# ANTARIS INC

# UHF SatOS GS-APP Open Source

---

## User guide document

9/17/2024

# Table of Index

# 1 Tables and Figures

# 2 Overview

This document captures the details of SatOS SW open sourced by Antaris for UHF specific SDR interfaces and the GS-APP used for command and control.

# 3 SatOS UHF and GS-APP overall architecture

## 3.1 SatOS UHF and GS-APP E2E Architecture block diagram

The overall architecture for SatOS UHF integration with GS segment is represented in the below block diagram

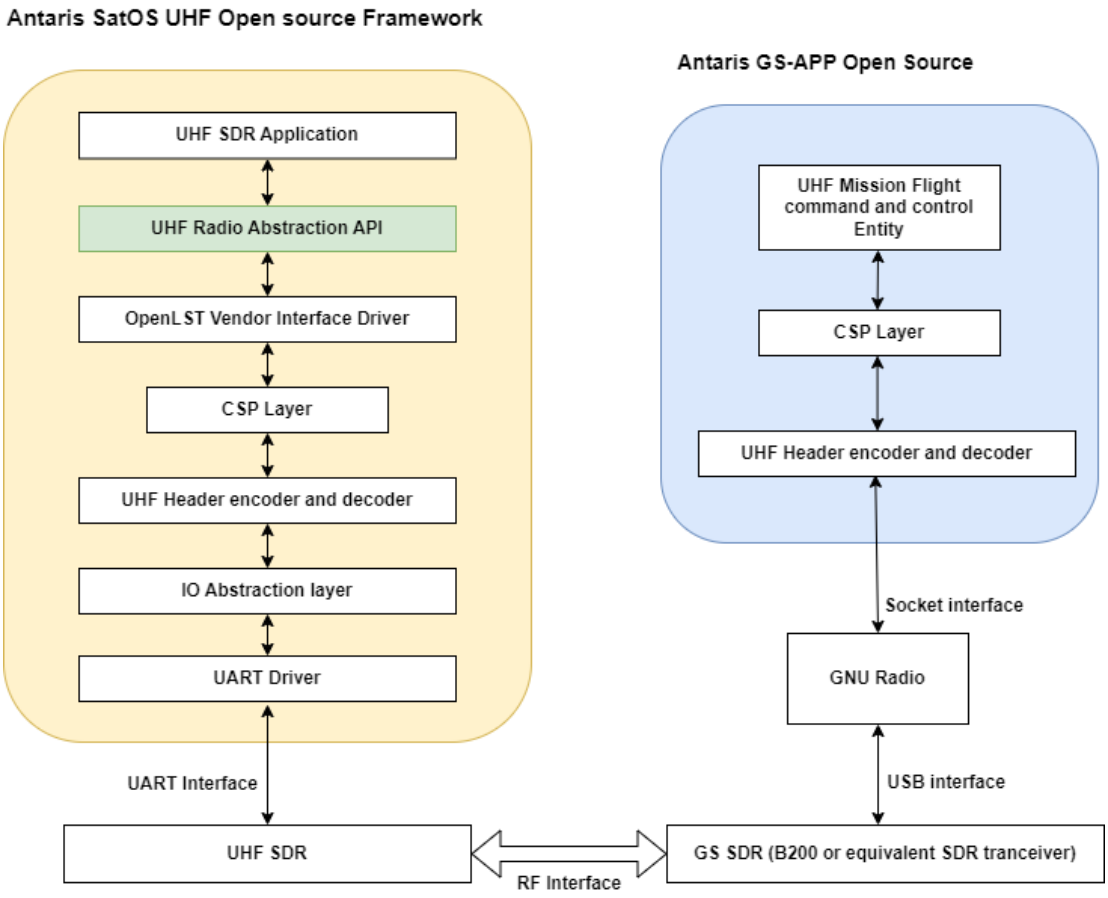


*Figure 1: SatOS UHF GS E2E architecture*

## 3.2 Reference platform used for UHF HW Integration

The SatOS UHF Open source SW support integration of UHF HW using both Linux based and Embedded RTOS based platform. The integration platforms are detailed in the below sections. The SatOS can be used for both the platforms by switching the SW compilation.

### 3.2.1 Embedded Processor

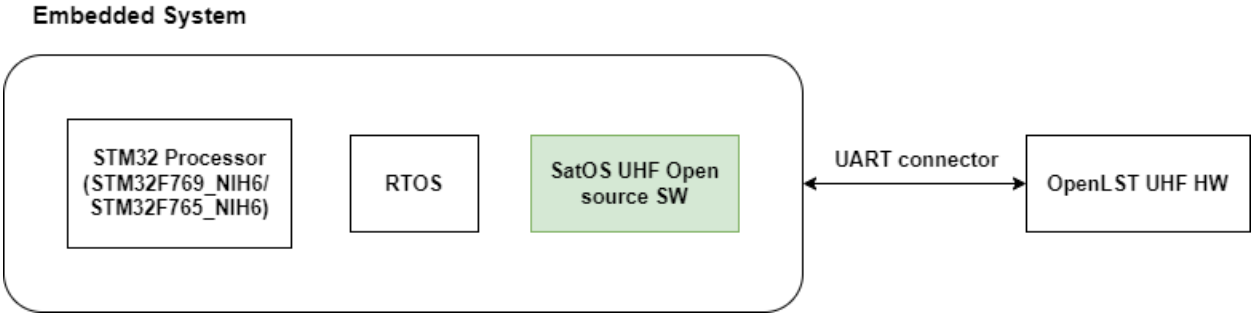


*Figure 2: UHF Integration platform with Embedded System*

### 3.2.2 Linux Subsystem

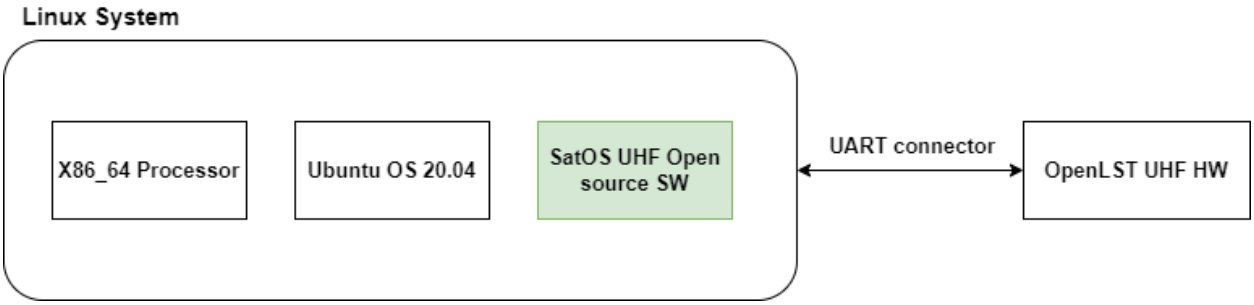


*Figure 3: UHF Integration platform with Linux System*

# 4 SatOS UHF Open Source SW

## 4.1 SatOS UHF Open Source SW supported features

This open-source SW supports the following aspects for UHF OpenLST integration

1. UHF OpenLST communication through UART from Linux Platform
2. UHF OpenLST communication through UART from STM32 target Platform
3. UHF internal command access and response through GS
4. UHF Periodic telemetry read and reporting
5. UHF periodic beacon transmission with SW customisable payload
6. UHF configuration APIs (These APIs can be used with respective support in OpenLST firmware on top of open-sourced OpenLST SW)
7. Optional Backdoor access from SatOS UHF SW to locally test the UHF HW with the GS setup before launch

## 4.2 UHF TT&C format

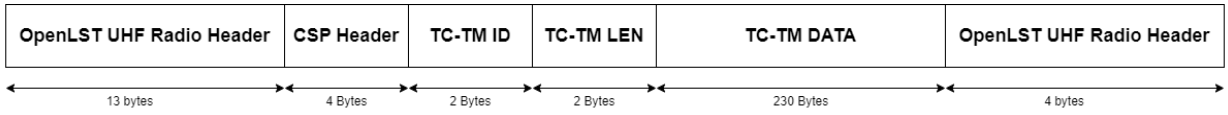


*Figure 4: UHF TT&C payload data format*

## 4.3 OpenLST Open Source supported commands

The supported UHF commands as per open sourced OpenLST SW is listed in the below table.

*Table 1: OpenLST commands support list*

| Command ID | TC/TM | Command Name | Command Description |
|---|---|---|---|
| 0x00 | TC | Bootloader Message PING | PING message that can be sent before initiating the file transfer of Bootload message |
| 0x01 | TM | Bootloader Message ACK | to send ACK message for command operation success of bootload commands like Bootloader message write/erase (0x02/0x0C) |
| 0x02 | TC | Bootloader Message write page | Command to initiate flash write operation of the bootload message |
| 0x0C | TC | Bootloader Message erase | Command to ease bootload message from the flash memory location |
| 0x0F | TM | Bootloader Message NACK | to send NACK message for command operation success of bootload commands like Bootloader message write/erase (0x02/0x0C) |
| 0x10 | TM | Common Message ACK | To positively ACK |
| 0x11 | TM | Common Message ASCII | To send the debug trace information |
| 0x12 | TC | radio_msg_reboot | To reboot watchdog timer |
| 0x13 | TC | radio_msg_get_time | To get RTC Time |
| 0x14 | TC & TM | radio_msg_set_time | To set RTC Time |
| 0x15 | TC | radio_msg_ranging | Configuring the radio mode to ranging TX (It is given that they support time-of-light ranging for coarse location and trajectory estimation) |
| 0x16 | TM | radio_msg_ranging_ack | ACK for ranging command |
| 0x17 | TC | radio_msg_get_telem | Command to request telemetry |
| 0x18 | TM | radio_msg_telem | Command to reply with telemetry payload information |
| 0x19 | TC | radio_msg_get_callsign | The radio can store a callsign in the persistent memory using the set_callsign command. The callsign is meant to ease compliance with US CFR Part 97 (or compliance with similar laws in other countries). It is up to the user to ensure compliance with local laws. This command is used to get the callsign value |
| 0x1a | TC | radio_msg_set_callsign | This command is used to set the callsign value |
| 0x1b | TM | radio_msg_callsign | This command is used to carry the callsign value in response to the get message |
| 0xFF | TM | Common Message NACK | To reply with a NACK |

The configuration APIs are implemented to support above configuration commands. These commands enhance the configurability of UHF modem during flight operation.

The debug commands provide additional insight of UHF SDR module to the On-Board controller or SDR controller.

## 4.4 SatOS UHF Open-Source configurations

### 4.4.1 CSP Library configuration

The CSP library has the following configuration

The CSP address of SatOS node is configured as 11

The CSP address for UHF internal command access is configured as 14

The CSP address for UHF space data transfer is configured as 15

The CSP port address for UHF UL data transfer is 25

The CSP port address for UHF DL data transfer is 35

## 4.5 SatOS UHF Open Source supported APIs

The APIs exposed in the SatOS open source can be categorized broadly as follows

1. APIs for UHF UART RX and TX with UHF specific command headers
2. APIs for UHF UART RX and TX with UHF specific command headers and CSP header
3. APIs for sending and processing the UHF module specific commands
4. APIs for UHF command configuration
5. APIs for forming UHF beacon data

The list of UHF APIs is further detailed in the upcoming section.

### 4.5.1 APIs for UHF UART RX and TX with UHF specific command headers

#### 4.5.1.1 void uhf_pack_uart_frame(csp_packet_t *packet, uint8_t *frame_buff)

In this function, the CSP packet addressed to node address 14 will be processed and the UART command header will be added as per the UHF OpenLST UART radio header format as mentioned in Appendix section.

#### 4.5.1.2 void uhf_unpack_uart_frame(csp_packet_t *packet, uint8_t *frame_buff)

The response commands from the UHF module will be processed in this function. It decodes the UHF headers and sent the received data with command ID information to the UHF controller.

### 4.5.2 APIs for UHF UART RX and TX with UHF specific command headers and CSP header

#### 4.5.2.1 void uhf_pack_uart_csp_frame(csp_packet_t *packet, uint8_t *frame_buff)

The function is used to encode the DL data with CSP header and UHF header. The encoded data will be sent to UART driver for transmission to UHF RF module.

#### 4.5.2.2 void uhf_unpack_uart_csp_frame(csp_packet_t *packet, uint8_t *frame_buff)

This function is used to decode the UHF data received and transfers the unpacked data to the command processing entity for further processing.

### 4.5.3 APIs for sending and processing the UHF module specific commands

#### 4.5.3.1 void uhf_upd_send_uart_cmd(uint8_t cmd_id);

This function is used to update the command payload based on UHF command ID before sending to the UHF header encoding entity.

#### 4.5.3.2 uint8_t uhf_proc_uart_cmd_rsp(uint8_t *pld);

This function extracts the command payload based on UHF command ID and perform the actions based on the received response or indication in UHF controller or sends relevant message or TM to GS.

### 4.5.4 APIs for forming UHF beacon data

#### 4.5.4.1 uint8 comms_uhf_beacon_data(void);

This function is used for packing the UHF beacon payload for periodic Beacon transmission from SatOS. The periodicity of Beacon is configurable through separate telecommand.

## 4.6 Periodic Beacon Feature

The SatOS UHF open source SW has the periodic Beacon reporting feature which is controlled by below mentioned timers. These timers are configurable through a Telecommand ID **0x101 UHF_BEACON_TM_TMR_CFG.**

*Table 2: UHF Beacon configurations*

| Sl.No | SDR Timers/Counters | Description | Value | Unit |
|---|---|---|---|---|
| 1 | UHF Beacon Enable timer | UHF Beacon transmission start time before initiating the first Beacon transmission | 5 | Minutes |
| 2 | UHF Beacon periodicity timer | Beacon transmission periodicity for UHF. In this periodicity unique beacon data will be sent. | 60 | Seconds |
| 3 | UHF Beacon repetition counter | Same UHF beacon data will be repeated for this counter value | 3 | N/A |
| 4 | UHF Beacon repetition interval | UHF beacon will be repeated in this interval to improve the decode probability in the GS receiver | 2 | Seconds |
| 5 | UHF TM timer | Duration in which the TM will be reported | 120 | Seconds |

The beacon payload is configured as mentioned below.

| | |
|---|---|
| Hello Earth | UTC Timestamp |

*Figure 5: UHF Periodic Beacon payload*

*Table 3: UHF Beacon Data Structure*

| Parameters | Type | Bytes |
|---|---|---|
| Beacon Message | Uint8[12] | 12 |
| UTC Timestamp | Uint64 | 8 |

## 4.7 Periodic TM reporting Feature

In the configured TM periodicity as mentioned in Table 2: UHF Beacon configurations, the UHF telemetry will be read from UHF HW and the same will be reported via TM to GS.

The TM payload that will be reported to ground is mentioned in below table with sample values

*Table 4: UHF TM payload information*

| UHF TM Parameter | Value |
|---|---|
| up time | 561 |
| uart0 rx count | 0 |
| uart1 rx count | 60 |
| rx mode | 0 |
| tx mode | 0 |
| adc | 1380 |
| adc | 1914 |
| adc | 2047 |
| adc | 4086 |
| adc | 2047 |
| adc | 2047 |
| adc | 2047 |
| adc | 2047 |
| adc | 1379 |
| adc | 1825 |
| last rssi | 4.29E+09 |
| last lqi | 0 |
| last freq set | 0 |
| Pkt sent | 146 |
| cs count | 299 |
| Pkt good | 0 |
| Pkt rejected checksum | 0 |
| Pkt rejected reserved | 0 |
| Pkt rejected other | 0 |
| reserved0 | 1 |

# 5 GS-APP application

The SatOS UHF software can be integrated with the GS-APP as mentioned the overall architecture section 3. The current release supports TCP socket for the GS-APP.

The GS-APP has the following sub entities

1. UHF frame encoder and decoder
2. CSP library

3. Command and control interface APIs

## 5.1 UHF frame encoder and decoder

The UHF frame encoder and decoder is implemented in the GS application to transmit TC and receive TM to and from UHF radio module. The frame format implemented in the GS is as per section 6.1.1.1 and 6.1.2.2.

## 5.2 CSP library

The CSP library routes the data to and from UHF encoder and decoder from and to UHF command and control interfaces.

## 5.3 Command and control interface APIs

The entity has the CLI interface-based command interfacing functions for command and response handling to and from UHF SDR on the satellite.

## 5.4 Backdoor socket access

This GS-APP is also available for back door access of Sat-OS, where the telecommand will flow through a backdoor TCP socket connection which can be used for HWIL testing or ground-based integration testing with UHF HW. When the backdoor socket is enabled through SatOS compilation, the SatOS will be listening to a TCP socket for receiving any telecommand similar to the command interface from GS via UHF HW. The Sat-OS will send the telemetry response to both back door socket and to UHF DL.

# 6 Appendix

## 6.1 UHF Command Header formats

### 6.1.1 UHF DL – Satellite to GS

#### 6.1.1.1 SatOS to UHF Radio in UART

**OBC to UHF Radio Module**

| Byte Order | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9-BN |
|---|---|---|---|---|---|---|---|---|---|---|
| Field Type | SYNC1 | SYNC2 | Length | HWID | | Sequence Number | | Dest | CMD No. | Data |
| Possible Values | 0x22 | 0x69 | 7 to 251 | 0x00 - 0xFFFF | | 0x00 - 0xFFFF | | 0-255 | 0-255 | Data_ByteStream |

#### 6.1.1.2 UHF Radio to GS on Air

**UHF Radio Module to GS**

| Byte Order | B0 - B3 | B4 | B5 | B6 | B7 | B8 | B9 | B10 | B11 | B12 | B13 | B14 - B N-4 | B N-3 | B N-2 | B N-1 | B N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Field Type | Preamble | SYNC1 | SYNC2 | SYNC3 | SYNC4 | Length | Flags | Sequence Number | | Dest ID | CMD No. | Data | HWID LSB | HWID MSB | CRC LSB | CRC MSB |
| Possible Values | 0xAA | 0xD3 | 0x91 | 0xD3 | 0x91 | 11 to 255 | 0 or 1 | 0x00 - 0xFFFF | | 0 to 255 | 0 to 255 | Data_ByteStream | 0x00 - 0xFFFF | | 0x00 - 0xFFFF | |

### 6.1.2 UHF UL – GS to Satellite

#### 6.1.2.1 GS to UHF Radio on Air

**GS to UHF Radio Module**

| Byte Order | B0 - B3 | B4 | B5 | B6 | B7 | B8 | B9 | B10 | B11 | B12 | B13 | B14 - B N-4 | B N-3 | B N-2 | B N-1 | B N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Field Type | Preamble | SYNC1 | SYNC2 | SYNC3 | SYNC4 | Length | Flags | Sequence Number | | Dest ID | CMD No. | Data | HWID LSB | HWID MSB | CRC LSB | CRC MSB |
| Possible Values | 0xAA | 0xD3 | 0x91 | 0xD3 | 0x91 | 11 to 255 | 0 or 1 | 0x00 - 0xFFFF | | 0 to 255 | 0 to 255 | Data_ByteStream | 0x00 - 0xFFFF | | 0x00 - 0xFFFF | |

#### 6.1.2.2 UHF Radio to SatOS in UART

| UHF Radio Module to OBC | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Byte Order | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9-BN |
| Field Type | SYNC1 | SYNC2 | Length | HWID | | Sequence Number | | Dest | CMD No. | Data |
| Possible Values | 0x22 | 0x69 | 7 to 251 | 0x00 - 0xFFFF | | 0x00 - 0xFFFF | | 0 to 255 | 0 to 255 | Data_ByteStream |

## 6.2 UHF OpenLST HW configuration

The default configuration of OpenLST can be referred under https://github.com/OpenLST/openlst/blob/master/open-lst/USERS_GUIDE.md#specifications

Changing the UHF RF parameters is possible through UHF firmware update after changing the static configurations under https://github.com/OpenLST/openlst/blob/master/open-lst/common/board_defaults.c

The UHF firmware update procedure are detailed under https://github.com/OpenLST/openlst/blob/master/open-lst/USERS_GUIDE.md

## 6.3 SatOS Compilation Steps

### 6.3.1 Target Compilation for Discovery / Stack Board

1. Open the file located at `SatOS_HAL/Makefile.def` and ensure that the macro `ENVIRONMENT=1` is set.
2. In the same `Makefile.def` file, set `BOARD=STM32_DISCO_F769NI` for the discovery board or `BOARD=GEN1` for the stack board.
3. Run the command `make clean all` from the `SatOS_HAL` folder.
4. After a successful compilation, the ELF file will be generated in the `SatOS_HAL/bin/` folder with the name `antaris_exo_core.elf`.
5. Update the IP and port values in the JSON file located at `SatOS_HAL/bin/obc_soc_cfg.json`. Under the tag `OBC<->UHF`, set the IP value to `192.168.1.111` and the port value to `6767`.

### 6.3.2 Linux Compilation to Generate EXE

1. Open the file located at `SatOS_HAL/Makefile.def` and ensure that the macro `ENVIRONMENT=0` is set.
2. After a successful compilation, the EXE file will be generated in the `SatOS_HAL/bin/` folder with the name `antaris_exo_core.exe`.
3. Specify any local IP and port values in the JSON file located at `SatOS_HAL/bin/obc_soc_cfg.json`.
4. If using the UHF hardware module, ensure it is connected to the USB port corresponding to the port value specified in the `SatOS_HAL/bin/obc_soc_cfg.json` file under the tag `LNX_UART_COM_PORT`.

### 6.3.3 GS APP exe

1. Under the folder `SatOS_HAL/bin/`, there will be two GS APP executable:
    - **For GS side GS APP**: `uhf_tc_tm_sample_app_gs`
    - **For backdoor GS APP**: `uhf_tc_tm_sample_app_bk`

## 6.4 SatOS Execution Steps

### 6.4.1 Execution of SatOS SW in Linux Environment

1. Navigate to the `SatOS_HAL/bin/` directory:
2. Locate the `antaris_exo_core.exe` file.
3. Before running the EXE, ensure that your IP configuration is properly set in the JSON file. This file should be in the same directory.
4. Execute the EXE in the Linux terminal.

### 6.4.2 Execution of SatOS SW in Target Environment

1. Navigate to the `SatOS_HAL/bin/` directory:
2. Locate the `antaris_exo_core.elf` file.
3. Use an IDE or flash tool compatible with your Discovery/Stack board to load and flash the ELF file onto the board. Follow the instructions provided with the IDE or tool for this process.

### 6.4.3 Execution of GS APP

1. Navigate to the `SatOS_HAL/bin/` directory:
2. Locate the appropriate GS APP executable file:

    **For GS side GS APP**: `uhf_tc_tm_sample_app_gs`

    **For backdoor GS APP**: `uhf_tc_tm_sample_app_bk`

3. Ensure that your IP configuration is correctly set in the JSON file.
4. Place the JSON configuration file in the same directory where executable is running
5. Execute the GS APP in the Linux terminal

# 7 Document Revision History (Reverse Chronological Order)

| **Document Revision History** | | | | |
|---|---|---|---|---|
| Date | Revision Number | Author Name | Sections Updated | Change Description |
| 05-09-2014 | V1.0 | SatOS Team | Initial draft | Initial draft |
| 17-09-2014 | V1.1 | SatOS Team | Section 3.1,4.3,4.5, 5 and Appendix | Updated the Block Diagram, Updated the supported list of commands and APIs, Updated details for back door based access, Added OpenLST open source references for UHF parameter update and firmware update procedure |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |