Libre Space Foundation UOM 0003.43

SatNOGS COMMS User Manual November 2020

Table of Contents

Index of Tables

Table of Figures

1 Preface

SatNOGS COMMS is an open hardware and open source software project. This allows the end user to customize all aspects of the MCU and FPGA Software to meet their needs.

Further information regarding software can be found in the software repositories under https://gitlab.com/librespacefoundation/satnogs-comms/

2 Unpacking and handling

SatNOGS COMMS is and ESD sensitive device. Standard ESD handling methods should be followed when handling the PCB.

Always use an ESD mat and ESD wrist strap. It is advised to wear gloves when handling the PCB and shield to avoid contamination.

Should cleaning of any parts is required, use ESD safe cleaning methods and non-reactive solvents

3 Hardware setup

3.1 RF Connection

SatNOGS COMMS should always be powered on with an antenna or RF load connected to its RF ports. Failing to do so may result in catastrophic failure of the system.

A 50 Ohm antenna or 50 Ohm RF load capable of sinking 33dBm of power must be present to each RF connector at all times.

3.2 Power supply

SatNOGS COMMS can operate over a voltage range of 6 to 14V DC. This can be provided either through a power supply or directly from the satellite battery.

A power supply should be connected to VIN and GND pins as described in ICD.

Maximum power consumption is xxW

4 Communication interfaces

4.1 Programming interface

MCU can be programmed and debugged via the provided SWD/JTAG interface available in connector Jxxx

4.2 External connectivity

4.2.1 CAN Bus

There are two CAN Bus channels that can be configured either as standard CAN or CAN FD for speeds up to 5Mbps.

COMMS uses CAN Bus to communicate with other subsystems and relay data or receive commands.

4.2.2 I2C

I2C is provided for communication with I2C capable devices like antenna deployers, robes etc.

4.2.3 SPI

COMMS provides an SPI slave connection for up to 8 Mbit data transfer speeds. Slave communication is coordinated via CAN Bus instead of a dedicated slave select line.

There are 2 CAN Bus commands for SPI control:

1. SPI Select
2. SPI Release

Proper SPI communication sequence is:

1. Issue “SPI Select” command
2. Wait for acknowledge
3. Preform SPI communication
4. Issue “SPI Release” command

5. Wait for acknowledge

4.2.4 Antenna

There are two antenna control lines and two antenna feedback lines available for managing antenna deployment.

These are available on the dedicated antenna control connector as well as on the PC104 connector.

Available commands are:

- Antenna 1 control On/Off
- Antenna 2 control On/Off
- Antenna Feedback Request

Antenna Control Feedback is broadcast on CAN Bus when any of the two feedback lines changes state or an Antenna Feedback Request command is received

4.3 RF Interfaces

5 Data Link Layer (Layer 2)

Data link layer utilizes CCSDS Space Packet implemented by Open Space Data Link Protocol. See https://gitlab.com/librespacefoundation/osdlp for further information

6 Physical Layer (Layer 1)

BPSK, FSK, MSK IEE802.15.4, CCSDS

6.1 Basic RF Mode

6.1.1 Modulation

6.1.2 Frequency

TX and RX at 390 MHz to 500 MHz

TX at 2200 MHz to 2290 MHz

RX at 2025 MHz to 2110 MHz

6.1.3 Bandwidth

6.2 Extended RF Mode

6.2.1 Modulation

6.2.2 Frequency

6.2.3 Bandwidth

7 Command and status messaging

7.1 Commands

7.1.1 Antenna control

7.1.2 Antenna feedback request

7.1.3 SPI Select

7.1.4 SPI Release

7.2 Messages

7.2.1 Antenna Control Feedback

8 Self-protection functions

8.1 Temperature supervisor

There are several temperature sensor monitoring key components of COMMS. If an over-temperature event occurs, COMMS will disable the affected part of the circuit until temperature is within the operating envelope

8.2 Power supervisor

COMMS power lines are monitored for over-current conditions. If an over-current event occurs, the affected part of the circuit can be disabled allowing the rest of COMMS systems to operate.

COMMS power circuits incorporate continuous voltage monitoring of all onboard power supplies.

There is a mechanism for mitigating SEL of MOSFETS in the DC-DC converters. Should a SEL occur, COMMS will power down and will need to be power cycled by the EPS in order to recover from the SEL.

9 Telemetry and Telecommand

The telemetry and telecommand interface of the SatNOGS COMMS follows the CCSDS 232.1-B-2 utilizing the functionality and capabilities of the CCSDS Space Data Link.

The interface software between the ground station and the SatNOGS COMMS is the OSDLP-Operator[1]. This software, exploits the capabilities of the Space Link layer and implements all the necessary validity checks, re-transmissions, state transitions, priorities, virtual channel and session management, totally abstracting any implementation details from the ground station operator.

The CCSDS Communications Operation Procedure specifies the Frame Operation Procedure (FOP) that executes within the sending entity, and a Frame Acceptance and Reporting Mechanism (FARM) that executes within the receiving entity.

The OSDLP-Operator can support an arbitrary number of FOP and FARM configurations, using a descriptive text-based configuration language. Listings 1 and 2 provide example configurations for FARM and FOP respectively. The configurations can be altered without requiring any modification on the OSDLP-Operator software. This provides great flexibility on the ground station control software and allows for rapid development and prototyping, as well as fast reaction times during the in-orbit period.

Listing 1: FARM example configuration

```
// Declare which instance to run. For ground station use "fop".
// For receiving side (e.g. spacecraft) use "farm"
```

1 https://gitlab.com/librespacefoundation/osdlp-operator

```
instance = "farm"

// Mission Parameters
mission:{
  tc_sent_queue_max_cap = 200  // Maximum capacity of TC sent queue
  tc_tx_queue_max_cap = 200    // Maximum capacity of TC TX queue
  tc_rx_queue_max_cap = 200    // Maximum capacity of TX RX queue
  scid = 20                    // The spacecraft ID
  out_port = 6667
  in_port = 6666
}

// Define the TC parameters
tc:{
// Define one such struct per virtual channel
  vc1:{
  // The following parameters apply to the whole VC
     vcid = 1                     // The VC ID
     crc = 1                      // CRC flag. 1 for on, 0 for off
     segmentation = 0             // Defines whether to use segmentation

     // Defines the available MAPs. Each MAP is a channel contained into
     // the virtual channel. The following params apply to each MAP
     map:{
       map1:{
         mapid = 1               // The MAP ID
         bypass = 1              // Bypass flag. 0 for TYPE_A ->use the
                                 // sequence checking mechanism. 1 for
                                 // TYPE_B -> bypass sequence checking
                                 // mechanism
         ctrl_cmd = 1            // Control command flag. 0 for mission
                                 //specific commands/data, 1 for FARM
                                 // control information
         data = ""               // For future use
       }
     }
     // In case of farm instance, define this element
     farm:{
       win_width = 5             //Sliding window width
     }
  }

  vc2:{
     vcid = 2
     crc = 1
     segmentation = 0
     map:{
       map1:{
         mapid = 1
         bypass = 1
         ctrl_cmd = 1
         data = ""
       }
     }
     farm:{
       win_width = 5
     }
  }
```

```
}

// Define the TM parameters
tm:{
// One such element per virtual channel
  vc1:{
     vcid = 1              // The virtual channel ID
     crc = 1               // CRC flag. 1 for on, 0 for off
     ocf_flag = 1          // OCF flag. 1 for on, 0 for off
     sec_hdr_on = 0        // Secondary header flag. 1 for on, 0 for off
     sync_flag = 0         // Sync flag
     stuff_state = 1       // Stuff packets state. 0 for ON 1 for OFF
  }
}
```

Listing 2: FOP example configuration

```
instance = "fop"

mission:{
  tc_sent_queue_max_cap = 200
  tc_tx_queue_max_cap = 200
  tc_rx_queue_max_cap = 200
  scid = 1
  out_port = 16881
  in_port = 16880
  stdout_port = 16882        // UDP port for printing log messages
}

tc:{
  vc0:{
     vcid = 0
     crc = 1
     segmentation = 1
     vc_name = "Management"
     map:{
       map1:{
         mapid = 1
         bypass = 0
         ctrl_cmd = 0
         data =
"xA0x0Cx20x0CxA0x0Cx20x0CxA0x0Cx20x0CxA0x0Cx20x0CxA0x0Cx20x0CxA0x0Cx20x0C"
         name = "Change periodic telemetry attributes"
       }
       map2:{
         mapid = 2
         bypass = 0
         ctrl_cmd = 0
         data = "xff"
         name = "Request periodic telemetry attributes"
       }
       map3:{
         mapid = 3
         bypass = 0
         ctrl_cmd = 0
         data = ""
```

```
        name = "Kill switch"
      }
      map4:{
        mapid = 4
        bypass = 0
        ctrl_cmd = 0
        data = "xff"
        name = "Deploy antenna"
      }
    }
    //In case of fop instance, define this element
    fop:{
      t1_init = 5                 // The initial timer value in sec
      transmission_limit = 20     // Transmission limit per packet
      timeout_type = 0            // Timeout type
      win_width = 5               // Sliding window width
    }
 }
   vc2:{
    vcid = 2
    crc = 1
    segmentation = 1
    vc_name = "Request TM"
    map:{
      map1:{
        mapid = 1
        bypass = 0
        ctrl_cmd = 0
        data = "xa4"
        name = "Request TM"
      }
    }
    fop:{
      t1_init = 5
      transmission_limit = 30
      timeout_type = 0
      win_width = 20
    }
 }

  vc3:{
    vcid = 3
    crc = 1
    segmentation = 1
    vc_name = "Experiment"
    map:{
      map1:{
        mapid = 1
        bypass = 0
        ctrl_cmd = 0
        data = "x01xFEx02x01x04x00x0A"
        name = "Execute experiment"
      }
    }
    fop:{
      t1_init = 5
      transmission_limit = 30
      timeout_type = 0
```

```
        win_width = 5
      }
   }
}

tm:{
   vc0:{
      vcid = 0
      crc = 1
      ocf_flag = 1
      sec_hdr_on = 0
      sync_flag = 0
      stuff_state = 1
      vc_name = "Management"
   }
   vc1:{
      vcid = 1
      crc = 1
      ocf_flag = 0
      sec_hdr_on = 0
      sync_flag = 0
      stuff_state = 1
      vc_name = "Regular TM"
   }
     vc2:{
      vcid = 2
      crc = 1
      ocf_flag = 1
      sec_hdr_on = 0
      sync_flag = 0
      stuff_state = 1
      vc_name = "Request TM"
   }
     vc3:{
      vcid = 3
      crc = 1
      ocf_flag = 1
      sec_hdr_on = 0
      sync_flag = 0
      stuff_state = 1
      vc_name = "Experiment"
   }
}
```

9.1 Ground control setup

The provided ground control software allows telemetry reception and telecommand of SatNOGS COMMS using an SDR transceiver.

9.2 Telemetry format

9.3 Telecommand format

9.4 SatNOGS integration

10 Diagnostics