LS1P Protocol Specification

Table of Contents

Date (version)	Author	Comment
2013-03-23 (0.01)	K. Petrauskas	Initial revision.
2013-03-25 (0.02)	K. Petrauskas	Updated according to Simonas's comments. Some ARM commands defined.
2013-03-28 (0.03)	K. Petrauskas	ARM commands defined.
2013-03-29 (0.04)	K. Petrauskas	
2013-07-21 (0.05)	K. Petrauskas	Meaning of the address and the port fields redefined in DAT frames.
2013-07-23 (0.06)	K. Petrauskas	Arduino commands specified in more details.
2013-07-23 (0.07)	K. Petrauskas	Variable block size added for "Get buffer fragment (arm, port=0x2)"
2013-08-01 (0.08)	K. Petrauskas	Beacon management commands added.
2013-08-04 (0.09)	K. Petrauskas	Photo medatada defined.
2013-09-21 (0.10)	K. Petrauskas	EPS Commands defined.
2013-09-29 (0.11)	K. Petrauskas	Buffer ids updated. Arm commands pwr_allow_nm and pwr_state added.
2013-10-13 (0.12)	K. Petrauskas	Command frame signature algorithm described.

Table 1: Revision history.

1. Introduction

LS1P stands for "LituanicaSAT-1 Protocol". The protocol is designed to be used on top of the AX.25 Unordered (UI) frames [1].

The protocol is designed not using frame boundary markers or length indicators. Each LS1P frame should be sent in one AX.25 frame and each each AX.25 frame should have exactly one LS1P frame.

NOTE: All data is sent in little-endian.

2. Protocol frames

This section defines frame types, used in the LS1P protocol. Common usage of the frames as well as relationship between them are also provided here.

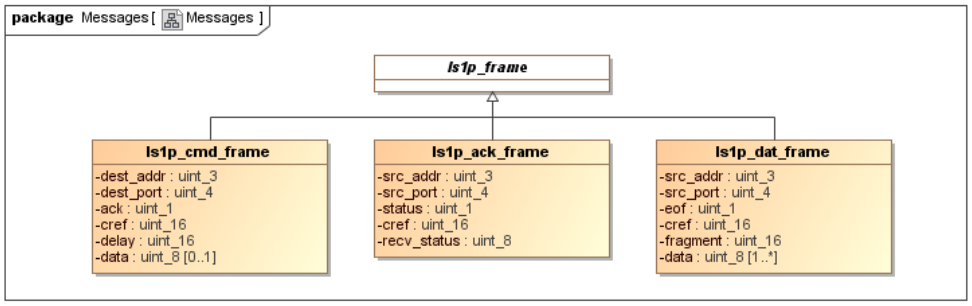

Figure 1: LS1P frame types.

Three types of frames are defined in the LS1P protocol:

1. `ls1p_cmd_frame` – command frame. This type of frames are sent from the ground station to the satellite.
2. `ls1p_ack_frame` – acknowledgement frame. Acknowledgement frames are sent in response to a command if the `ack` field was set to 1. Direction of these frames is always from the satellite to the ground station and length is fixed and equals to 4 octets.
3. `ls1p_dat_frame` – data frame is also sent from the satellite to the ground station in response to a command. This type of frame always carries payload data and is 6 or more octets in length.
4. `ls1p_tm_frame` – telemetry frames, that are periodically sent by the SAT without any request from the ground station (not shown in the diagram).

2.1. Common scenarios

Figure 4 shows relations between different types of frames in a typical communication scenario.

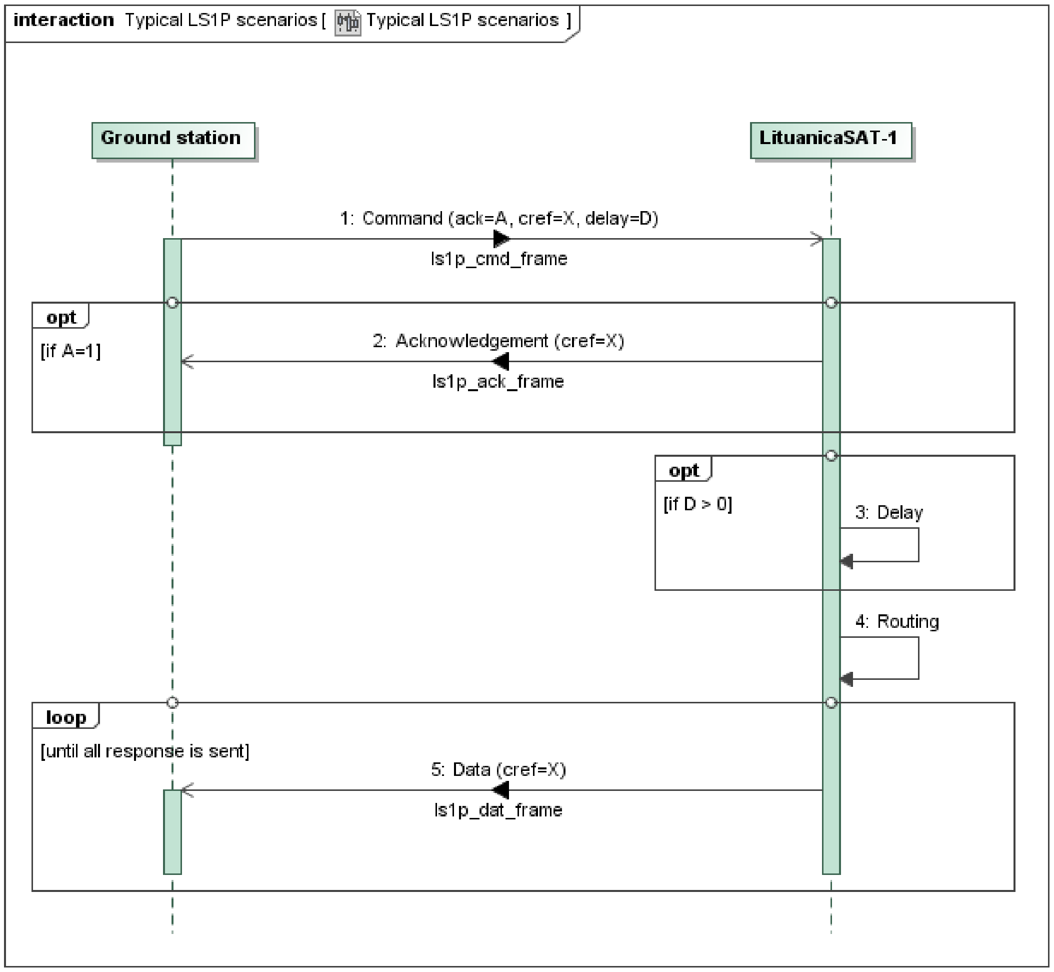
interaction Typical LS1P scenarios [Typical LS1P scenarios]
Ground station
LituanicaSAT-1
1: Command (ack=A, cref=X, delay=D)
ls1p_cmd_frame
opt
[if A=1]
2: Acknowledgement (cref=X)
ls1p_ack_frame
opt
[if D > 0]
3: Delay
4: Routing
loop
[until all response is sent]
5: Data (cref=X)
ls1p_dat_frame

Figure 2: Relation between protocol frame types in a typical communication scenario.

TODO: Add scenario with retransmission of missing frames in the ground station.

2.2. Common encoding rules

The following table lists all possible values for the address fields (`dest_addr` and `src_addr`). Ports are specific for each addressable node.

Address	Addressed Node	Comment
0b000 (0)	ARM	Main on-board processor
0b001 (1)	Arduino	Controller responsible for the beacon and the camera.
0b010 (2)	EPS	Energy supply subsystem.
0b011 (3)	GPS	Global positioning subsystem.
0b100 (4)	Helium-100	Main communication transceiver.
0b111 (7)	Ground	Ground station / MCS

Table 2: LS1P addresses.

All bytes are sent with most significant bit first. The same applies to bytes in a field.

3. Command frames

The following fields compose each command frame:

Field	Size	Description
`dst_addr`	3 bits	Destination address. See Table 2.
`dst_port`	4 bits	Port for the specific service, 4 bits (can also be called as a command code). Each addressable node have its own set of ports. They are listed and described bellow.
`ack`	1 bit	Indicates, if the acknowledgement should be sent upon reception of the command in the satellite's router but before the delay.
`cref`	16 bits	Command reference. Each command should have unique reference. The uniqueness is handled in the ground station. The number is auto-incremented with each command. The number overflows to 0.
`delay`	16 bits	If set to non-zero, satellite's router will delay execution of a command for number of seconds specified in this field.
`data`	N * 8 bits.	Length and contents of this field are specific for each command. For more details see description of particular command.

Table 3: Command frame structure.

Each command should be logged in the satellite's command log with exception of the following commands in the case of zero delay:

1. Ping – we will send a lot of ping commands and there is no need to have the command log filled with such entries.
2. Get buffer command – this command will be used to downlink the command buffer so it would make a loop to log such entries.

The following table lists several command examples. All possible commands are documented in the sections bellow.

Message	dest_addr	dest_port	ack	cref (eg.)	delay	data
Ping	0b000 (arm)	0x0	1 (true)	0xE14A	0x0000	N/A
Kill command	0b000 (arm)	0x1	0 (false)	0xE14B	0x0000	0xE14B
Get buffer fragment	0b000 (arm)	0x2	0 (false)	0xE14C	0x0000	0x00 0x7F 0x0002 0x0027
Get real-time telemetry	0b000 (arm)	0x3	0 (false)	0xE14D	0x0000	N/A
Set job period	0b000 (arm)	0x4	1 (true)	0xE14E	0x0050	0x01 0x0100
Allow PWR nominal reentry	0b000 (arm)	0x5				0x01
Set PWR mode manualy	0b000 (arm)	0x6				0x01
Terminate SCI mode	0b000 (arm)	0x7				
Start FM repeater	0b000 (arm)	0x8				0xFFFF (duration in seconds)
Perform multi-command	0b000 (arm)	0xF	1 (true)	0xE150	0x0000	0x05 0x0604ED00000002001B
Take a photo	0b001 (ard)	0x0	1 (true)	0xE151	0x0101	TODO
Get photo metadata	0b001 (ard)	0x1	0 (false)	0xE152	0x0000	TODO
Get photo data	0b001 (ard)	0x2	0 (false)	0xE153	0x0000	TODO
Set beacon status	0b001 (ard)	0x3	1 (true)	0xE153	0x0000	0x00 (off)
Perform EPS command	0b010 (eps)	0x0	1 (true)	0xE154	0x0111	Opaque
Perform GPS binary cmd.	0b011 (gps)	0x0	1 (true)	0xE155	0x1000	Opaque
Perform GPS NMEA cmd.	0b011 (gps)	0x1	1 (true)	0xE156	0x1000	Opaque
Perform Helium command	0b100 (he)	0x0	1 (true)	0xE157	0x1000	Opaque
He: set TM broadcast period	0b100 (he)	0x0	1 (true)	0xE158	0x0040	TODO

Table 4: Example messages.

3.1. Command signature

Signing (in the MCS) is done as follows:

1. Calculate 16-bit checksum using Fletcher's 8-bit algorithm on the entire LS1P frame.
2. Calculate a signature by doing bitwise XOR of the checksum with a 16-bit password.
3. Replace first 16 bits of the frame with 32 bits calculated by merging bit by bit the signature with the first 16 bits of the data. i.e. the resulting 32 bits are "S15, F15, S14, F14, … S00, F00" where S stands for the bit from the signature and F stands for the bit from the frame. 15 is the MSB and 00 stands for the LSB.
4. First 32 bits of the frame are sent as separate bytes, i.e. they are NOT a single integer in little-endian.

Signature validation is performed as follows:

1. Take first 32 bits of the frame and extract the 16 bit "provided signature" and the first 16 bits of the frame data (see step 3 in the signing procedure above).
2. Calculate checksum over the frame data including first 16 bits extracted from the first 32 bits. The "provided signature" should be excluded here.
3. Calculate a signature by performing bitwise XOR of the checksum with the password.
4. Compare the "provided signature" with the calculated signature. Drop the frame if they are not equal.

3.2. Commands for ARM

3.2.1. Ping (arm, port=0x0)

Description: This command is used to check a connectivity with the satellite. The ping command can also be used as a ground station heartbeat message.

Data: No data is needed for this command. A frame is always exactly 5 octets long.

Behaviour: No action is needed for this command. The acknowledgement will be sent by the router if the `ack` bit is set to 1.

Examples in binary for ping with acknowledgement (A) and ping with no acknowledgement (B):

```
A: 00000001 11100001 01001010 00000000 00000000
B: 00000000 11100001 01001010 00000000 00000000
   aaappppk rrrrrrrr rrrrrrrr dddddddd dddddddd
```

where "a" stands for the `addr`, "p" – `port`, "c" – `ack`, "r" – `cref` and "d" for the `delay`.

3.2.2. Kill command (arm, port=0x1)

Description: this command is issued when a previously scheduled message should be terminated.

Data: Single parameter is needed for this command:

1. `cref_to_kill` – 16 bit command reference of the command to be terminated if not executed yet.

Behaviour:

Scheduled task is terminated if found and is not executed yet. The command log should be updated accordingly (set command status to killed) for the command with `cref=cref_to_kill`. The command log should be updated only if command was found, it was pending and killed successfully.

Example in binary to kill command with `cref=3`:

```
A: 00000101 1110000101001100 0000000000000000 0000000000000011
   aaappppk rrrrrrrrrrrrrrrr dddddddddddddddd LLLLLLLLLLLLLLLL
```

Here “a”, “p”, “k”, “r” and “d” has the same meaning as in previous section. Additionally “L” stands for the `cref_to_kill` parameter.

3.2.3. Get buffer fragment (arm, port=0x2)

Description: Initiates transmission of the specified buffer (the telemetry archive, the command log, GPS binary or NMEA output) fragment from the satellite to the ground station. The requested buffer data is sent in blocks of raw data. The data will be collected and analysed in the ground station. Multiple data frames of type “Raw data frame” (section 5.2) can be sent in response to this command.

Data: Four parameters are required by this command:

1. `buffer_id` – 8 bit number indicating particular buffer, who's information should be sent to the ground station. Valid buffer ids are listed in the table bellow.
2. `block_size` – 8 bit number indicating block size.
3. `from_block` – 16 bit number indicating first block of the archive to send (inclusive).
4. `till_block` – 16 bit number indicating block till which the archive should be sent. This parameter is exclusive.

buffer_id	Buffer name	Comments
`0x00`	Command log	
`0x01`	Housekeeping data archive	
`0x02`	Attitude data archive	

Table 5: Possible values for the buffer_id parameter.

Behaviour: Satellite reads the specified cyclic buffer, splits the specified fragment it into multiple LS1P frames and sends them to the ground station. Each frame sent to the ground station has unique `fragment`, starting at 0 and increased by 1 with each subsequent frame. Last frame must have `eof` bit set to 1.

If the command is issued with arguments `buffer_id=1`, `from_block=2` and `till_block=`**5**, blocks with numbers 2, 3 and **4** should be sent to the ground station.

Example in binary for the "Get buffer fragment" messages to get 2th to 4th block of the telemetry archive without an acknowledgement (A) is provided bellow:

```
A: 00000100 1110000101001101 0000000000000000 00000001 01111111 0000000000000010 0000000000000101
   aaappppk rrrrrrrrrrrrrrrr dddddddddddddddd bbbbbbbb ssssssss ffffffffffffffff tttttttttttttttt
```

Here "a", "p", "k", "r" and "d" has the same meaning as in section 3.2.1. Additionally "b" stands for the `buffer_id`, "s" - for block size, "f" - `from_block` and "t" for the `till_block`.

3.2.4. Get real-time telemetry (arm, port=0x3)

Description: This command collects all the sensors telemetry upon reception of the command and sends the data to the ground station.

Data: This command has no parameters.

Behaviour: The data is collected the same way, as it is done for the telemetry archive. The data is returned via single "Telemetry data frame" (see section 6 for payload structure, excluding frame header). The returned frame structure matches with the archive record structure.

Example: The following is an example of the "Get real-time telemetry" command without an acknowledgement in binary:

```
A: 00000110 1110000101001110 0000000000000000
   aaappppk rrrrrrrrrrrrrrrr dddddddddddddddd
```

Here "a", "p", "k", "r" and "d" has the same meaning as in section 3.2.1.

3.2.5. Set job period (arm, port=0x4)

Description: Sets new period for the specified job.

Data: Two parameters are needed for this command:

1. `job_id` – 8 bit unsigned integer identifying single periodic job.
2. `job_interval` – 16 bit unsigned integer specifying interval in seconds between telemetry updates.

job_id	Job name	Comments
`0x00`	Telemetry broadcast update	
`0x01`	Housekeeping telemetry collection	
0x02	Attitude telemetry collection	
0x03	GPS telemetry collection	

Behaviour: This command should only update the scheduler configuration. No other actions should be needed.

Example in binary for the “Set job period” message for updating the telemetry broadcast buffer every 5 seconds (A) and “Housekeeping telemetry collection” every 2 seconds (B):

```
A: 00001001 1110000101001101 0000000000000000 00000000 0000000000000101
B: 00001001 1110000101001101 0000000000000000 00000001 0000000000000010
   aaappppk rrrrrrrrrrrrrrrr dddddddddddddddd jjjjjjjj iiiiiiiiiiiiiiii
```

Here “a”, “p”, “k”, “r” and “d” has the same meaning as in section 3.2.1. Additionally “j” stands for `job_id` and “i” for the job`_interval`.

3.2.6. Allow PWR nominal reentry (arm, port=0x5)

Description: Allow reentry to the nominal PWR mode after falling to the safe mode.

Data: Single parameter is accepted by this command:

1. `allow` – 8 bit unsigned integer, 0 – false (disallow), 1 – true (allow).

Behaviour: More details needed.

3.2.7. Set PWR state manually (arm, port=0x6)

Description: Sets PWR mode manually.

Data: Single parameter is accepted by this command:

1. `pwr_mode` – 8 bit unsigned integer, 0 – auto, 1 – safe, 2 – nominal.

Behaviour: More details needed.

3.2.8. Terminate SCI mode (arm, port=0x7)

Description: Terminate sientific mode (fm repeater, phodo download process etc).

Data: No parameters are needed for this command.:

Behaviour: More details needed.

3.2.9. Start FM Repeater (arm, port=0x8)

Description: Start FM repeater for the specified period of time.

Data: Single parameter is accepted by this command:

1. `duration` – 32 bit unsigned integer, operation duration in seconds.

Behaviour: More details needed.

3.2.10. Perform multi-command (arm, port=0xF)

Description: Schedule execution of multiple LS1P commands at once.

Data: Argument for this command is a list of LS1P command frames each prefixed its length, i.e.:

1. `subcommand_count` – 8 bit unsigned integer indicating number of sub-commands in this frame.
2. List of records with the following structure:
    1. `subcommand_length[i]` – 8 bit unsigned integer indicating length of the i'th subcommand,
    2. `subcommand_data[i]` – byte array of length `subcommand_length[i] * 8` bits. Its internal structure is exactly the same as for typical `ls1p_cmd_frame`.

Behaviour: Delay and acknowledgement handling is the same, as for other commands. i.e. acknowledgement should be sent upon reception of the multi command if `ack` bit is set. Processing of the multi-command as a whole should be delayed, if the `delay` field is non-zero. The same applies for the command logging – the multi-command should be logged as a whole.

Execution of the multi-command should be performed as follows:

1. Data field is split into sub-commands.
2. Each sub-command is validated according to general router's validation rules.
3. If all commands pass the validation, they are submitted back to the router/scheduler in the same

way as usual commands coming from the Helium.

Example: The following is an example in hex of a multi-command carrying two ping commands (A):

```
A: 1F25CD0000 02 05 0125CE0000 05 0125CF0000
   ccrrrrdddd CC LL SSSSSSSSSS LL SSSSSSSSSS
   (Header  )       (1st ping )    (2nd ping )
```

Where "c" stands for control byte (`src_addr + src_port + eof`), "r" - the "Perform multi-command" `cref`, "d" - the "Perform multi-command" command execution `delay`. Additionally:

- "C" stands for a sub-command count (2 in this case);
- "L" is length of the following sub-command.
- "S" is the data of particular sub-command (both are "Ping" in this example).

Please note, that unique `cref`s are provided for all three commands in this example.

3.3. Commands for Arduino

3.3.1. Take a photo (arduino, port=0x0)

Description: Takes new photo.

Data: Two parameters are required by this command:

1. `cref` – 16 bit number indicating "correlation reference".
2. `res` – 8 bit number, indicating resolution (code).

Behaviour:

1. ARM scheduler receives the command, sends an acknowledgement and schedules the command.
2. After the specified delay, ARM scheduler sends command to Arduino.
3. Arduino takes a photo, writes it to SD card as well as its metadata: timestamp, cref, photo size.
4. Done. Following actions will be initiated by other commands from the GS.

Example: …

3.3.2. Get photo metadata (arduino, port=0x1)

Description: Downlinks a metadata of a photo previously produced by the Arduino "Take Photo" command.

Data: no parameters are required for this command.

Behaviour:

1. ARM receives the command, sends acknowledgement, and sends it to Arduino after specified delay.
2. Arduino reads photo metadata (from a separate file in the SD card), packs it to a LS1P frame and sends to ARM.
3. ARM forwards the frame to the Helium.

Example: …

3.3.3. Get photo data (arduino, port=0x2)

Description: Downlinks a photo previously produced by the Arduino "Take Photo" command.

Data: The following parameters are required for the command.

1. `block_size` – 8 bit number indicating block size.
2. `from_block` – 16 bit number indicating first block of the archive to send (inclusive).
3. `till_block` – 16 bit number indicating block till which the archive should be sent. This parameter is exclusive.

Behaviour: …

1. ARM receives the command, sends acknowledgement, and sends it to Arduino after specified delay.
2. Arduino reads photo from byte (`block_size*from_block`) till (`block_size*(till_block-1)`). Splits it to (`till_block - from_block`) frames and sends them to ARM one by one.
3. ARM forwards all frames to He.

Example: …

3.3.4. Set beacon status (arduino, port=0x3)

Description: Turns the beacon on or off.

Data: Single parameter is needed for this command:

1. `beacon_status` – 8 bit number, 0 – off, 1 – on.

Behaviour: See description.

Example: N/A.

3.4. Commands for EPS

3.4.1. Set channel status (eps, port=0x0)

Description: Switches power in the specified EPS channel on or off.

Data: The following parameters are needed:

1. `channel` – 8 bit number, 0 – 5V1, 1 – 5V2, 2 – 5V3, 3 – 3.3V1, 4 – 3.3V2, 5 – 3.3V3.
2. `status` – 8 bit number, 0 – off, 1 – on.

Behaviour: Sends the corresponding command to the EPS.

Example: N/A

3.5. Commands for GPS – not needed

3.5.1. MNP-binary port (gps, port=0x0)

Description: …

Data: ...

Behaviour: ...

Example: …

3.5.2. NMEA port (gps, port=0x1)

Description: …

Data: ...

Behaviour: ...

Example: …

3.6. Commands for Helium-100 (helium, port=0x0)

Description: …

Data: ...

Behaviour: ...

Example: …

3.6.1. Set TM broadcast period – not needed.

Description: Sets new period for telemetry broadcast via Helium's beacon.

Data: Single parameter is needed for this command:

2. `tmb_interval` – 8 bit unsigned integer specifying interval in seconds between telemetry broadcast sessions.

Behaviour: ...

Example: …

4. Acknowledgement frames

Acknowledgement frames are frames sent by the satellite to the ground station in response of a command, who's `ack` field was set to 1.

Field	Size	Description
`dst_addr`	3 bits	Always 7 – ground station. See Table 2.
`dst_port`	4 bits	Always 0 – acknowledgement frame.
`status`	1 bit	Indicates, if the command is received by the satellite successfully. 1 indicates success, and 0 – failure.
`cref`	16 bits	Command reference to which the acknowledgement is being sent.
`recv_status`	8 bits	Return code can be used to report specific processing errors. 0 should be used for positive acknowledgements.

Table 6: Acknowledgement frame structure

5. Data frames

The following fields compose each command frame:

Field	Size	Description
`dst_addr`	3 bits	Always 7 – ground station. See Table 2.
`dst_port`	4 bits	Always 1 – data frame.
`eof`	1 bit	Indicates, if this frame is the last one in the stream. If set to 1, no more frames will be sent in response to the command with the specified `cref`.
`cref`	16 bits	Command reference to which the response is being sent.
`fragment`	16 bits	Fragment number in the response stream. This field is auto-incremented by the satellite and is started from 0 for each command.
`data`	N * 8 bits	Length and contents of this field are specific for data frame type (`addr/port`). For more details see description of the particular frame.

Table 7: Data frame structure

5.1. Photo metadata

This section describes structure of the data field for data frames, returned in response to arduino, photo_meta command.

Field	Size	Description
`cref`	16 bits	Photo CREF, that was passed with the take_photo command.
`size`	16 bits	Size of the photo in bytes.

Table 8: Photo metadata structure

5.2. Raw data frame

This section specifies response structure to the ARM's “Get command log” command. The response data is sent using fixed-length records with record structure defined in Table 9. Each data frame should carry as much data records as possible. Each frame can carry non-integral number of log records (records can be split to different LS1P frames.

5.3. Command log data

Field	Length	Description
`cref`	16 bits	Reference ID of the logged command.
`recv_time`	32 bits??	Command reception time in the satellite's time, in seconds?
`recv_status`	8 bits	Command reception status (the same as in the acknowledgement frame).
`exec_time`	32 bits??	Command reception time in the satellite's time, in seconds?
`exec_status`	8 bits.	Command execution status. 0 should be used to indicate success.

Table 9: Command log entry structure

Examples: header in binary and data in hex. The following are two frames forming a response to single “Get command log” command with `cref=0xE14B`. Please note that the B message has `eof` flag set to 1 indicating the last frame in the stream.

```
A: 02E14B0000 E14AFFFFFFFF00FFFFFFFF00 E14BFFFFFFFF00FFFFFFFF01 E14CFFFFFFFF00FFFFFFFF01
B: 03E14B0000 E1FAFFFFFFFF00FFFFFFFF00 E1FBFFFFFFFF00FFFFFFFF01
   ccrrrrdddd RRRRTTTTTTTTSSIIIIIIIIAA RRRRTTTTTTTTSSIIIIIIIIAA RRRRTTTTTTTTSSIIIIIIIIAA
   (Header  ) (1'st log rec. in frame) (2'nd log rec. in frame) (3'rd log rec. in frame)
```

Where “c” stands for control byte (`src_addr + src_port + eof`), “r” - the “Get command log” `cref`, “d” - the “Get command log” command execution `delay`, “R” - `cref` of the logged command, “T” - `recv_time`, “S” - `recv_status`, “I” - `exec_time`, “A” - `exec_status`.

5.4. Telemetry archive data

TODO

5.5. Opaque data (EPS, GPS, Helium)

TODO

6. Telemetry frame

The following fields compose each telemetry frame:

Field	Size	Description
`dst_addr`	3 bits	Always 7 – ground station. See Table 2.
`dst_port`	4 bits	Always 2 – telemetry frame.
unused	1 bit	Always 0.
`timestamp`		
`Field1...`		
`Field2...`		
`Field3...`		

Table 10: Telemetry frame structure

Bibliography

1: , AX.25 Link Access Protocol for Amateur Packet Radio , 1998