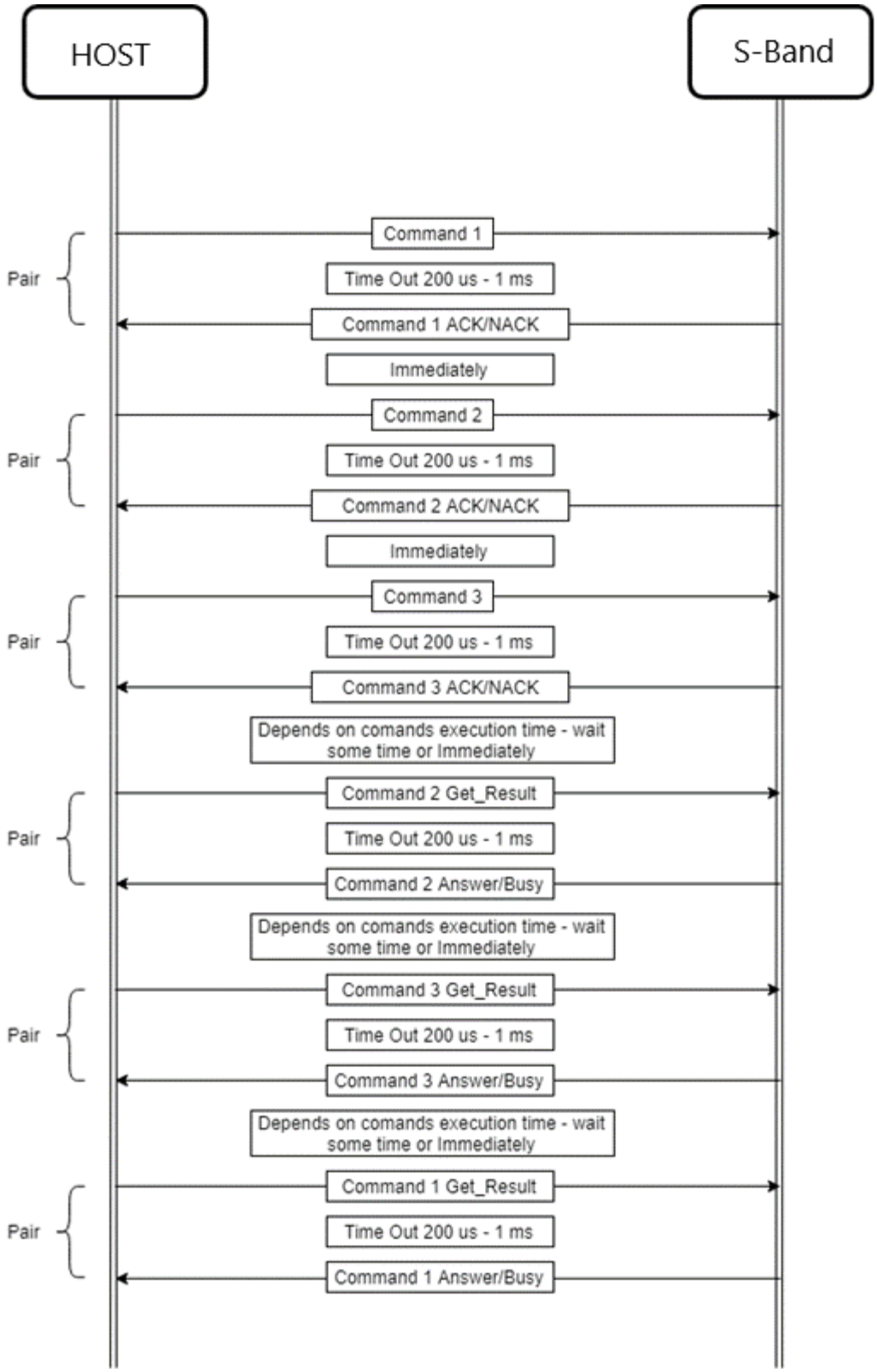

Figure 9: Example for Message Flow from Container

Figure 8 presents the simulation work with three commands, which is carried out through a command container within the host system. In order to operate optimally with the command container in the module, the host system must have implemented a container as well.

Figure 9 shows an example flow of messages from the host container to the module container. It is the logic implemented within the host that determines how and to what extent the host container (module container) will be filled. The sequence of message pairs can be arranged in a different way than the one in the example. It is important that the messages go in pairs and that the pair with the command preceeds the pair with the result of command execution.

5 LVDS INTERFACE

The LVDS interface consists of two LVDS pairs - DATA and CLOCK. The CLOCK LVDS pair works at constant clock rate which can be set up to 160 MHz. The rate of the CLOCK is defined by the Payload (Data Source). The signal rates at both DATA and CLOCK pairs must always be equal and constant during transmission. This means that if the clock rate is 100 MHz, the transmission rate on the DATA pair must be 100 MBaud/s.

The signal rate on the DATA LVDS pair is called transmission rate. It is different from the data rate. Since data may be sent with lower rates than the transmission rate, a data stuffing is implemented (filler). The special character K28.5 (comma) is used for this purpose. Also, two consequent K28.5 characters must be sent occasionally for synchronization. The last requirement will always be fulfilled as long as data rate is less than half the transmission rate. If data rate is equal to or more than half the transmission rate, it may happen that there are never two consequent K28.5 characters being sent. In order to avoid this scenario, at least two consequent K28.5 symbols must be sent for every 1880 bits sent.

The LVDS interface is capable of dealing with up to 130 Mbps data rate. Data is encoded in 8b/10b as described in EN 50083-9 Annex B and C. The transmission format is characterized with data bursts as shown in Figure B.8 in the same document. The difference is that for EnduroSat use case there are no MPEG-2 Transport Packets for the LVDS transmission. Instead we use random data format which is multiple of 8 bits (1 byte) and the minimum burst of data that can be sent is 8 bits (10 bits after 8b/10b encoding).

When data is received on the LVDS lanes, the S-Band Transmitter (Data Sink side) removes the K28.5 symbols used for data stuffing and synchronization and prepares the MPEG-2 Transport Packets prior to the transmission to the DVB-S2 Modulator data path. The MPEG-2 Transport Packet is 188 bytes and the header is 4 bytes: x47 x1 x2 xCC.

Technical parameters summary:

- CLOCK rate – up to 160 MHz;
- Useful Data Rate – up to 130 Mbps;
- Encoding 8b/10b with K28.5 symbol for data stuffing;
- At least two consequent K28.5 symbols must be sent for synchronization at every 1880 bits;

- Transmission format is as per Figure B.8 of EN 50083-9 Annex B but the data is random and not mandatorily using MPEG-2 Transport Packets;
- The S-Band Transmitter removes K28.5 data stuffing symbols so that only the useful data is sent through the DVB-S2 Modulator;
- The S-Band Transmitter prepares the MPEG-2 Transport Packets prior to the DVB-S2 data path transmission.

6 DOWNLINK INTERFACE

The Downlink interface connects the S-Band Transmitter with the onboard S-Band Antenna and through the air interface to the Ground Station modem. Two layers of the OSI Model are distinguished here – Physical Layer 1 and Data Link Layer 2 (see Table 144).

Table 144: Communication Layers of the Downlink Interface

Layer	Interface From	Interface To	Description of Interface	Other Information
Data Link	S-Band Transmitter	GS modem	DVB-S2 MPEG-TS	See Chapter 6.2
Physical	S-Band Transmitter	S-Band Atenna	SMA female conector	See Chapter 6.1

6.1 Layer 1

Figure 10 shows the SMA 50 Ω female connector with normal polarity which interfaces the output RF signal. Its position allows easy assembling and routing of the coaxial cable to the S-Band Antenna. Depending on the order it can be in either straight or right-angled configuration.

The characteristic impedance of the coaxial cable connecting the antenna must be 50 Ω. The cable must be as short as possible for minimum insertion losses for the RF signal.

6.2 Layer 2

The Data Link Layer 2 of the so-called Downlink interface ensures the connectivity between the S-Band Transmitter and the Ground Station modem. The standard configuration of the EnduroSat's S-Band Transmitter module comes with DVB-S2 as per ETSI EN 302 307-1 V1.4.1.

According to the standard it supports:

- Modulation type: MODCOD (1 – 23)
- Filter Roll-Off factor: 0.35 / 0.25 / 0.2
- Frame Size: Normal FECFRAME, Short FECFRAME
- Pilot Signal: On / Off

Figure 10: SMA Connector

The payload data can be sent either from the LVDS interface or from the internal memory. In either case the data is encapsulated into Transport Stream User Packets - MPEG2. This data processing is done in real time when the module is in Transmit Mode. The MPEG2 packets have a constant length of 188 bytes (i.e. 188 x 8 bits). The first 4 bytes are intended for the header and the other 184 bytes - for the payload data (see Table 145). The header is fixed (i.e. constant PID), only the last 4 bits of byte 4 are cycling from 0x00 to 0x0F, this is the Continuity Counter. Please, contact EnduroSat if different PID is required.

Table 145: MPEG-TS User Packet

	M	PEG-TS USER PACKET		
Byte 1	Byte 2	Byte 3	Byte 4	Bytes 5 - 188
0x47	0x01	0x02	0x00 – 0x0F	Payload Data

In Transmit Mode the S-Band Transmitter starts sending NULL frames with a constant bit rate according to the transmission parameters previously set. While the module is in Transmit Mode, it sends NULL frames except for the periods of time when file/s is/are being transmitted from the internal memory or if real data is present on the LVDS interface (i.e. other than k28.5 symbols).

Every time the module is switched to Transmit Mode, there is a period of time called pretransmission delay during which only NULL frames are being sent. This feature is used to provide time for the Ground Station modem to lock (i.e. synchronize) the received signal, so that no real data is lost.

It is important to note that there is a significant difference between sending files from the internal memory and sending data through the LVDS interface. Sending payload data through the LVDS interface happens in real time. In this case the HOST must secure that the data rate on the LVDS lanes is always less than the radio channel's throughput. Otherwise, a huge amount of errors will be present within the received data at the GS modem. On the other hand, when data is sent from the internal memory, the S-Band Transmitter automatically keeps the data rate at which the file is being sent slightly below the radio channel throughput. This allows the HOST to change MODCOD and Symbol Rate on the fly without losing any data. This feature allows establishing Variable Coding and Modulation (VCM) downlink.

In both cases (LVDS or internal memory) the radio channel capacity is always higher than the data rate. This means that NULL frames are always present among the MPEG2 TS packets (data stuffing). In other words, in addition to the MPEG2 TS packets with payload data that are received by the Ground Station, NULL frames are also received.

Once the TS file has been received and recorded on the Ground Station, the NULL frames and the MPEG TS packet headers must be removed so that only the payload data is left. Thus, the original payload data is decapsulated and restored.