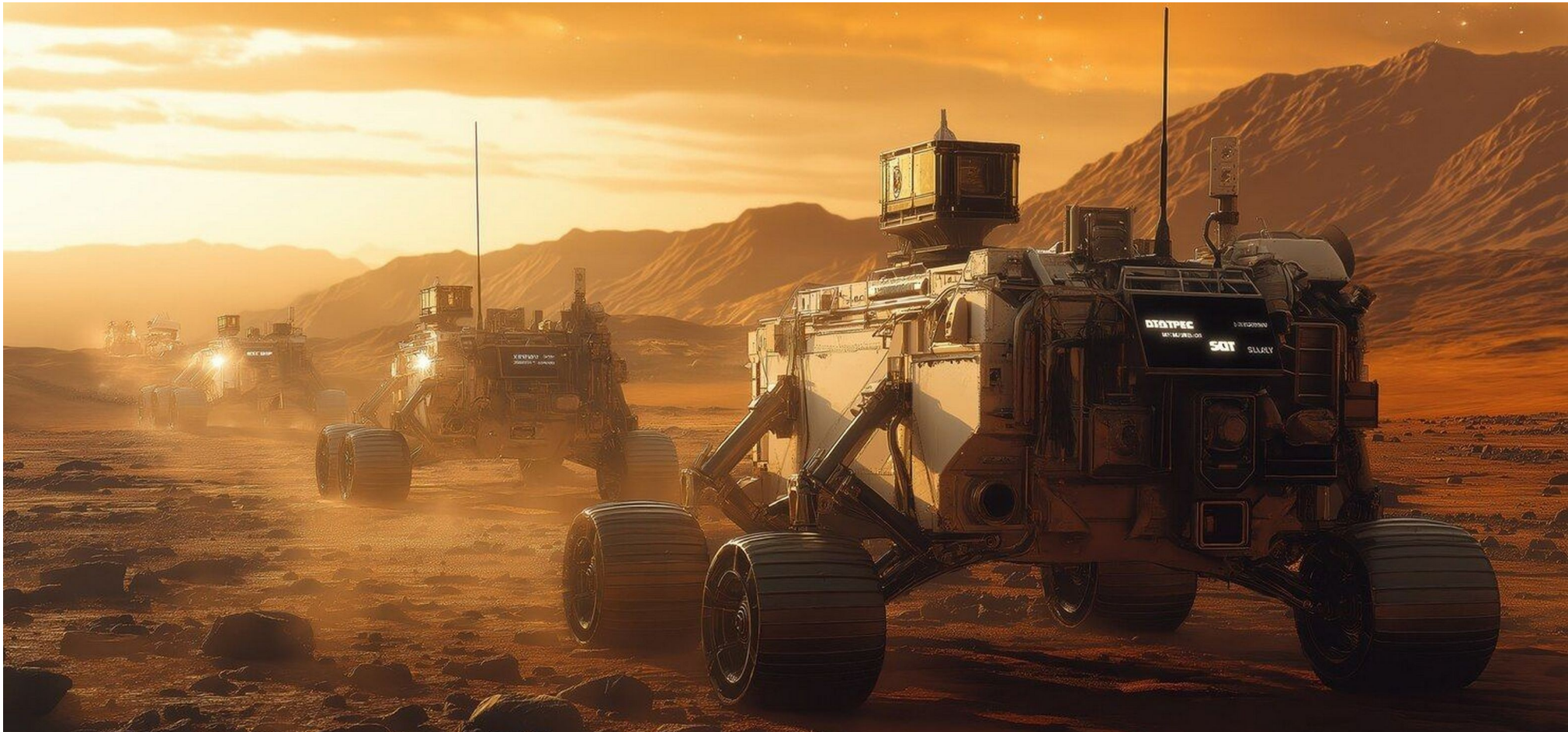
SpaceCAN Overview

LIBRE CUBE
open source space exploration

Release date: 2025-05-16

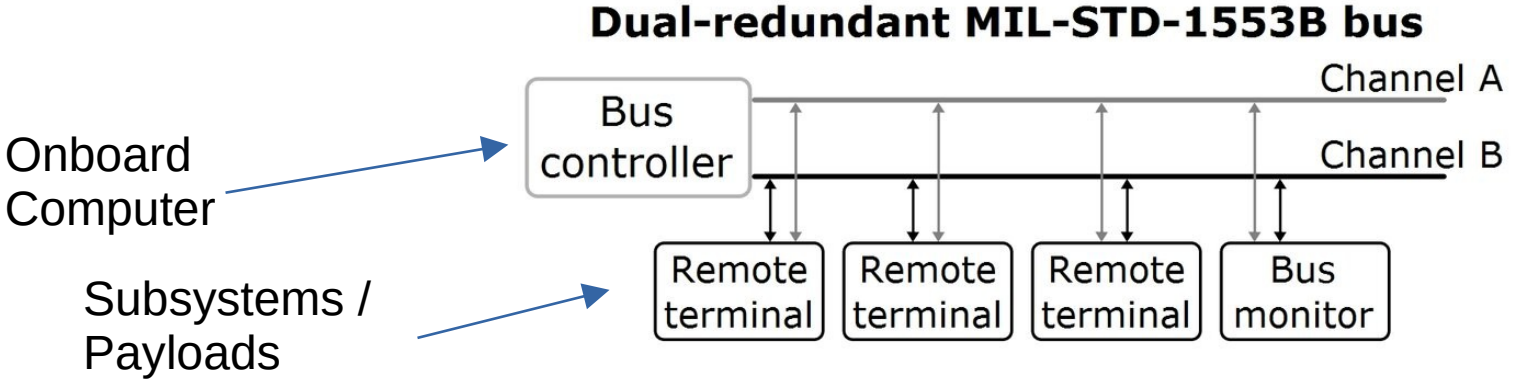


Introduction

Onboard Communication Bus

LIBRE CUBE
open source space exploration

- For most satellites, a central processing unit is commanding other intelligent nodes and collects status information from them.
- The data to be exchanged on this bus is typically of low volume but must be transmitted in a reliable way.
- For this, small messages are sent between nodes that must arrive without error and with very little delay.



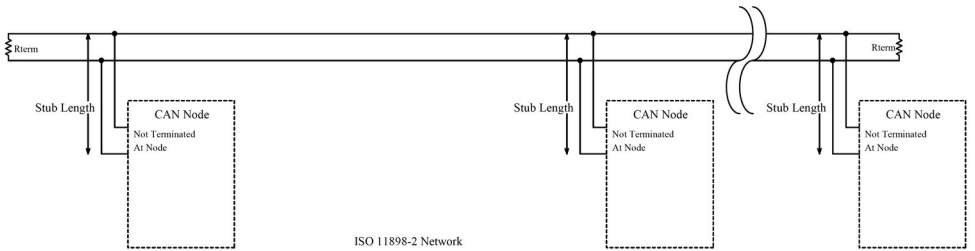
Examples of Onboard Busses

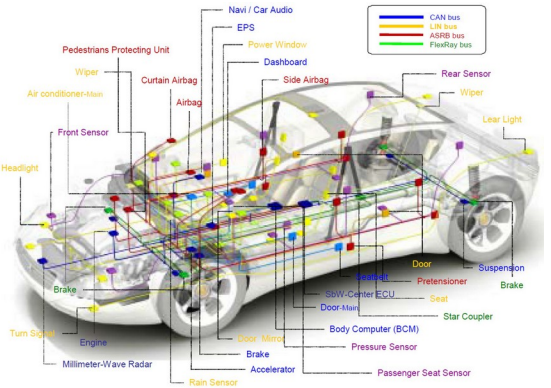
LIBRE CUBE
open source space exploration

- **MIL-STD-1553**: Used on almost all agency and industry satellites. Introduced by US Airforce in 1973.
- **SpaceWire**: A relatively new space bus protocol that supports routing with high data rates.
- **CAN Bus**: A low-cost, reliable communication bus, coming from automotive.
- **I2C**: A simple bus invented for microcontroller peripherals. Used on almost every CubeSat – a common source of mission failure!

CAN Bus

- The controller area network bus (CAN bus) developed by Bosch GmbH in 1983 is a vehicle bus standard designed to enable efficient communication primarily between electronic control units.
- Originally developed to reduce the complexity and cost of electrical wiring in automobiles through multiplexing, the CAN bus protocol has since been adopted in various other contexts, such as robotics and industrial automation.





CAN Bus for Space

LIBRE CUBE
open source space exploration

CAN Bus is used in space since many years:

- Surrey Satellite Technology Ltd (SSTL) introduced the use of CAN on a spacecraft in 1998 with FASat-Bravo, followed by UoSat-12 in 1999.
- SMART-1 was the first ESA satellite to integrate CAN bus
- Eurostar 3000 platform and many more
- Also on some CubeSats (via CSP protocol)

ECSS CANBus Extension Protocol

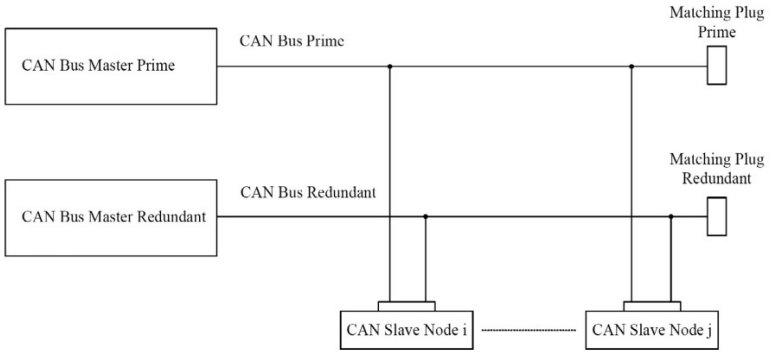
LIBRE CUBE
open source space exploration

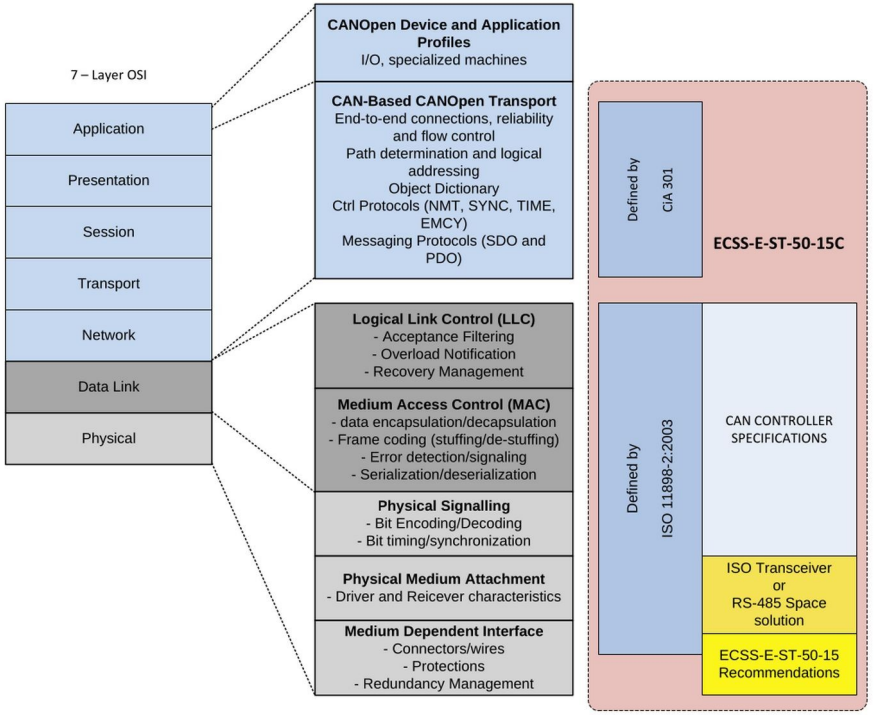
- ECSS (European Cooperation for Space Standardization) developed the CANbus Extension Protocol Standard for on-board communications and control: **ECSS-E-ST-50-15C**
- All ECSS standards freely available, (but restricted to ESA member states) at https://ecss.nl/

ECSS CANBus Extension Protocol

LIBRECUBE
open source space exploration

- The standard defines the physical layer (via standard CAN protocol) as well as the layers above (via industrial CANopen protocol).

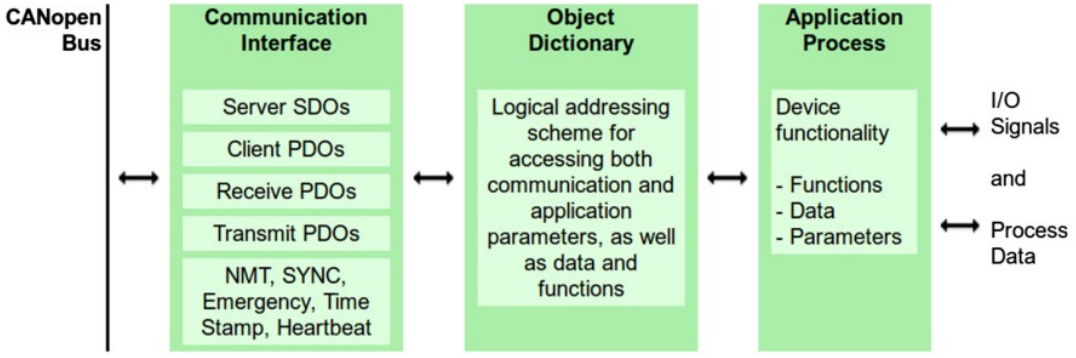


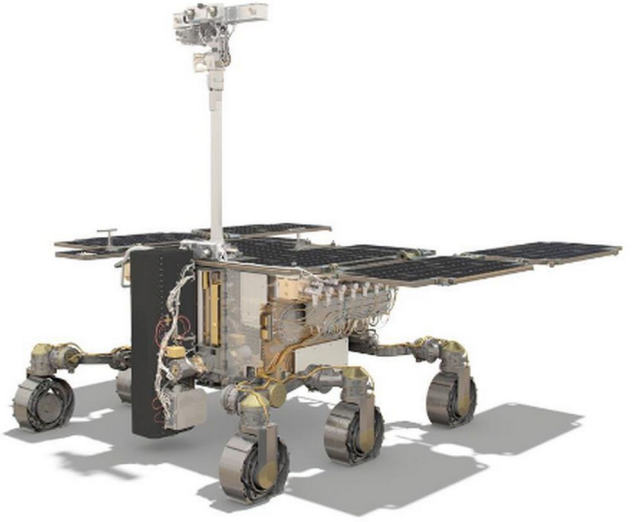


ECSS CANBus in Space

LIBRECUBE
open source space exploration

- To be used on ESA's Exomars Rover (named Rosalind Franklin).
- The CANopen protocol uses a number of additional concepts for communication, including time distribution and redundancy scheme.



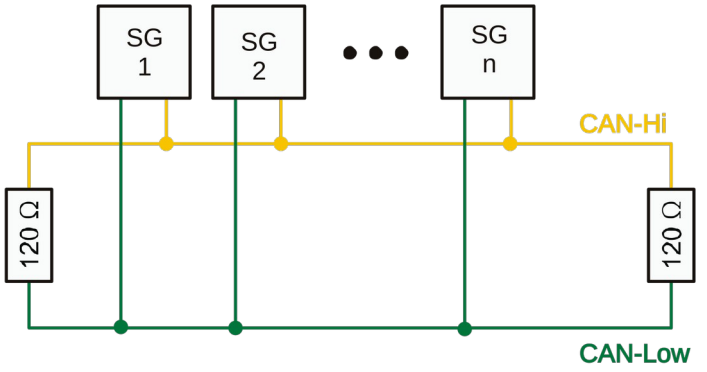


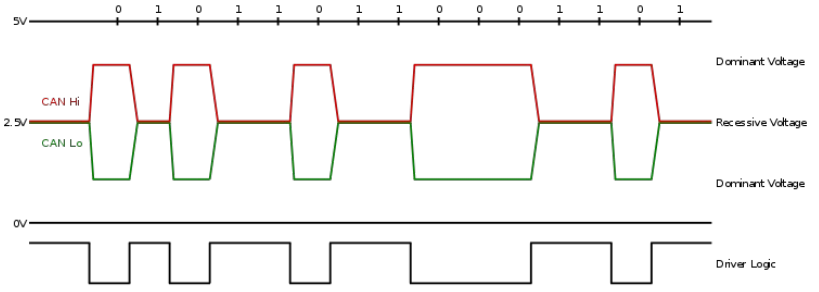
CAN Bus Details

CAN Physical Layer

LIBRECUBE
open source space exploration

- Nodes are connected to each other through a two-wire bus. The wires are a twisted pair with a 120 Ω characteristic impedance.
- This bus uses differential wired-AND signals. Two signals, CAN high (CANH) and CAN low (CANL) are either driven to a "dominant" state with CANH > CANL, or not driven and pulled by passive resistors to a "recessive" state with CANH ≤ CANL.
- A “0” data bit encodes a dominant state, while a “1” data bit encodes a recessive state, supporting a wired-AND convention, which gives nodes with lower ID numbers priority on the bus.

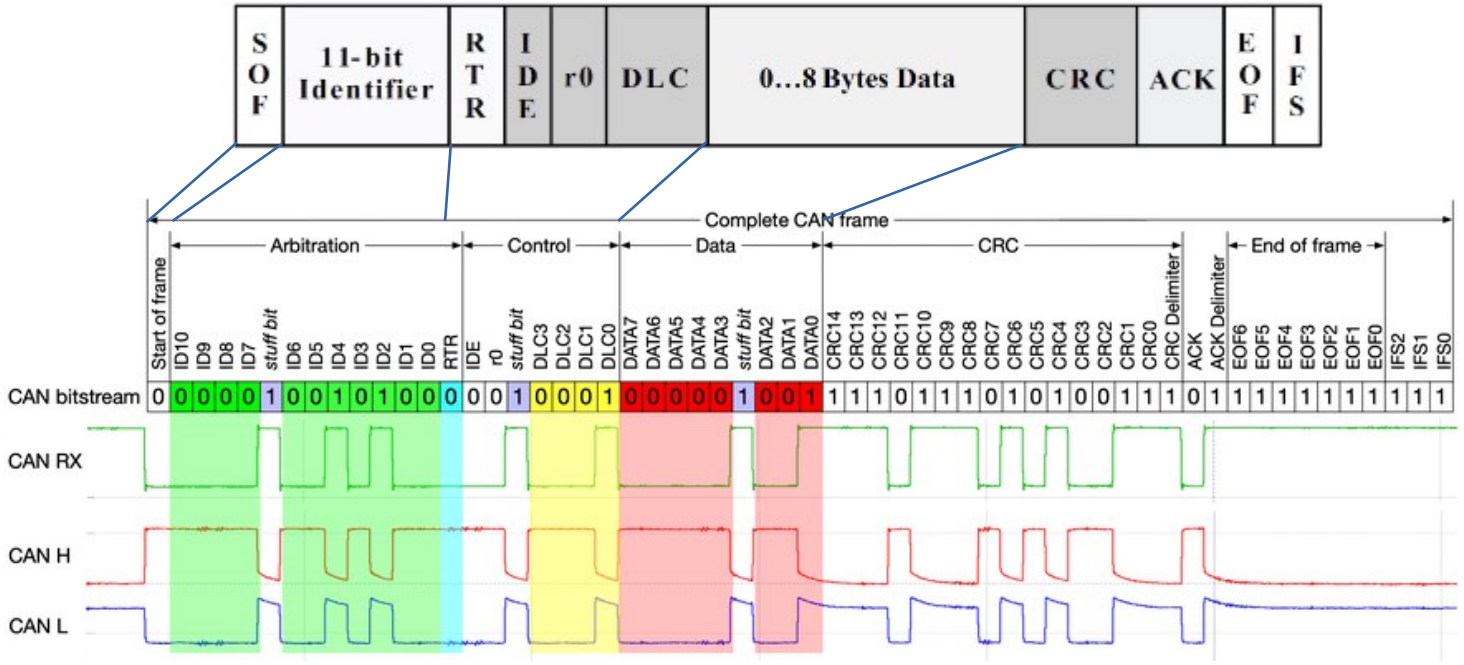




Source: Wikipedia

CAN Transfer Layer

- The CAN frame carries up to 8 bytes. It is composed of a header, a data field, and a tail.

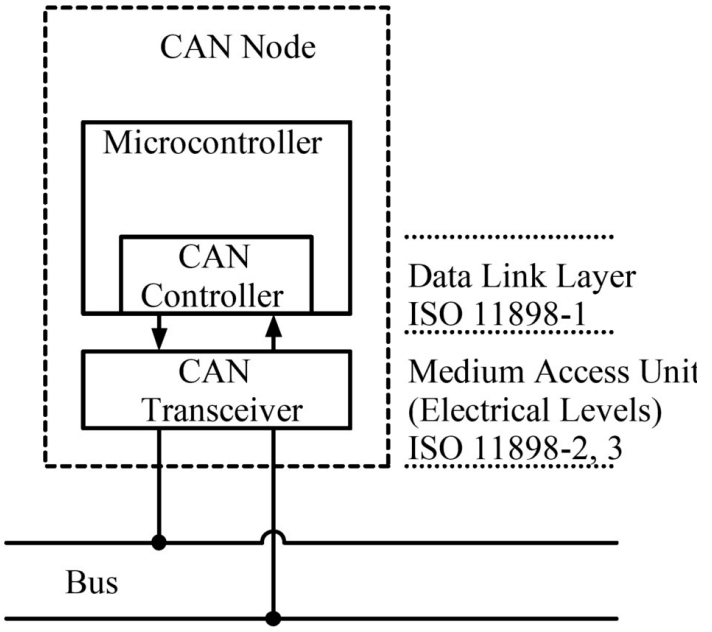


Source: Wikipedia

CAN Nodes

A CAN network is made up my interconnected nodes. Each node requires:

- A processing unit, microprocessor, or host processor, which consumes and generates CAN messages.
- A CAN controller (often part of the microcontroller), which takes care of the datalink layer protocol:
  - Receiving: the CAN controller stores the received serial bits from the bus until an entire message is available, which can then be fetched by the host processor (usually by the CAN controller triggering an interrupt).
  - Sending: the host processor sends the transmit message(s) to a CAN controller, which transmits the bits serially onto the bus when the bus is free.
- A transceiver
  - Receiving: it converts the data stream from CAN bus levels to levels that the CAN controller uses. It usually has protective circuitry to protect the CAN controller.
  - Transmitting: it converts the data stream from the CAN controller to CAN bus levels.



Source: Wikipedia

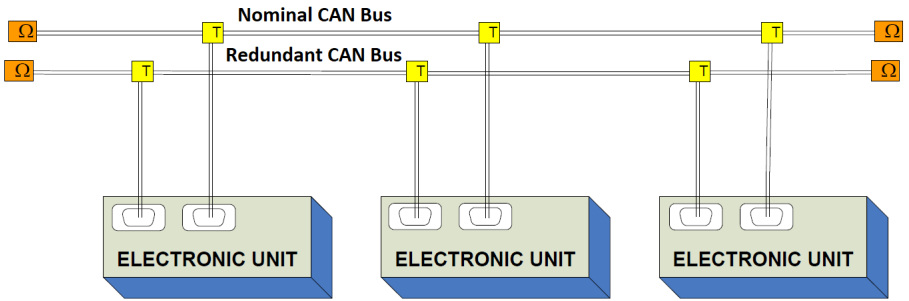
LibreCube SpaceCAN Bus

Background

- For typical CubeSat missions the CANopen part of ECSS CANbus is deemed too complex.
- LibreCube has modified the ECSS CANbus standard, to be easy to use and implement even in constrained environments (eg. microcontrollers).
- SpaceCAN bus is used for control and monitoring, allowing the exchange of commands and telemetry among spacecraft subsystems.
- Documentation: https://librecube.gitlab.io/standards/spacecan/

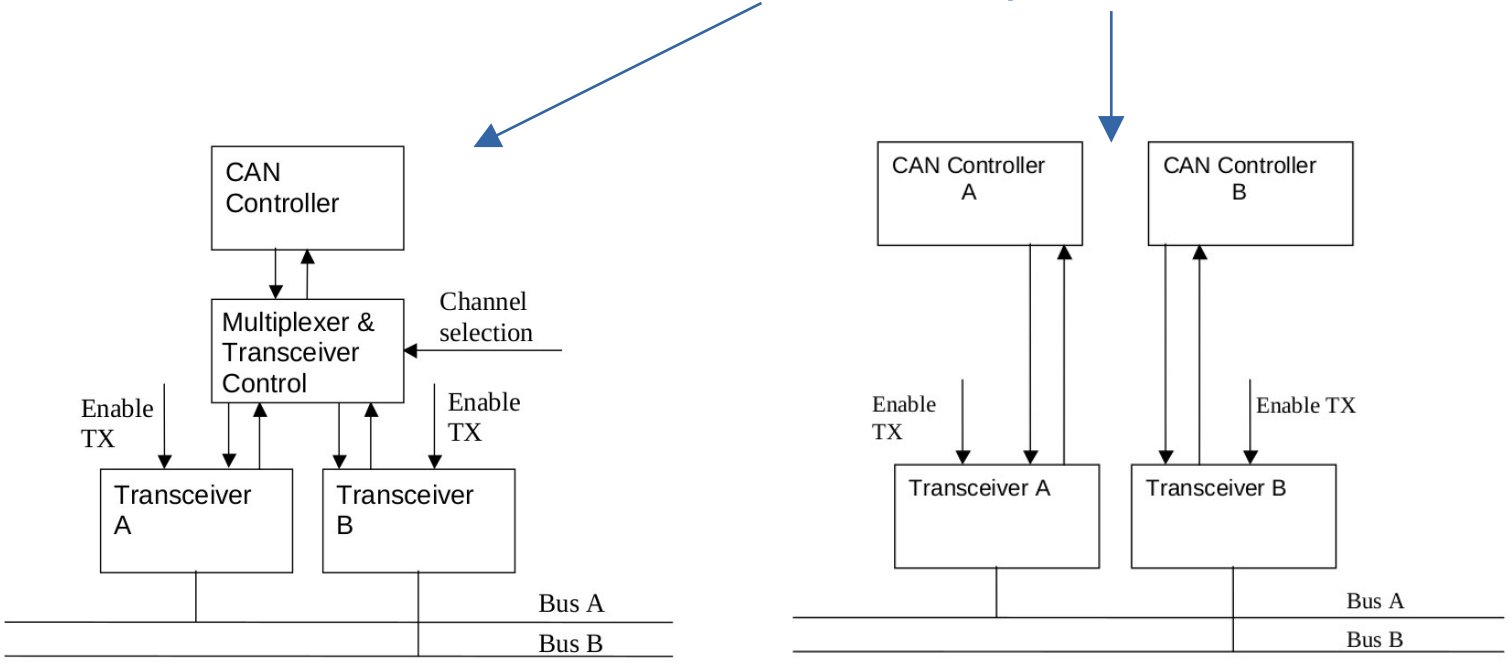
Topology

- Linear bus, composed of a nominal and redundant chain (bus A and bus B).
- The controller node can talk to responders and the responders can talk to the controller. The responders do not talk with each other.
- The network is thus composed of a single controller (with node ID 0) and up to 127 responders (with node ID 1 to 127)



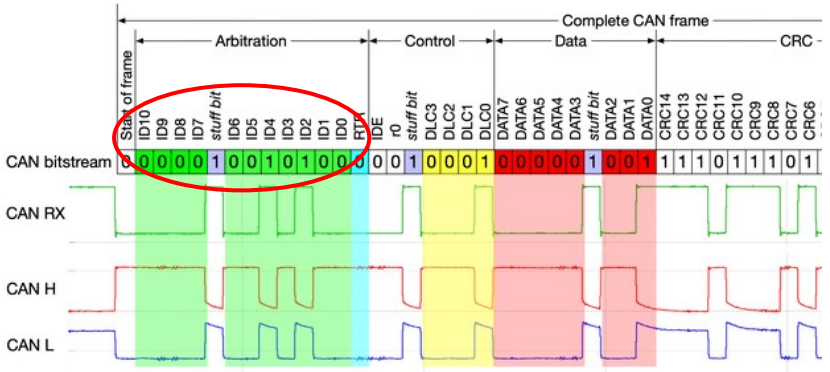
Bus Architecture

- Nodes can connect to the bus via selective or parallel access.



CAN ID Format

- Every CAN frame has an 11-bit field for the CAN ID.
- This is split in a 4-bit function code and a 7-bit node ID.
- The function code identifies the type of message that is sent, the node ID defines the sender (for TM) or recipient (for TC), respectively.



Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

Features

LIBRE CUBE
open source space exploration

- Redundancy Management
- Synchronization
- Time Distribution (SCET and UTC)
- Telecommand and Telemetry
- Packet Protocol
- SpaceCAN Service Protocol (SSP)

Redundancy Management

LIBRE CUBE
open source space exploration

- Bus is resilient to single point failure (such as problem in cabling or connector fault) through redundant physical media topology.
- The selected scheme for redundancy applies the concept of node monitoring via heartbeat frames.
- The controller node defines the active bus by periodic transmission of heartbeat frames (CAN ID = 0x700, no data) on it.

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

Responder Heartbeat Monitoring

LIBRE CUBE
open source space exploration

The responder nodes monitor the presence of the heartbeat from the controller to determine the active bus. It follows this logic:

- When a responder node misses the controller heartbeat for a defined number of times it shall switch to the alternate bus and listen there for heartbeats.
- If it detects a heartbeat on that bus, it shall consider this one as the active bus.
- If no heartbeat is received after times, it shall switch again to the other bus.
- This bus toggling shall be continued for a predefined number of times, or for infinite time (configurable).

Synchronization

LIBRECUBE
open source space exploration

- Synchronous network behavior is achieved with the SYNC protocol.
- The controller node periodically transmits SYNC frames (CAN ID = 0x080, no data) to indicate to the responders to start their application-specific behavior.
- This could trigger for example the initiation of measurements or the sending of telemetry to the controller node.
- The SYNC period is usually in the range of a few seconds.

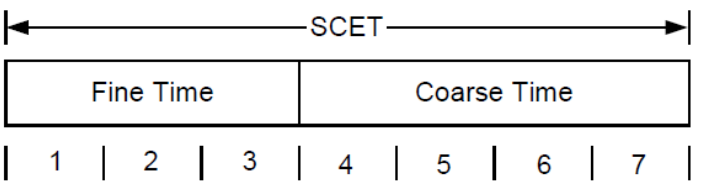
Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

Time Distribution – SCET

LIBRE CUBE
open source space exploration

- The central computer manages the spacecraft time. In addition, other systems on the spacecraft may also maintain a local time (for example an attitude control system with its own processing unit).
- SCET stands for “Spacecraft Elapsed Time”.
- Basically it is a counter of seconds (coarse) and subseconds (fine) from a defined time (eg. system boot up).

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

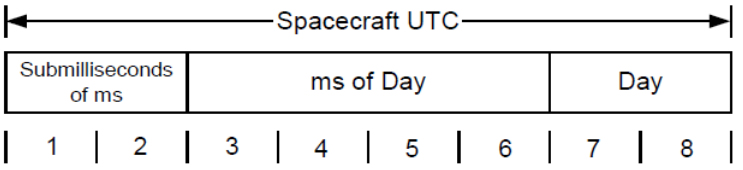


Time Distribution – UTC

LIBRE CUBE
open source space exploration

- UTC stands for “Universal Time Coordinated”.
- It counts days and milliseconds (and submilliseconds), with references to a predefined date (eg. 1 January 2000).

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder



Telecommands

LIBRECUBE
open source space exploration

- Telecommands are sent from controller to responder, whereas telemetry are frames sent from responders to the controller. Each frame contains data of up to 8 bytes. How the data is interpreted is application specific.
- The controller node can send telecommands to responder nodes. Telecommands usually trigger some kind of action, like switching a unit on or off, etc. A telecommand frame contains the node ID of the node to which the telecommand is being sent to (in the CAN header) and the data field with up to 8 bytes.

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

Telemetry

LIBRECUBE
open source space exploration

- The responder nodes send telemetry to the controller node. Telemetry comprises of essential information about the nodes (also called housekeeping data), such as operational status and sensor readings (temperatures, currents, voltages, etc.).
- A telemetry frame contains the node ID of the responder node (as indication to the controller from which node the data originates) and the data field with up to 8 bytes.

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

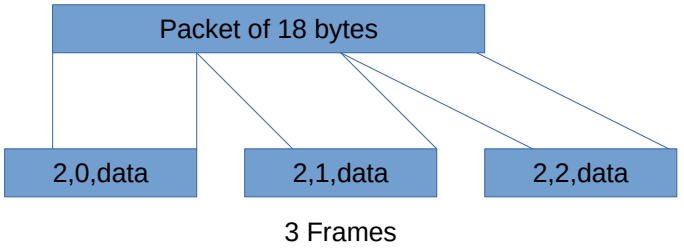
Packets (1/2)

LIBRE CUBE
open source space exploration

- To circumvent the limitation of 8 bytes of data to be exchanged, the packet protocol was added. The concept is simple: a packet (that is, a large piece of data) is fragmented into smaller parts of up to 6 bytes that fit into a frame.
- The first two bytes of a frame data field are used for reassembling them back into the packet. The first byte tells the total number of frames that make up the packet and the second byte is the counter of each frame:
  - Byte 1: total number of frames for this packet minus 1
  - Byte 2: current number of this frame, starting from 0
  - Byte 3-8: data chunk

Packets (2/2)

- Shown below is an example of a packet with 16 bytes.
- It is split into 3 CAN frames where the first two frames carry 6 bytes of data and the last frame the remaining 4 bytes.

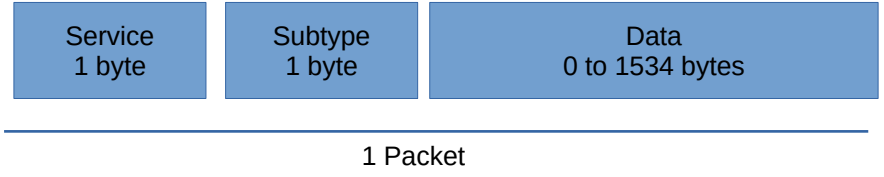


- Important: The packet protocol utilizes the telecommand and telemetry frames and thus they cannot be used at the same time. So either use the packet protocol, or the telecommand/telemetry protocol.

SpaceCAN Service Protocol (SSP)

LIBRECUBE
open source space exploration

- The packet protocol allows for sending larger amount of data, but it still leaves the interpretation of this data completely to the user.
- To care for this, SpaceCAN supports an (optional) packet service protocol. This protocol is very much inspired from the ECSS Packet Utilization Protocol ECSS-E-ST-70-41C and tries to be as close as possible to its syntax.
- The implementation is simple: each packet (which can have a maximum length of 256 * 6 bytes = 1536 bytes) carries as first byte a service and as second byte a subtype identifier.



Service Types

LIBRE CUBE
open source space exploration

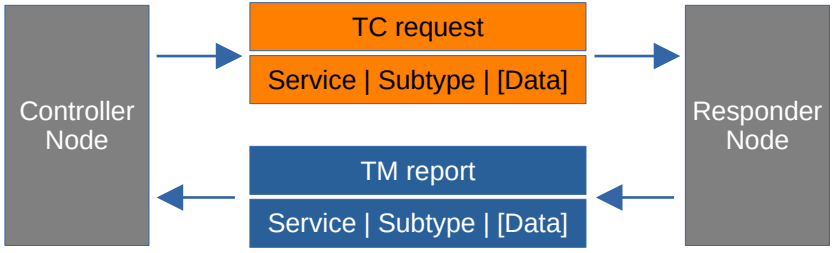
- Service 01: Request Verification
- Service 03: Housekeeping Management
- Service 08: Function Management
- Service 17: Test
- Service 20: Parameter Management

Note that the service numbering follows the convention of “ECSS-E-ST-70-41C – Telemetry and telecommand packet utilization”.

Service Types and Subtypes

LIBRE CUBE
open source space exploration

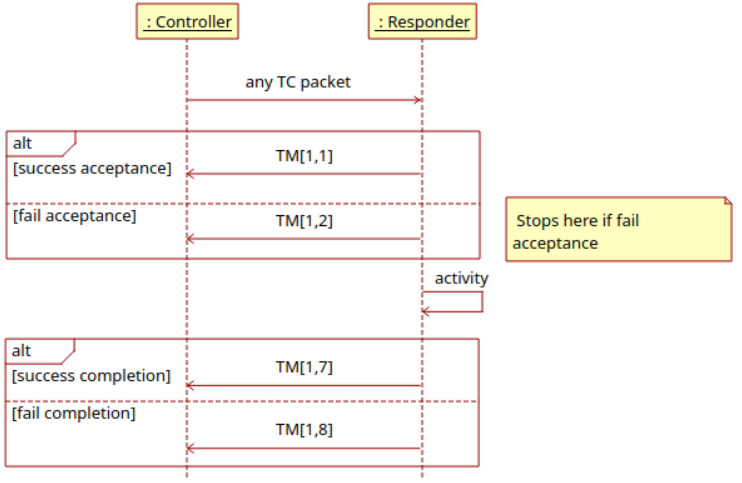
- Service 01: Request Verification
  - TM 1,1 successful acceptance
  - TM 1,2 failed acceptance
  - TM 1,7 successful completion
  - TM 1,8 failed completion
- Service 03: Housekeeping Management
  - TC 3,5 enable periodic housekeeping reports
  - TC 3,6 disable periodic housekeeping reports
  - TC 3,27 request single housekeeping report
  - TM 3,25 housekeeping parameter report
- Service 08: Function Management
  - TC 8,1 perform function
- Service 17: Test
  - TC 17,1 perform connection test
  - TM 17, 2 connection test report
- Service 20: Parameter Management
  - TC 20,1 report parameter values
  - TM 20,2 parameter value report



Service 01: Request Verification

LIBRE CUBE
open source space exploration

- This service reports to the controller the status of the acceptance and execution of a request sent to a responder.
- An acceptance report is generated immediately after completion of validity checks of the received request; an execution report is generated after the execution of the request.
- This service is active for every TC that the controller sends to a responder. It is therefore omitted from the diagrams of the other services for simplicity.
  - TM 1,1 successful acceptance — 1 | 1 | request service | request subtype
  - TM 1,2 failed acceptance — 1 | 2 | request service | request subtype
  - TM 1,7 successful completion — 1 | 7 | request service | request subtype
  - TM 1,8 failed completion — 1 | 8 | request service | request subtype



Service 03: Housekeeping Management

LIBRECUBE
open source space exploration

- The housekeeping service type provides the visibility of any on-board parameters assembled in housekeeping reports. The structure of the reports used by the housekeeping service are predefined.
  - TC 3,5 enable periodic housekeeping report

3 | 5 | N |	housekeeping id

repeated N times

  - TC 3,6 disable periodic housekeeping report

3 | 6 | N |	housekeeping id

repeated N times

  - TC 3,27 request single housekeeping report

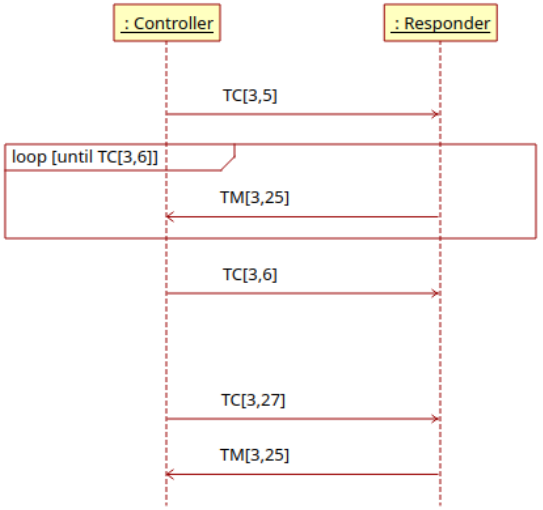
3 | 27 | N |	housekeeping id

repeated N times

  - TM 3,25 housekeeping parameter report

3 | 25 | housekeeping id | parameter value (encoded)

deduced repeated number of times

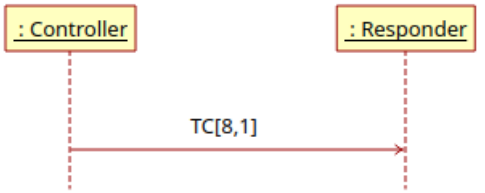


- Note: The size of each of the encoded parameter values is deduced from the housekeeping id.

Service 08: Function Management

LIBRECUBE
open source space exploration

- The function management service type provides the capability for performing specific functions of an application process.
  - TC 8,1 perform function

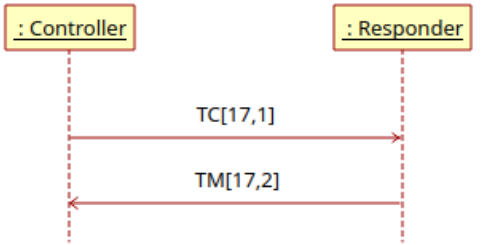


8	1	function id	N	argument id	argument value (encoded)
					repeated N times

- Note: The size of the encoded argument value is deduced from the argument id.

Service 17: Test

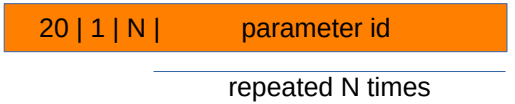
- The test service type provides the capability to activate test functions implemented on-board and to report the results of such tests.
- It can be used for simple ping to see if responders are available and if specific application processes are available.
  - TC 17,1 perform connection test 17 | 1
  - TM 17,2 connection test report 17 | 2



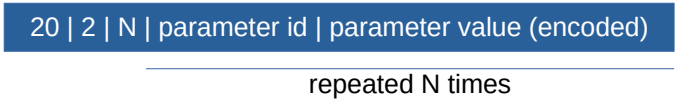
Service 20: Parameter Management

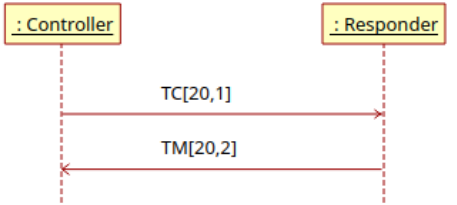
LIBRE CUBE
open source space exploration

- The parameter management service type provides capabilities to the controller for reading current values of on-board parameters from responders.
  - TC 20,1 request report of parameter values



  - TM 20,2 parameter value report





- Note: The size of each of the encoded parameter values is deduced from its parameter id.