CAN BUS in space systems

7th ESA Workshop on Avionics, Data, Control and software systems

AOE74

Boleat Christian // 22/10/2013

All the space you need

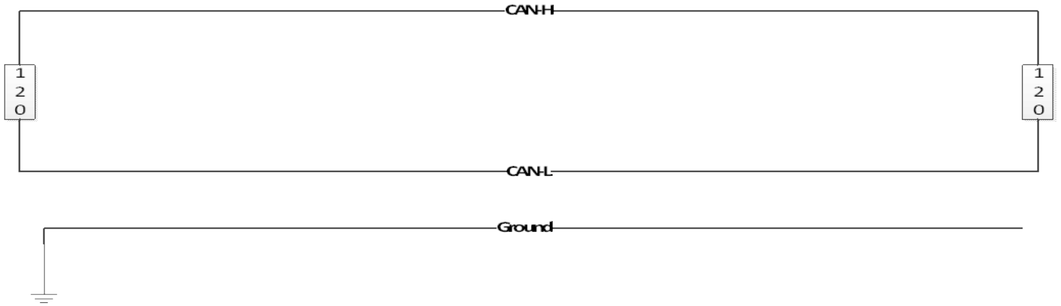
ASTRIUM
AN EADS COMPANY

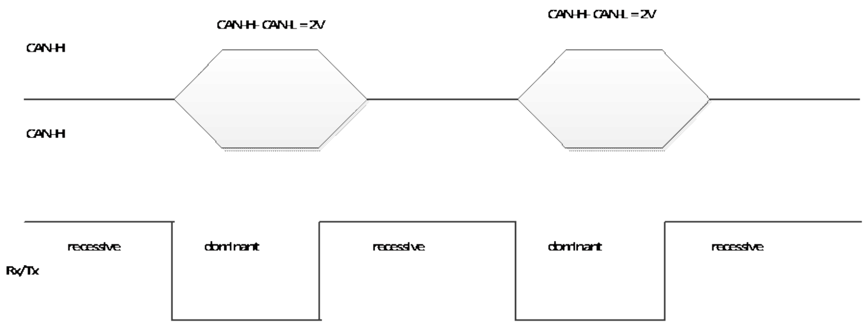
CAN BUS in space systems

- Table of contents

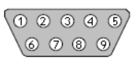
CAN BUS HARDWARE

CAN bus is a twisted pair of cables and a ground cable





Pin #	Signal Names	Signal Description
1	Reserved	Upgrade Path
2	CAN_L	Dominant Low
3	CAN_GND	Ground
4	Reserved	Upgrade Path
5	CAN_SHLD	Shield, Optional
6	GND	Ground, Optional
7	CAN_H	Dominant High
8	Reserved	Upgrade Path
9	CAN_V+	Power, Optional



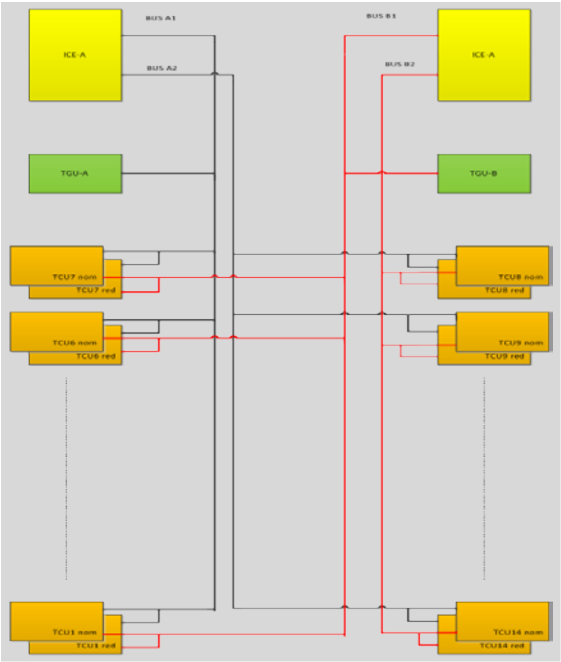
The standard connection is a 9 points cannon connector

A dominant state is potentiel between CAN-H and Can-L

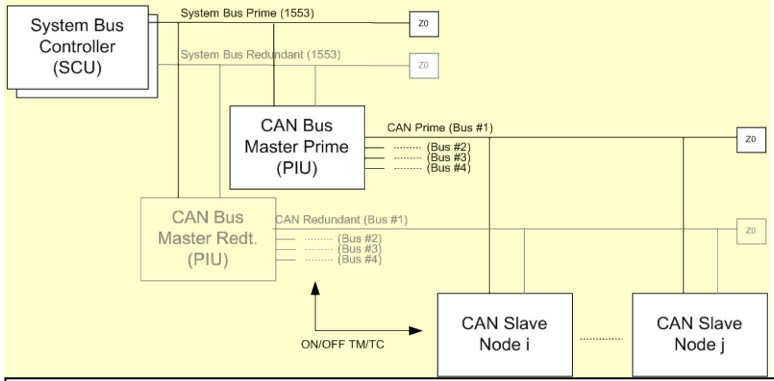
A recessive state is no potentiel between CAN-H and CAN-L

Redundancy in space systems

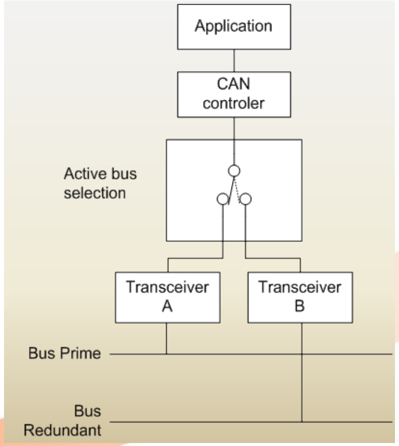
This document and its content is the property of Astrium [Ltd/SAS/GmbH] and is strictly confidential.
It shall not be communicated to any third party without the written consent of Astrium [Ltd/SAS/GmbH].



Exemple of redundancy in Sentinel 1



redundancy on telecom architecture exemple

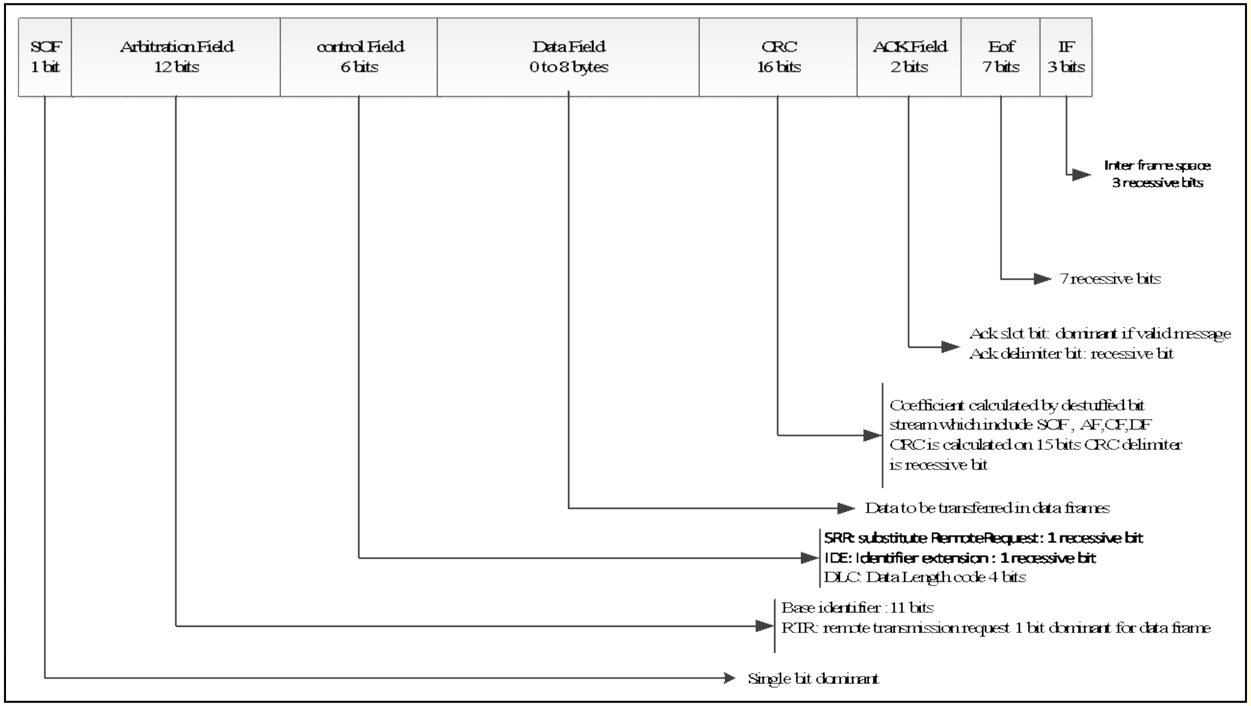


redundancy in can slave node exemple

All the space you need

ASTRIUM
AN EADS COMPANY

CAN 2.0A frame



SOF: defines the exchange start, it is a dominant bit.

The bus must be idle before all nodes must synchronize on the edge before the start bit transition

Arbitration field: The arbitration is made by this set of 11 bits , ID10 to ID4 must not be all recessive.

RTR: remote transmission request is recessive if a request frame, is dominant during data frame

Reserve bits: are used to guarantee future ascending compatibilities.

DLC: indicate data number in data field , min is 0 max = 8

Data field: useful data transmitted with Most Significant Bit firstly.

CRC: 15 bits of CRC and 1 bit CRC delimiter

ACK: each time a receiver node has correctly received a valid message , it superimposes during the time slot of ACK slot a dominant bit . the ACK delimiter must always be recessive, when a message has been correctly received the bit ACK slot (dominant) is surrounded by two recessive bits.

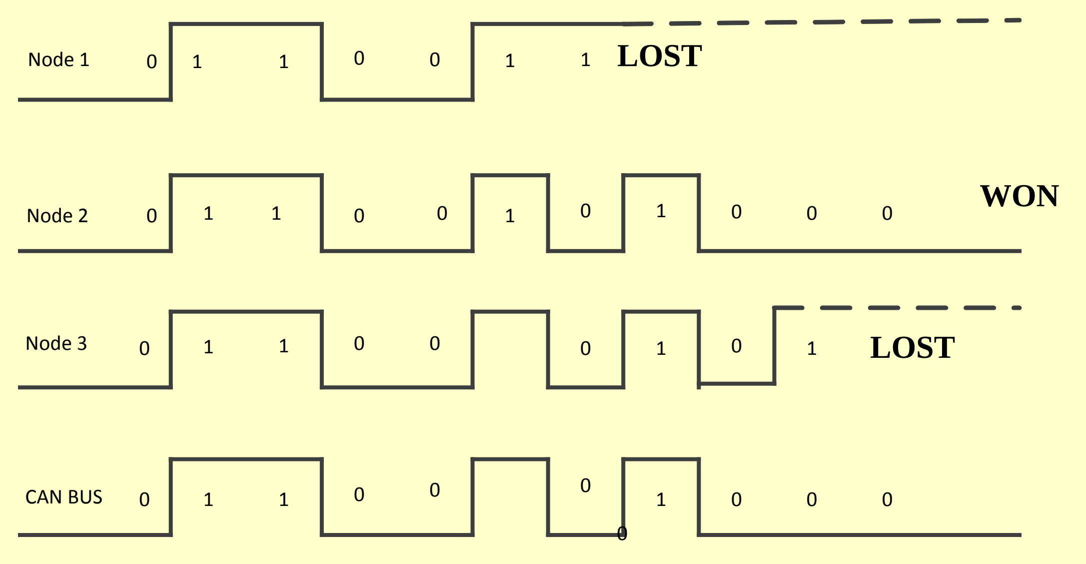
End Of Frame: sequence of 7 successive recessive bits , the stuffing and destuffing are deactivated during this sequence of end of frame.

Interframe: no one node is authorized to transmit a frame, only an overload action is enabled

For a request frame the RTR bit is recessive, this bit allows to differentiate both frame types. Data field is always empty.

CAN BUS ARBITRATION

- The arbitration is a solution which is provided to give the communication support to one node by trying to get its control.
- When the bus is free, two or more nodes can start simultaneously, so there is a conflict on the bus which is resolved by a not destructive bit to bit arbitration all along the identifier content. This solution makes that there is no loss of time or information.

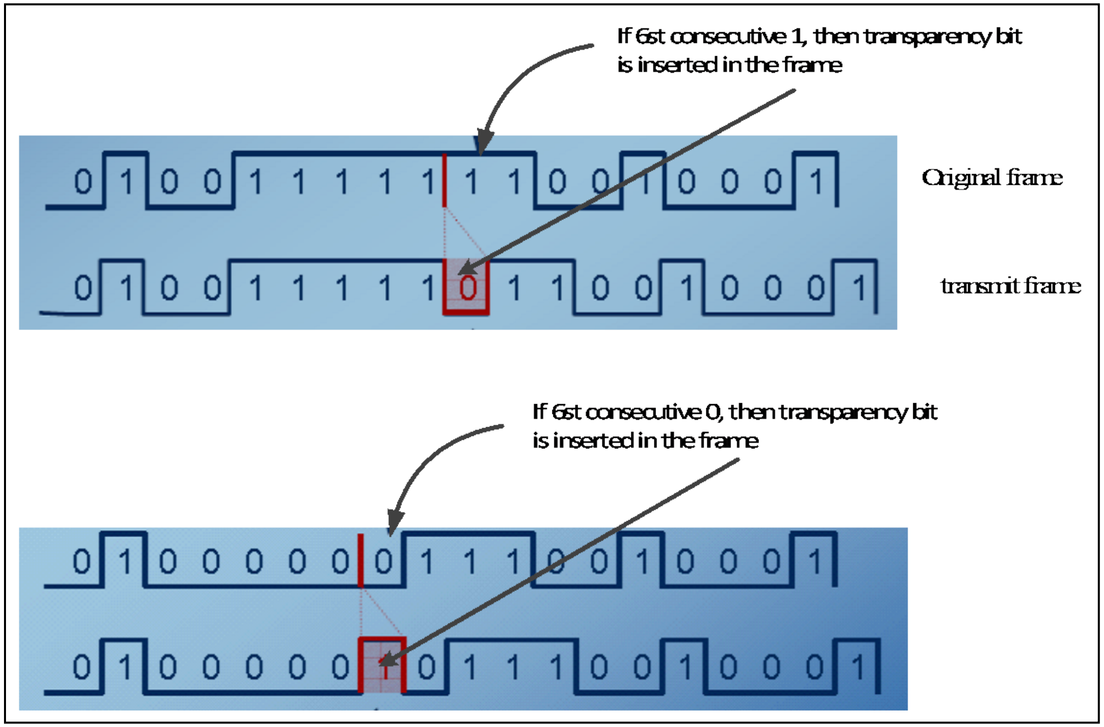


When 3 nodes try to transmit, they are emitting and listening at same time, the rule is that if the node does not emit (recessive bit) and it measures a potential difference (dominant bit) then it stops its transmission.

This arbitration is only done on the field identifier

There are two types of frames, data frame which drive data and remote frame which inform others nodes of the network that it wants to receive data from them.

CAN BUS BIT STUFFING



The worst case happens when a set of 5 consecutive identical bits is followed by 4 opposite identical bits, this situation generates 25% bits more

- NRZ coding type is used in order to limit the number of transitions, but lay down to a very stable clock to avoid jitter problems, the bit stuffing method resolves this problem
- The worst case happens when a set of 5 consecutive identical bits is followed by 4 opposite identical bits, this situation generates 25% bits more
- The bit stuffing consists in:

Insert a ‘0’ after five consecutives ‘1’

Insert a ‘1’ after five consecutives ‘0’

CAN BUS error management

- 5 different types of error:

Bit Error: a bit error occurs when the bit monitored is different from the bit sent , except if sending a recessive bit during the stuffed bit stream of the arbitration field or during the ACK slot

Stuff error: a stuff error must be detected at the bit if six consecutive equal bit level in a message field which should be coded by the method of bit stuffing

CRC error: the receiver calculates the CRC in the same way than the emitter, a CRC error is set if the calculated result is not the same than in CRC sequence;

Form error : detected when a fixed form bit field contains one or more illegal bits

Ack error : detected by a transmitter whenever it does not monitor a dominant bit during the ack slot.

Two error counters are implemented, a receive error counter, a transmit error counter

Counter features:

The transmit error is incremented if a transmission error occurs.

The receive error is incremented if a reception error occurs.

The counter decreases if correct message, it increases if erroneous message

The counting is proportional, fast increase (8 unities if bit error), low decrease

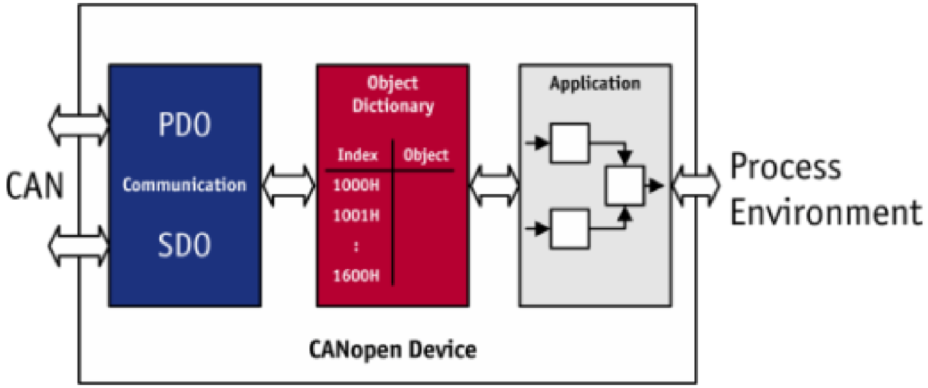
Error active definition:

If both counters are between 0 and 127, the node goes in error mode active. The functionality is working, but in case of error, there is a transmission of active error flag during error frame.

Error passive definition:

If one the counters are between 128 and 255 the node goes in passive error mode. The functionality is working, but in case of error, there is a transmission of passive error flag during error frame.

CAN OPEN

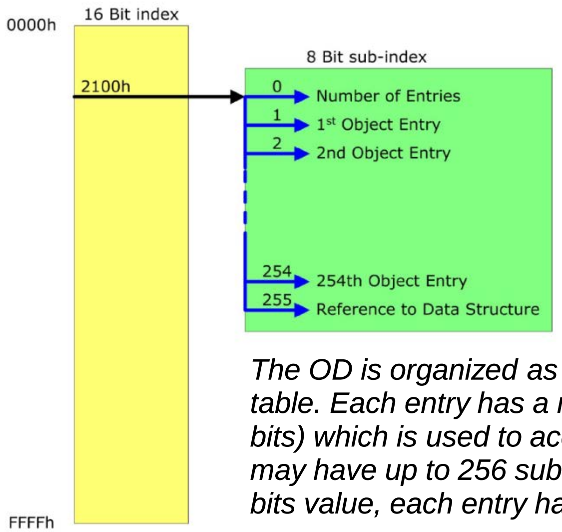


CAN open is a standardized application for distributed automation systems based on CAN (Controller Area Network) offering the following performance features:

- Transmission of time-critical process data according to the producer consumer principle
- Standardized device description (data, parameters, functions, programs) in the form of the so-called "object dictionary". Access to all "objects" of a device with standardized transmission protocol according to the client-server principle
- Standardized services for device monitoring (node guarding/heartbeat), error signalization (emergency messages) and network coordination ("network management")
- Standardized system services for synchronous operations (synchronization message), central time stamp message
- Standardized help functions for configuring baud rate and device identification number via the bus
- Standardized assignment pattern for message identifiers for simple system configurations in the form of the so-called "predefined connection set"

CAN OPEN

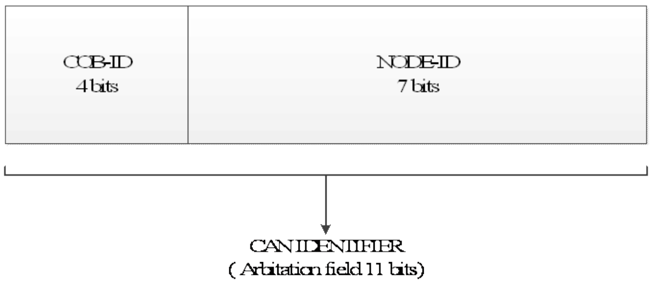
Index	Object	
0000h	Not Used	Common to any Device
0001h – 025Fh	Data Type	Common to any Device
0260h – 0FFFh	Reserved	Common to any Device
1000h – 1FFFh	Communication Profile Area	Common to any Device
2000h – 5FFFh	Manufacturer Specific Profile Area	Device Specific
6000h – 9FFFh	Standardized Device Profile Area	Device Specific
A000h – AFFFh	Network Variables	Device Specific
B000h – FFFFh	Reserved for Future Use	



The OD is organized as a collection of entries like a table. Each entry has a number called an index(16 bits) which is used to access the entry , each entry may have up to 256 sub entries , reference using a 8 bits value, each entry has at least one subentry.

- **object dictionary** : The object dictionary is like a table that holds all network accessible data and each CAN open node must implement its own object dictionary (OD). **The OD contains a description of the CAN open configuration and functionality, and may be read and written to by other CAN open nodes.**
- **data types objects**: contain the different used data types (integer32, char , float32, complex structures…..)
- **Communication objects:** contain data allowing to configure how the equipment will communicate on the network, (exchanged data , consumer server protocol, synchronization activation…..
- **Manufacturer specific objects :** objects which be defined by the manufacturer in order to store parameters , specific data …..
- **Standardized objects :** contain applicative data which can be exchanged on the network. The object content can be standardized if the object is in accordance with a profile area.

CAN OPEN object dictionary access



CAN identifier	Use	Comment
000h	NMT NETWORK MANAGEMENT	
001	Global Failsafe Command	
80	SYNC	
81-FF	Emergency	80+NODEID
181-1FF	Transmit PDO	180+NODEID
201-27F	Receive PDO	200+NODEID
----	----------	-------
581-5FF	TRANSMIT SDO	580+NODEID
601-67F	RECEIVE SDO	600+NODEID

Highest priority → lowest priority

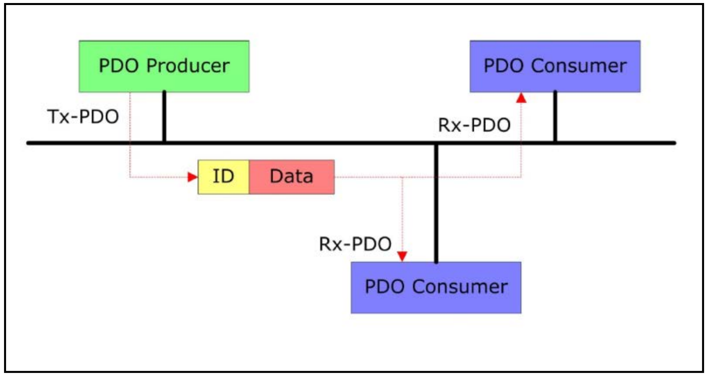
The COB-ID is the communication object Identifier and is coded on 4 bits

The Node-ID is the node identifier is coded on 7 bits, so allow having 127 nodes (node identifier 0 is used for broadcast)

Object	Function code	COB –ID		Comm parameters at OD index (hexa)
		Calculation (hexa)	Range ID	
EMERGENCY	0001	080 + Node ID	081 - 0FF	1024, 1015
PDO 1 (transmit)	0011	180 + Node ID	181 - 1FF	1800
PDO 1 (receive)	0100	200 + Node ID	201 - 27F	1400
PDO 2 (transmit)	0101	280 + Node ID	281 - 2FF	1801
PDO 2 (receive)	0110	300 + Node ID	301 - 37F	1401
PDO 3 (transmit)	0111	380 + Node ID	381 - 3FF	1802
PDO 3 (receive)	1000	400 + Node ID	401 - 47F	1402
PDO 4 (transmit)	1001	480 + Node ID	481 - 4FF	1803
PDO 4 (receive)	1010	500 + Node ID	501 - 57F	1403
SDO (transmit/server)	1011	580 + Node ID	581 - 5FF	1200
SDO (receive/client)	1100	600 + Node ID	601 - 67F	1200
NMT Error Control	1110	700 + Node ID	701 - 77F	1016, 1017

CAN open Process Data object (1/2)

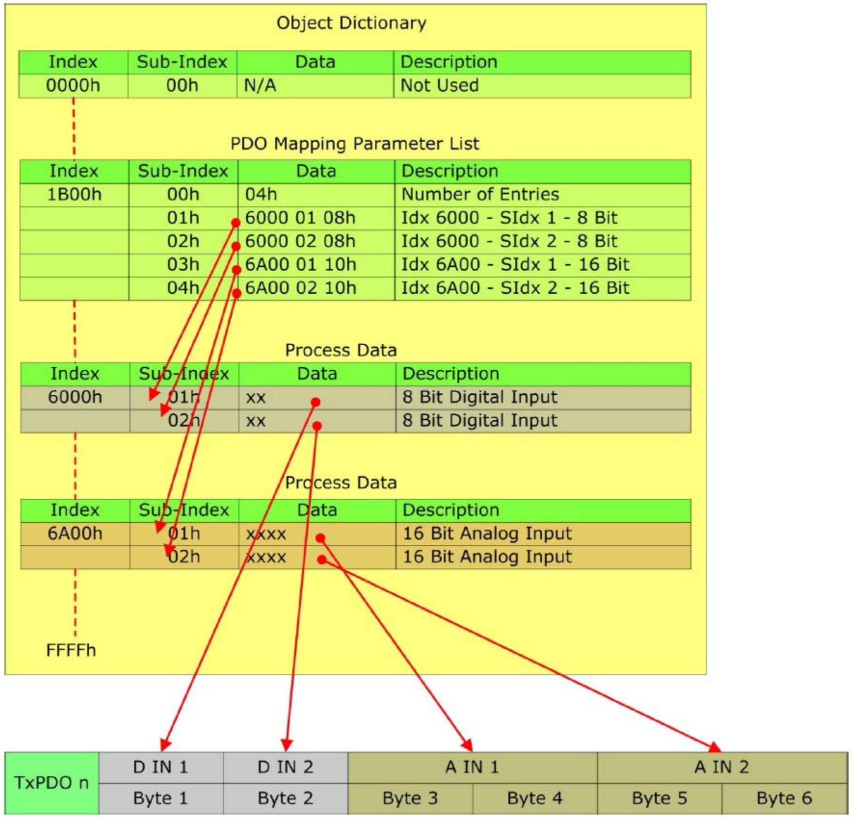
Goal of PDO is to give the possibility for a node to transmit their data whenever they want and to place multiple process data variables into a single message.



For each PDO in a system there is only one node producing it, and for that node this PDO is a TPDO, there are also one or several nodes that receive and consume the PDO, for all node consuming it, the PDO is a RPDO.

The PDO is used for real time transmission of process data. The transfer is limited to max 8 bytes, the definition of data is described per "PDO mapping", the transfer is unconfirmed, there is 1 CAN identifier per PDO

CAN open Process Data object (2/2)



There are 4 major transmit triggers methods in CAN open :

Event driven : PDO is transmitted on occurrence of an event , e.g. change of input

Time driven: triggered by a time periodic event

Individual polling : PDO is transmitted only upon request from a remote device

Synchronized ,group polling: PDO is only transmitted upon reception of a SYNC message

PDO mapping provides the description of process data to be transmitted or received within a PDO.

The PDO mapping specifies how the data is mapped in a message.

Process data which is transmitted or received within a PDO is specified in a “Mapping Parameter List” in form of a reference (Index, Sub-Index) in the Object Dictionary.

Each PDO has its own “Mapping Parameter List” which can also be accessed through the Object Dictionary

CAN open SDO communication

3 Major communication modes:

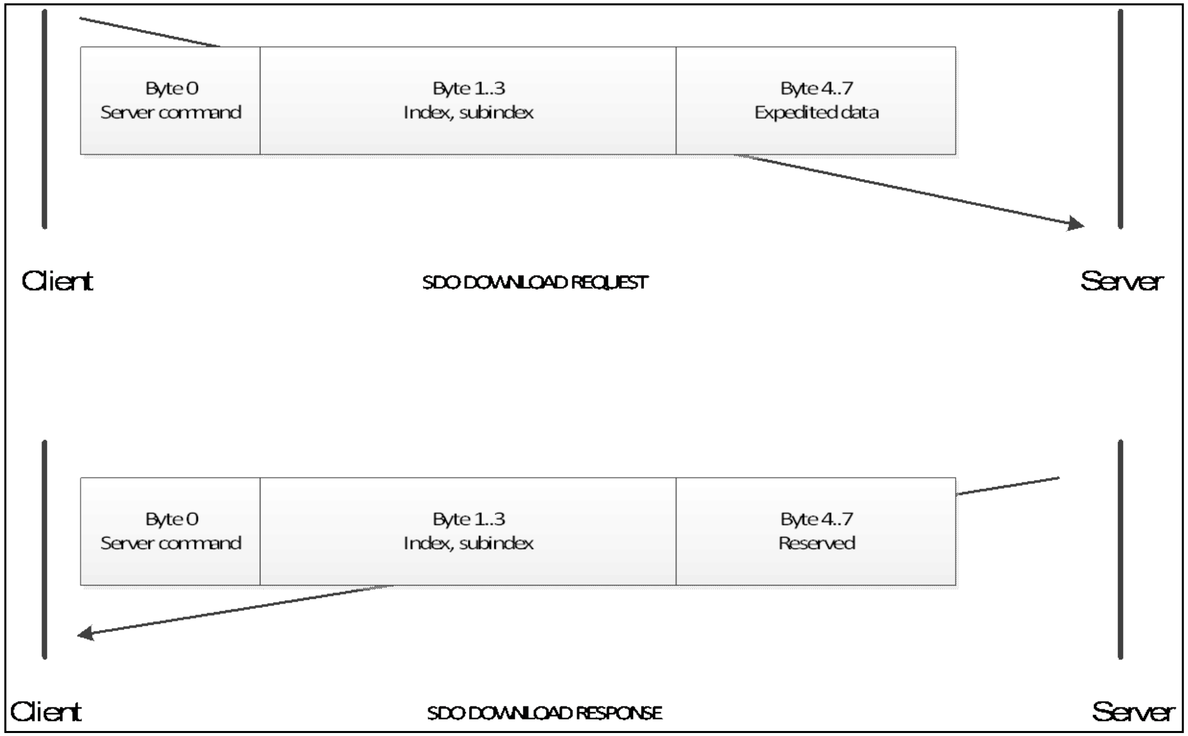
Expedited transfer: up to 4 bytes which can be directly embedded in a SDO request or response, suitable for access to OD entry

Segmented transfer: allows for transmission bigger than 4 bytes

Block transfer :optimized transfer for OD entries that contain large amounts of data , up to 889 bytes segmented into 127 messages of 7 bytes

CAN open SDO communication

Expedited transfer download



Initiation command

Byte 0 :

Client Comm Specifier = 1

Nb data

Byte 1..3:

 OD index and sub-index this write should go to

Byte 4..7 :

Data bytes

Server response

Byte 0:

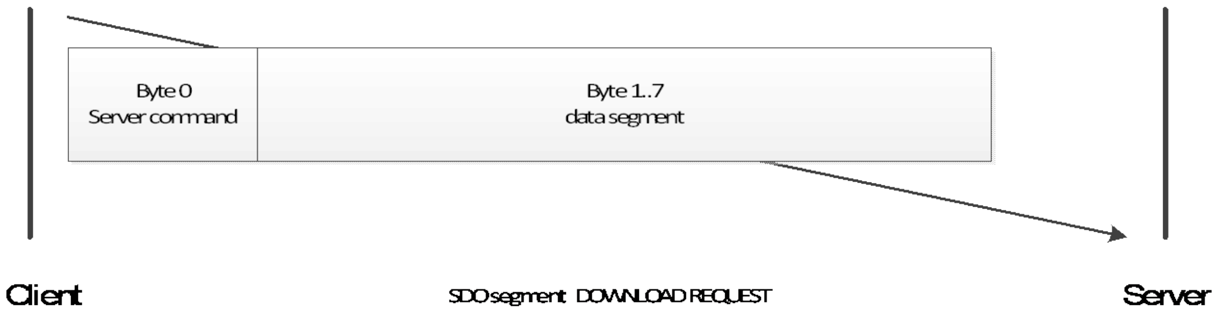
Server Command specifier = 3

Byte 1..3:

Index and sub-index that received the wriet access

CAN open SDO communication

Segmented transfer download



When the initiation sequence has been negotiated this message is used to transmit the next sequence

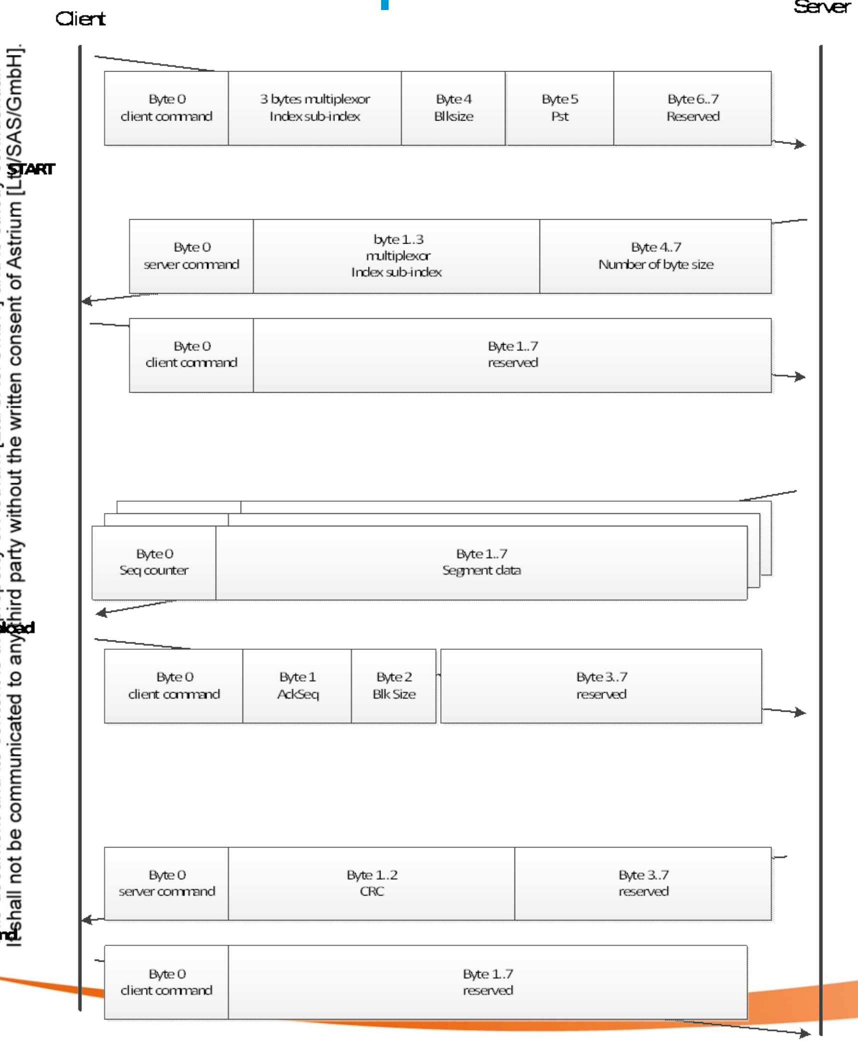
Byte :0 ccs = 0

Byte 1..7: data segment

For each transfer there is a response

Byte 0 : scs = 1

CAN open SDO communication



Block transfer upload

Initiation command :

Byte 0 : Ccs = 6; size indicator

Byte 1..3 : index and sub-index of OD entry the client wants to read

Byte 4: number of segment per block

Response from server gives the index and sub-index and number of bytes that need to be transmitted.

Client sends the command to start the transfer

Server sends block until end of transfer

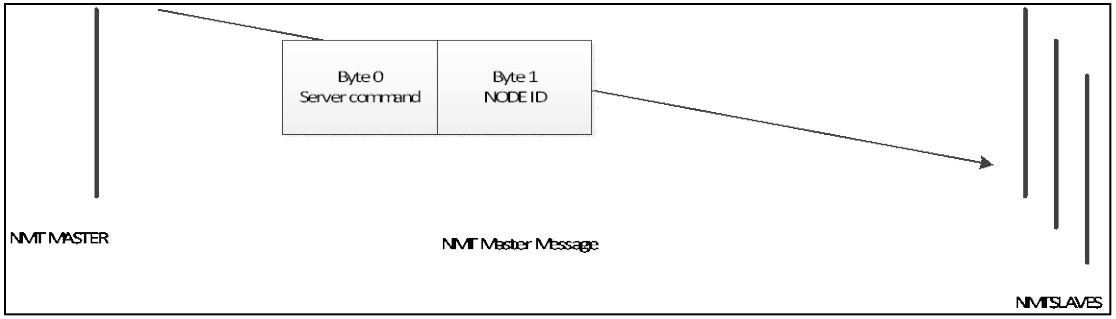
Client can (if programmed) send the number of segement acknowledged , the server must re-transmit those that are not acknowledged

Message of server ‘confirmation at the end of upload block

Client confirmation that the transfer is finished

CAN open Network management communication

NMT Master message

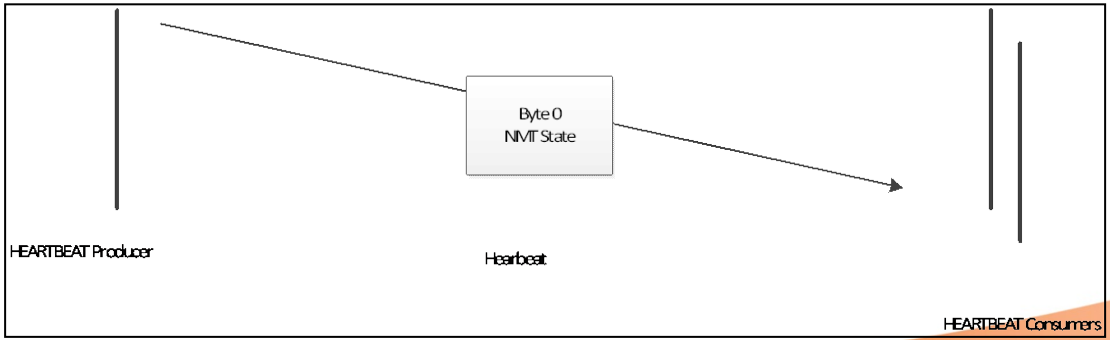


The NMT message has the CAN message identifier 0 and contains 2 bytes. All slave nodes must be able to receive this message and act upon its content

The byte 0 commands to switch in a specific NMT state .

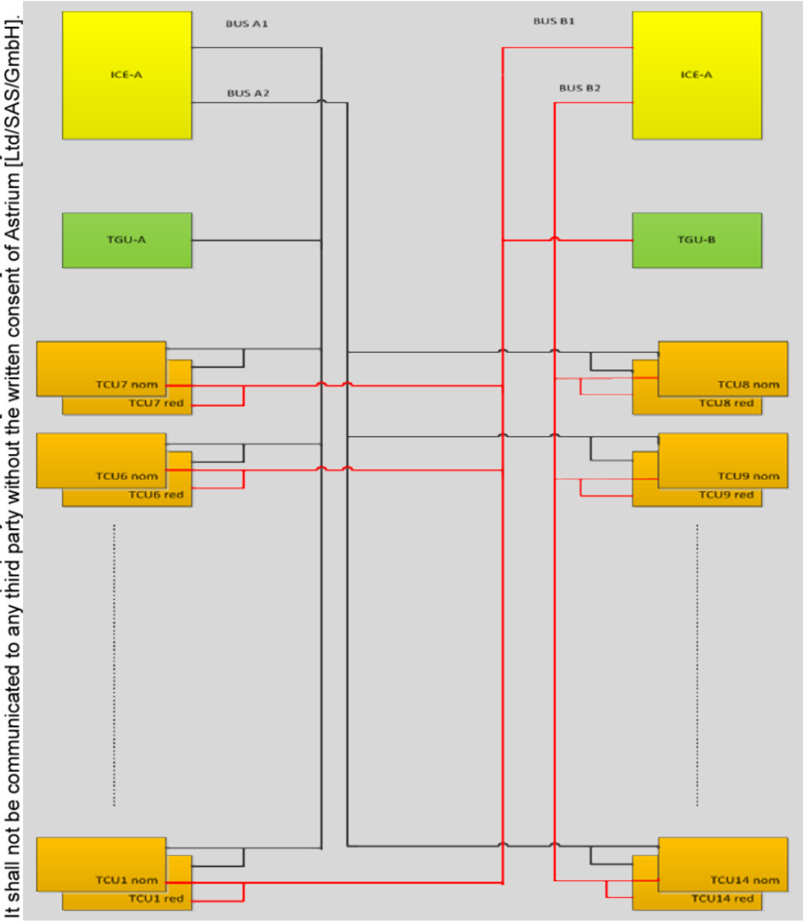
Byte 1 addresses all nodes if 0 or a specific node ID

HeartBeat



The heartbeat message sent by a node has the CAN message identifier 700 plus the node ID , it only contains one byte showing the NMT state of that node

Can Bus on sentinel 1



The communication between SES-ICE (integrated control unit) and TCU (thermal compensation units) is done through ACB (Antenna Control bus) which is a **CAN 2-0B bus**.

Additionally SES-TGU (Tx Gain Units) is also connected to ACB**.**
The transmission rate is 500 Kbits/s

4 types of messages are provided :

Data frame: carries data from a transmitter to the receivers

Remote Frame: transmitted by a unit to request the transmission of a Data Frame

Error Frame: transmitted by any unit upon detection of a bus error.

Overload Frame: used to provide for an extra delay between the preceding and the succeeding Data Frame or Remote Frame.

CAN arbitration field on sentinel 1

BASE ID											Extended ID																	
Bit 28	Bit 27	Bit 26	Bit 25	Bit 24	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Function code				NODE ID = Terminal Address							Spare		Message type					Spare		Buffer address					Spare			

Function code	Value	Priority	function
TM/TC Protocol	1000	3	Protocol for transmission of PUS Telecommand and Telemetry Packets between ICM and TCUs
TGU Protocol	0100	2	Protocol for transmission of messages between ICM and TGU
Time Distribution Protocol	0010	1	Protocol for setting of Instrument Onboard Time in TCUs

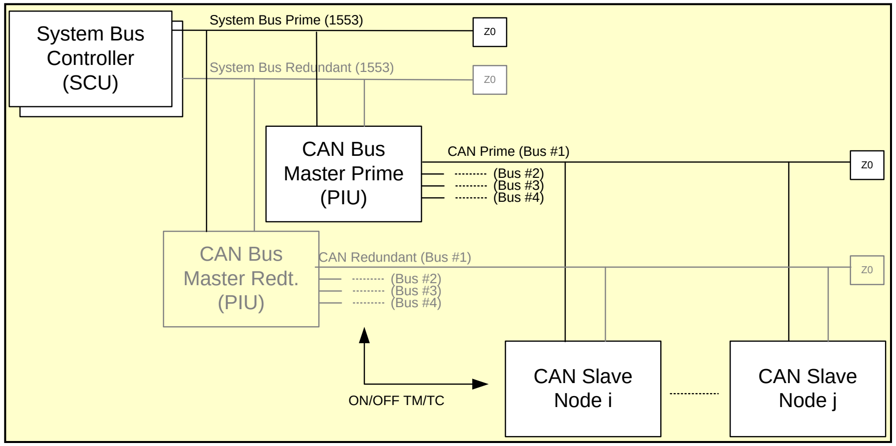
Function code	Value	Function	size
ICM-T1/TCU-R1	00001	Buffer for storage of one TC Packet, which is transferred by a series of Transfer TC Messages	512 Octets
ICM-T2/TCU-R2	10110	Buffer for storage of a Complete TC Message, which triggers the execution of a TC	8 Octets
ICM-T5/TCU-R5	00100	Buffer for storage of a Time Code Message	8 Octets
ICM-T3/TCU-R3	01011	Buffer for storage of a Request BSA Message, which requests transmission of status data	8 Octets

Node ID	Value (bin)
ICM	1000001
TGU	1000010
TCU1	1010001
TCU2	1010010
TCU3	1010011
TCU4	1010100
TCU5	1010101
TCU6	1010110

Message type	Value
SCET(Time code)	00100
TGU status	00111
TC packet	01000
TM packet	01100

-The Function Code field is the most significant field for bus medium arbitration. Therefore all messages transferred using this protocol will have a well-defined priority level. This layer has the highest priority.
-The 7 bits determine the node Id (or terminal address), the terminal address shall always be the destination message. The value indicates a second level of priority. The ICM has a higher priority than TCUx if TM/TC protocol messages wants to have access to bus at same time
-The buffer address determines the buffer in which the data is to be transferred, (it is quite similar to 1553 sub-adresses) there are currently 12 buffers addresses defined listed below

CAN open use on telecom



- Requirements

LSSB suffer from some limitations. The bus is limited to a maximum length of less than 8 meters and a data rate of between 8 and 16kbps. LSSB utilizes differential signaling and has 5 signal/clock lines each requiring their own twisted pair. This means that a total of 10 lines, excluding power and ground, are required for LSSB operation. Furthermore the maximum number of nodes which can be connected to the host using a single LSSB is 32 and these nodes must be connected in a daisy chain network.

The purpose of this serial asynchronous bus is to allow serial data transfer between one bus *Master* (MPIU) to several *Nodes* or *Slave* payload equipment's (Channels Amplifiers, Antenna (CAMP) Pointing Mechanism Electronics (APME), Centralized Power Supply Unit (CPSU)…).

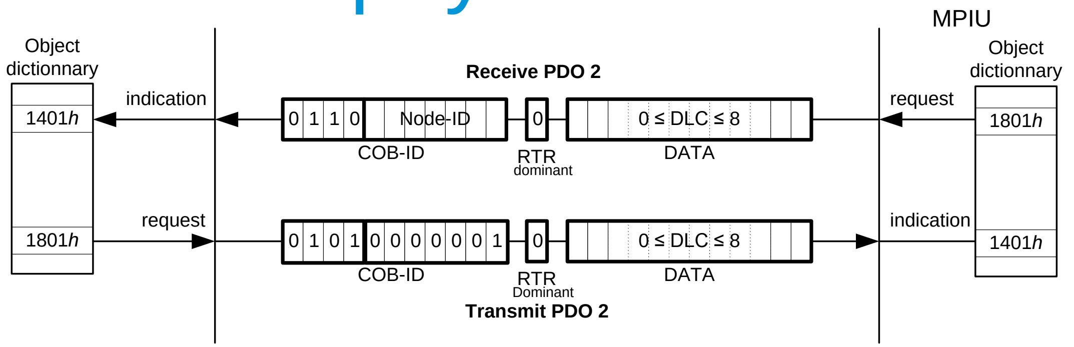
A maximum of 63 slaves nodes are connected to a CAN bus in addition to the CAN bus Master.

CAN open use on telecom payload

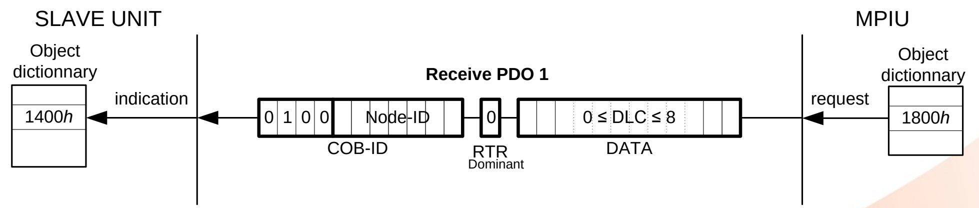
- The present activity on E3000 is to change LSSB bus to CAN open bus with a minimum of modifications
- Complexity will be integrated step by step in the system, so in a first phase, only telecommands and telemetries will be adapted to CAN open
- The functional mode is master/slave mode and it will not be changed, the slaves do not generate a large amount of data, the LSSB bandwidth is lesser than CAN open , so a secure functionality which respect the standard and the previous functions is sufficient
- To respect previous requirements and remarks:
  - Network management is not used
  - SDO communication mode is not used or limited to ground for integration
  - Remote request mode is not encouraged
  - Sync protocol is not authorized
  - Heartbeat protocol is disabled by default
  - Emergency objects are not produced by slaves

CAN open use in telecom payload

- TM request is sent by MPIU to bus terminal
  - It is data frame containing a function code (one of the four PDO receive, the slave node ID, 0 to 8 bytes of data
- DATA transmission follows immediately
  - Data transmission contains (a function code (one of the four PDO transmit, the master node (01 for nominal n, 2 for redundant),0 to 8 bytes of data



- DATA transmission is sent MPIU to slave node
  - Data transmission contains (a function code (one of the four PDO receive, the slave node ID, 0 to 8 bytes of data
  - The service based on RPDO is used by the CAN master to send **unconfirmed command** to one slave unit.



The exchanges between PIU and payload equipments are defined with a Master/Slave approach. The PIU is the Master of the CAN bus and initiates all exchanges.

CAN open on EXOMARS

- Network Management (NMT) Communication follows a typical Master/Slave approach.
  - Talking about masters and slaves in a network implies that the master has some sort of control function over the slaves. Typically this includes shutdown and/or reset of single nodes or the entire network. Once the master/ slave relationship is established, the direction of control is always from the master to the slave(s).
- Service Data Objects (SDO) follow a Client/Server model
  - In a client/server environment the server provides “services” to the network. A typical service could be serving data access points (inputs and outputs) to the network. A client is a network node making use of these services. Whether a network module becomes a server or client is completely unrelated to its status as a master or slave. The master implements some client and server functionalities.
- Process Data Objects (PDO), that will typically embed periodic status data transfers, are implemented as a Producer/ Consumer model.
  - A producer transmits data to the network and a consumer receives data from the network. For a specific set of data there can only be one producer and there is at least one, but possibly multiple consumers for that data.
- Heartbeat protocol (HB) which monitors the network state also follows a Producer/Consumer model.

CAN open on EXOMARS functionality

- CAN BUS data exchanges 1Mbits/s
  - 2 CAN bus interfaces
    - platform Bus
    - Payload Bus

- CAN management function shall handle for each CAN bus slave device:
  - A device name
  - An associated bus (platform or payload)
  - A base node id (range 1...127) •
  - A node mask. The number of nodes supported by the device (node id count) is “128 - node_mask”, having consecutive numbers starting from the base node id)
  - The number of PDOs supported by the device (4 per node => up to 128 PDO for a 32 nodes device)

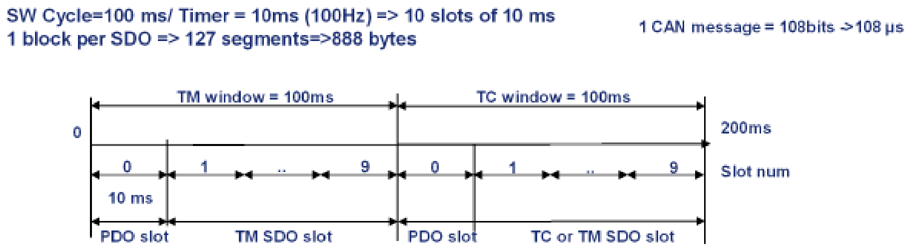
- ## Object dictionary
  - The CAN management function shall manage an internal private and non-dynamically built representation of the object dictionary based on an EDS template.

- ## COB-ID
  - The CAN management function shall send COB-IDs on standard CAN 2.0A 11-bit identifiers

CAN open on EXOMARS Framework

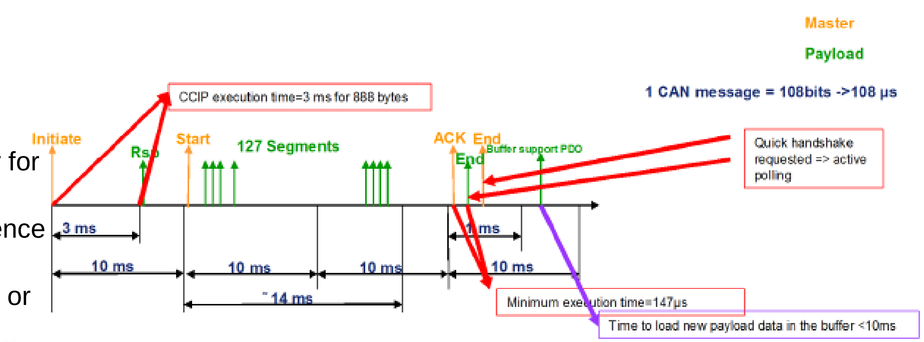
SYNC Message at slot 0

- HK is received during PDO slot in response to SYNC message
- A SDO slot allows to process 2 transfer blocks SDO at a time (upload or download) a transfer block SDO is up to 127 segments in 40 ms
- Network management TC (NMT) cyclic commands (TC) are sent in the TC slot NMT/HB capability at 10 Hz
- HeartBeat messages can be received at any slot
- If expedited SDO are requested , they also use the SDO slot



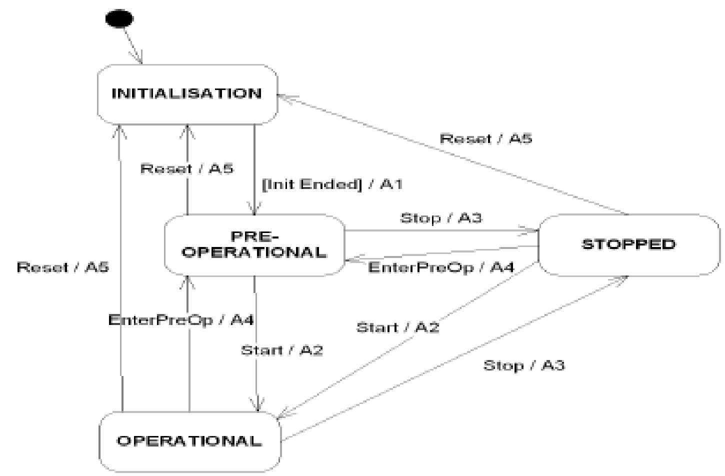
Example of SDO block download

- The CAN management function shall process asynchronous slave TPDO with index 0x6001 from any base node id as a Buffer Support PDO notifying either the successful completion of the previous SDO download command or the request for an SDO upload.
- 6001h (index)
- Subindex
  - 0 : Number of sub-entries
  - 1 Buffer status: 0x1: Ready for uploading 0x2: Ready for downloading
  - 2 SDO content: 0x0 HK data 0x1 Dump data 0x2 Science data
  - 3 Size: number of bytes to be transferred, via a single or multiple SDO blocks transfers



CAN open on EXOMARS function availability

- In INITIALISATION state, the CAN management function shall configure the CAN controller to work at a predefined bit-rate of 1 Mbits/s, and send the boot-up object (first HB then automatically enter the PRE-OPERA-TIONAL state. In INITIALISATION state, the CAN management function shall ignore incoming messages.
- In PRE-OPERATIONAL state, the CAN management function shall start the SYNC producer service, and authorize SDO and NMT management.
- In STOPPED state, the CAN management function shall disable the SYNC producer service, the SDO service and the PDO service. (only NMT management is allowed, inc. HB
- In OPERATIONAL state, the CAN management function shall start the PDO service and enable the SDO and SYNC services if they are disabled).



The CAN management function shall support the following NMT services and protocols:

- Start Remote Node
- Stop Remote Node
- Enter Pre-Operational
- Reset Node
- Reset Communication