CAN-TS protocol

Protocol for memory constrained embedded application

CAN in Space 2019, Gothenburg/Sweden

Dejan Gačnik (CTO)

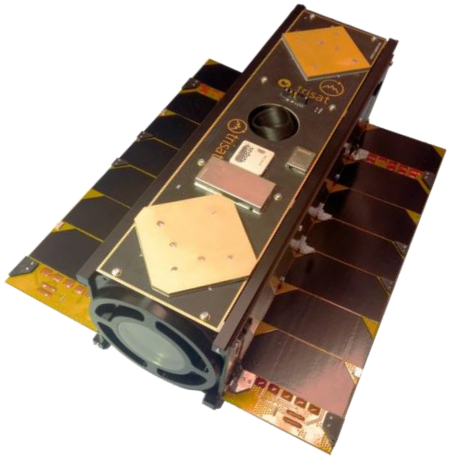
skylabs

Motivation

- A need for a lightweight CAN application layer protocol
- Implementation to fit also 16-/8-bit MCU architectures
- Highly efficient (*raw* vs. *payload* data rate)
- Services to fits satellites requirements

CAN-TS Maturity

- First protocol specification released in 2012 v1.0 for ESMO mission
- 2019 v1.4 released and becomes a open source
- Flight heritage CAN-TS v1.2 on board of a TRISAT satellite (launch Q3/2019)
  - Hot redundant CAN bus configuration (v1.2 specific)
- picoRTU distributed remote terminal unit
  - CAN-TS support v1.4
  - ECSS CANbus extension protocol ECSS-E-ST-50-15C supported


TRISAT satellite PFM

picoRTU system EM

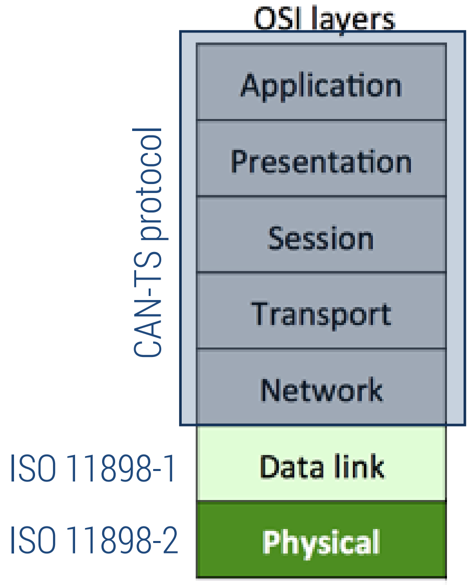
skylabs

CAN-TS

- CAN-TS is communication protocol and device services specification for embedded systems used in space applications.

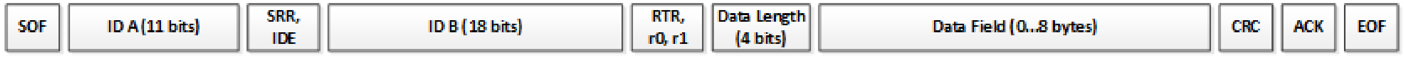
- CAN-TS implements the layers above and including the network layer (OSI reference model).

- The CAN-TS protocol defines an addressing scheme and several memory efficient communication services are support
  - Fully compliant with CAN Data link layer ISO 11898-1

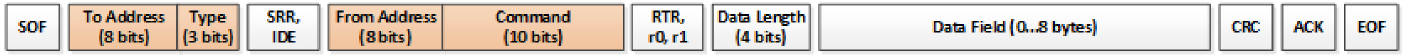


CAN-TS Frame

- CAN-TS exploits CAN2.0B extended frame
    - To / From address, Transfer type and Command ID encapsulated in 29-bit CAN identifier
    - Origin validation: message acceptance additionally validated by source address (implementation specific)
- Considered use of CAN ID dominances
    - Up to 255 nodes
    - User defined broadcast messages (Time SYNC address = 0)
- Software based acknowledgment
    - Assurance of SW message processing
    - Higher reliability trade-off against bus utilization
- Efficient data field usage
    - Complete 8 bytes of data field available for protocol services.



Extended frame CAN 2.0B



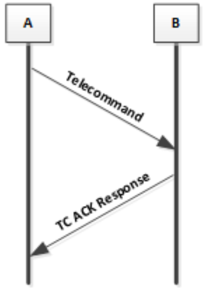
CAN-TS frame

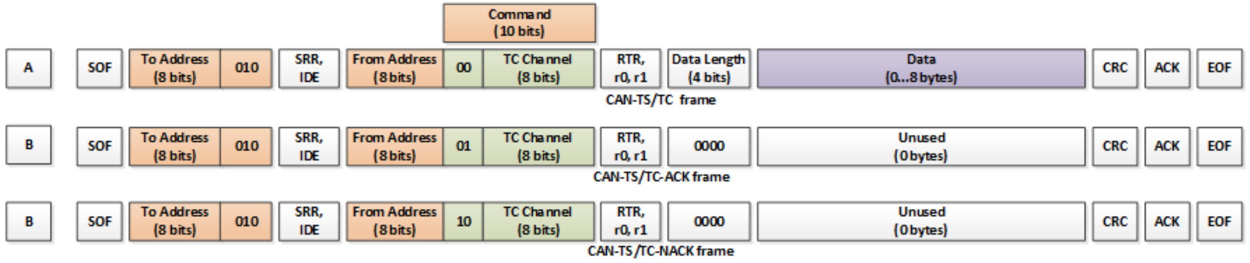
CAN-TS Transfer type

Services

CAN-TS supports 6 application layer services (transfer types)

- Telecommand and Telemetry as single-message transfers
  1. **Telecommand**
     - Acknowledged transfer (request/acknowledge)
     - Single TC Response has up to 8 bytes of payload data
     - Up to 256 TCs per device





						Command (10 bits)							
A	SOF	To Address (8 bits)	010	SRR, IDE	From Address (8 bits)	00	TC Channel (8 bits)	RTR, r0, r1	Data Length (4 bits)	Data (0...8 bytes)	CRC	ACK	EOF

CAN-TS/TC frame

B	SOF	To Address (8 bits)	010	SRR, IDE	From Address (8 bits)	01	TC Channel (8 bits)	RTR, r0, r1	0000	Unused (0 bytes)	CRC	ACK	EOF

CAN-TS/TC-ACK frame

B	SOF	To Address (8 bits)	010	SRR, IDE	From Address (8 bits)	10	TC Channel (8 bits)	RTR, r0, r1	0000	Unused (0 bytes)	CRC	ACK	EOF

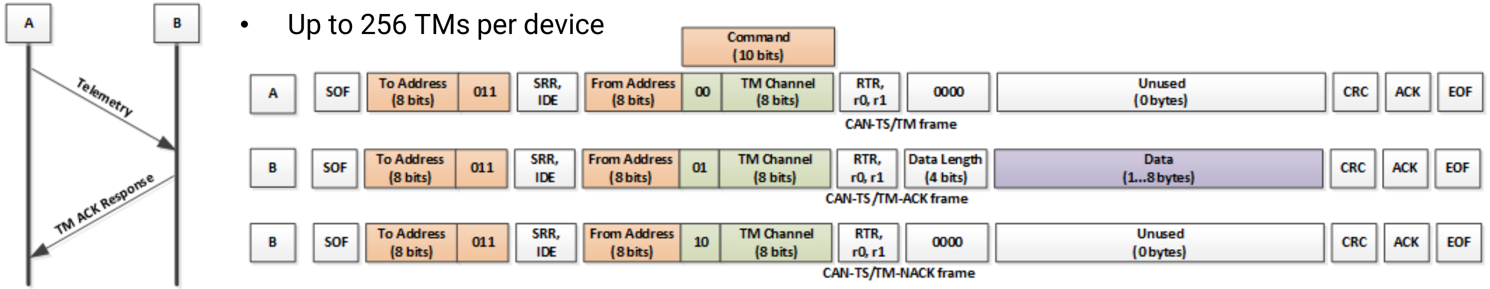
CAN-TS/TC-NACK frame

CAN-TS Transfer type

Services (cont'd)

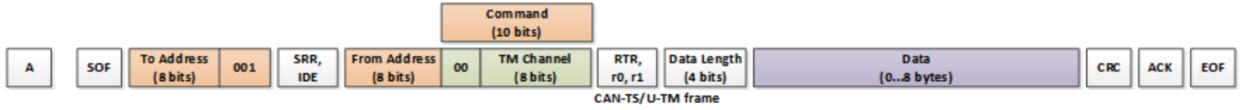
2. Telemetry

- Acknowledged transfer (request/acknowledge)
- Single TM Request has up to 8 bytes of payload data
- Up to 256 TMs per device


A
B
Telemetry
TM ACK Response
Command (10 bits)
A | SOF | To Address (8 bits) | 011 | SRR, IDE | From Address (8 bits) | 00 | TM Channel (8 bits) | RTR, r0, r1 | 0000 | Unused (0 bytes) | CRC | ACK | EOF
CAN-TS/TM frame
B | SOF | To Address (8 bits) | 011 | SRR, IDE | From Address (8 bits) | 01 | TM Channel (8 bits) | RTR, r0, r1 | Data Length (4 bits) | Data (1...8 bytes) | CRC | ACK | EOF
CAN-TS/TM-ACK frame
B | SOF | To Address (8 bits) | 011 | SRR, IDE | From Address (8 bits) | 10 | TM Channel (8 bits) | RTR, r0, r1 | 0000 | Unused (0 bytes) | CRC | ACK | EOF
CAN-TS/TM-NACK frame

3. Unsolicited Telemetry

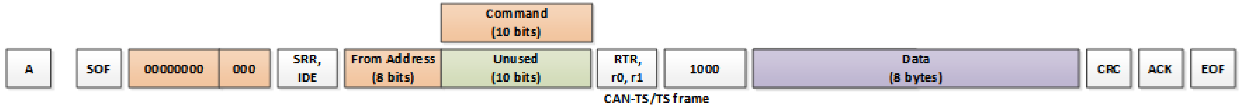
- TM message send directly, without request
- Periodic transfer (keep-alive for redundancy mgmt., HSK TM,...) or event driven (warning limit,...)
- Single Un. TM message up to 8 bytes of payload data
- Point to point (e.g. Cycling through HSK TM) or Broadcast transfer


Command (10 bits)
A | SOF | To Address (8 bits) | 001 | SRR, IDE | From Address (8 bits) | 00 | TM Channel (8 bits) | RTR, r0, r1 | Data Length (4 bits) | Data (0...8 bytes) | CRC | ACK | EOF
CAN-TS/U-TM frame

CAN-TS Transfer type

Services (cont'd)

4. **Time Synchronisation**
    - Distribution of time (time format is mission specific)
    - Unsolicited telemetry message broadcasted by time master
    - Broadcast and highest priority message (To address = 0)

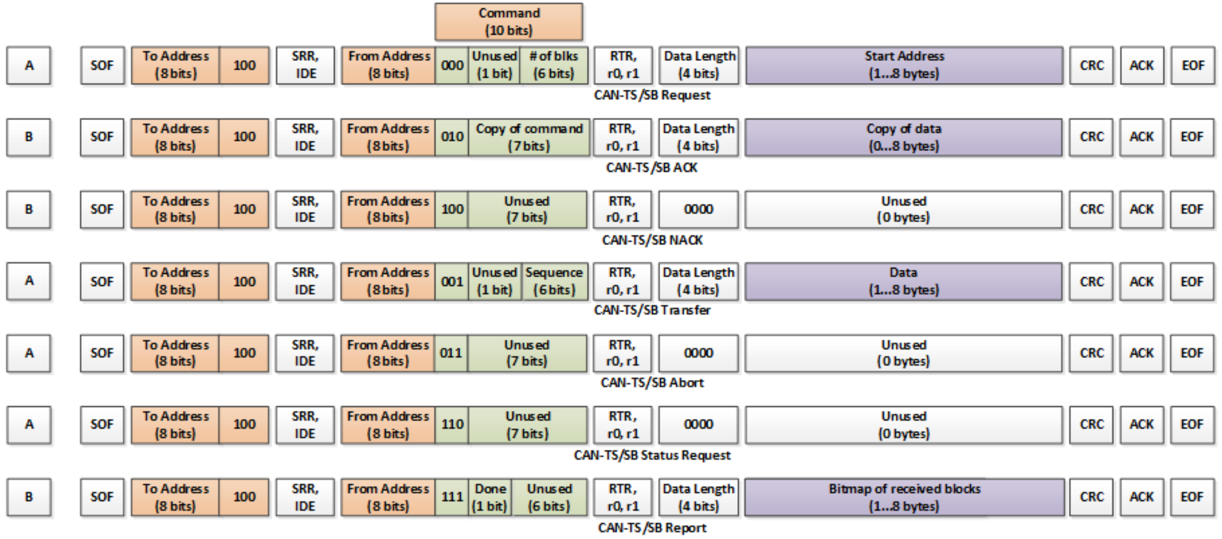


- Set and Get Block as multi-message transfers
    - Suited for large data exchanges between nodes (addressable space up to 64-bit)
    - Acknowledged transfer type with re-transmission capability
    - Abort mechanism and predetermined timeout interval
    - Up to 512 bytes per transfer cycle
    - Address ranges from 8-bit up to 64-bit address width

CAN-TS Transfer type

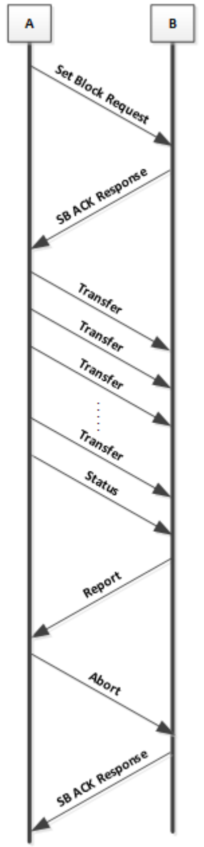
Services (cont'd)

5. **Set Block**
   - Used to transfer larger blocks of data **from source to sink**
   - Data reconstruction possible even if frames are received out of order
   - Sink side tracks time from last received valid frame, if time exceeds predefined time, sink shall close session (Abort message).



Command (10 bits)

A	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	000	Unused (1 bit)	# of blks (6 bits)	RTR, r0, r1	Data Length (4 bits)	Start Address (1...8 bytes)	CRC	ACK	EOF	CAN-TS /SB Request
B	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	010	Copy of command (7 bits)		RTR, r0, r1	Data Length (4 bits)	Copy of data (0...8 bytes)	CRC	ACK	EOF	CAN-TS /SB ACK
B	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	100	Unused (7 bits)		RTR, r0, r1	0000	Unused (0 bytes)	CRC	ACK	EOF	CAN-TS/SB NACK
A	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	001	Unused (1 bit)	Sequence (6 bits)	RTR, r0, r1	Data Length (4 bits)	Data (1...8 bytes)	CRC	ACK	EOF	CAN-TS/SB Transfer
A	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	011	Unused (7 bits)		RTR, r0, r1	0000	Unused (0 bytes)	CRC	ACK	EOF	CAN-TS/SB Abort
A	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	110	Unused (7 bits)		RTR, r0, r1	0000	Unused (0 bytes)	CRC	ACK	EOF	CAN-TS/SB Status Request
B	SOF	To Address (8 bits)	100	SRR, IDE	From Address (8 bits)	111	Done (1 bit)	Unused (6 bits)	RTR, r0, r1	Data Length (4 bits)	Bitmap of received blocks (1...8 bytes)	CRC	ACK	EOF	CAN-TS /SB Report

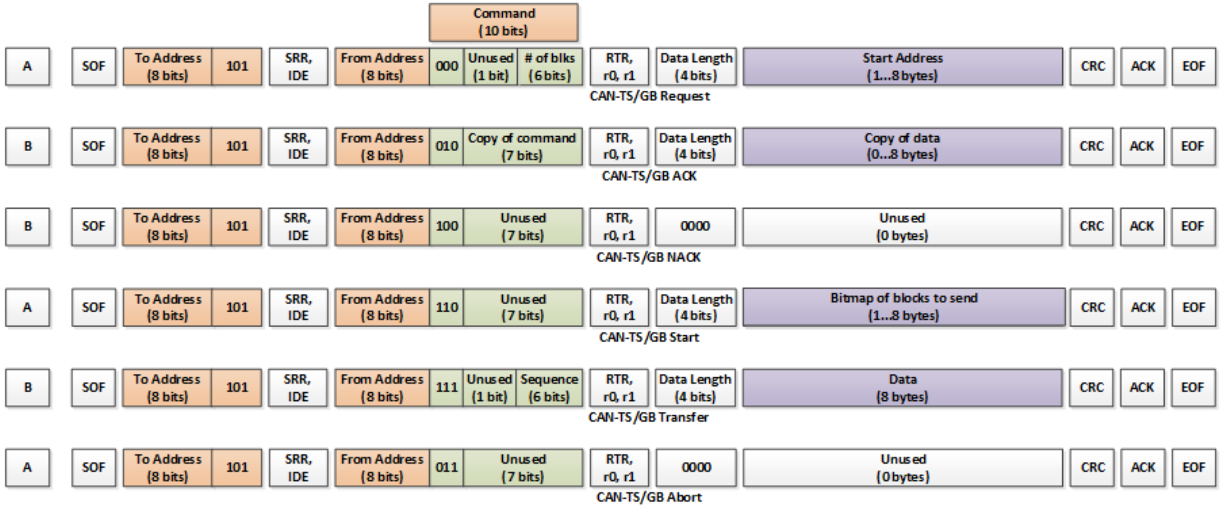


CAN-TS Transfer type

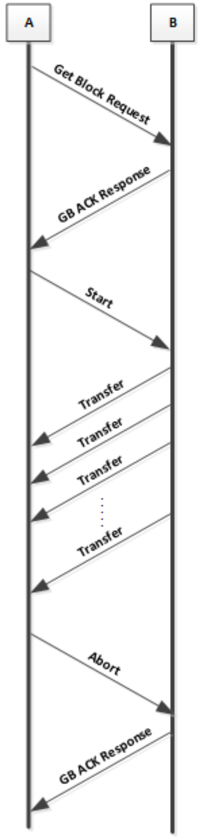
Services (cont'd)

6. Get Block

- Used to transfer larger blocks of data **from sink to source**
- Data reconstruction possible even if frames are received out of order
- Sink side tracks time from last received valid frame, if time exceeds predefined time, sick shall close session (Abort message).

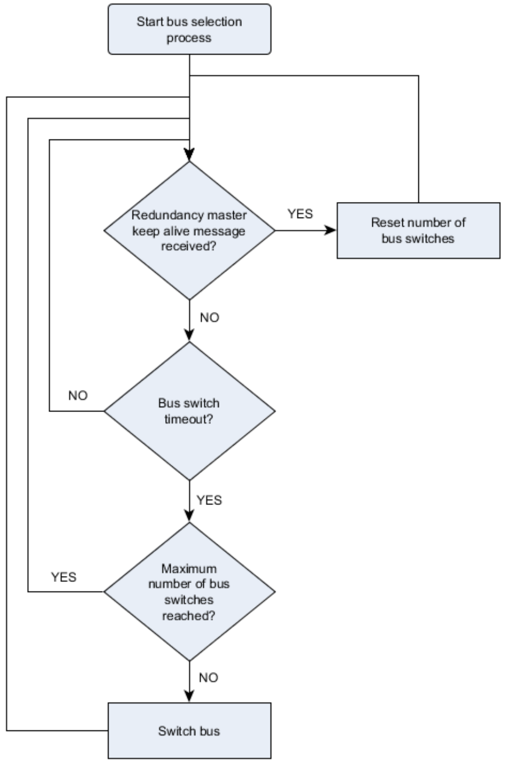


Command (10 bits)
A | SOF | To Address (8 bits) | 101 | SRR, IDE | From Address (8 bits) | 000 | Unused (1 bit) | # of blks (6 bits) | RTR, r0, r1 | Data Length (4 bits) | Start Address (1...8 bytes) | CRC | ACK | EOF
CAN-TS/GB Request
B | SOF | To Address (8 bits) | 101 | SRR, IDE | From Address (8 bits) | 010 | Copy of command (7 bits) | RTR, r0, r1 | Data Length (4 bits) | Copy of data (0...8 bytes) | CRC | ACK | EOF
CAN-TS/GB ACK
B | SOF | To Address (8 bits) | 101 | SRR, IDE | From Address (8 bits) | 100 | Unused (7 bits) | RTR, r0, r1 | 0000 | Unused (0 bytes) | CRC | ACK | EOF
CAN-TS/GB NACK
A | SOF | To Address (8 bits) | 101 | SRR, IDE | From Address (8 bits) | 110 | Unused (7 bits) | RTR, r0, r1 | Data Length (4 bits) | Bitmap of blocks to send (1...8 bytes) | CRC | ACK | EOF
CAN-TS/GB Start
B | SOF | To Address (8 bits) | 101 | SRR, IDE | From Address (8 bits) | 111 | Unused (1 bit) | Sequence (6 bits) | RTR, r0, r1 | Data Length (4 bits) | Data (8 bytes) | CRC | ACK | EOF
CAN-TS/GB Transfer
A | SOF | To Address (8 bits) | 101 | SRR, IDE | From Address (8 bits) | 011 | Unused (7 bits) | RTR, r0, r1 | 0000 | Unused (0 bytes) | CRC | ACK | EOF
CAN-TS/GB Abort

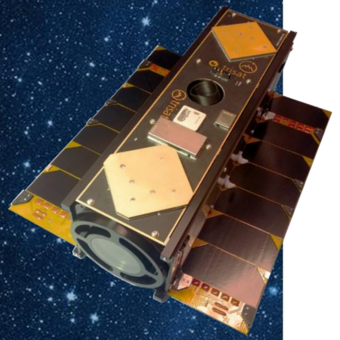


CAN-TS Bus redundancy

- Bus redundancy management is based on Keep Alive messages (To address =1).
    - Unsolicited telemetry messages sent to address 1.
    - Telemetry channels used by keep alive messages are node specific. They are usually selected in a way that any other module knows module health just by listening to keep alive messages.
- Redundancy management is based on 3 mission specific parameters (inherited approach as ECS-E-ST-50-15C):
    - **Keep alive interval** – Amount of time between two consecutive keep alive messages sent by node.
    - **Bus switch timeout** – Amount of time to wait on one bus before switching to another one, if expected keep alive message is not received. It should be several times longer than keep alive interval.
    - **Number of bus switches** – Maximum number of bus switches. It must be odd number. That way, node will settle on secondary, if no master keep alive message is received.


Start bus selection process
Redundancy master keep alive message received?
YES
Reset number of bus switches
NO
NO
Bus switch timeout?
YES
YES
Maximum number of bus switches reached?
NO
Switch bus

CAN-TS: Keep Alive monitor logic



Implementation example

TRISAT Satellite

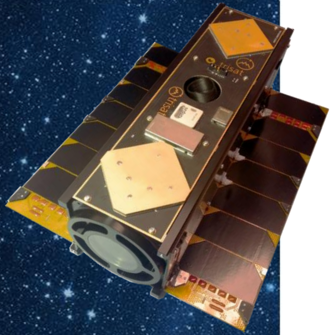
- 6 CAN nodes on-board: OBC, CPDU, TM/TC COMM, Payload Hi-Speed COMM, ADCS, MMSI Payload OBC
  - 13 physical addresses

1x CPDU:	local
2x COMM:	**GS** + local
4x OBC:	local + low level + scheduler + job table
2x AIM:	local + low level
2x PAYLOAD	local + low level
2x SBAND	**GS** + local

low level access ensured over PicoSkyFT SoC CAN controller (specific function) - no FW required.

  - and 4 broadcasts

CAN Address: 0 – Time Synchronisation (OBC time master)
CAN Address: 1 – Subsystem Keep-alive (heartbeat with HSK TM)
CAN Address: 2 – Subsystem Error unsolicited
CAN Address: 3 – Warning unsolicited



Implementation example

TRISAT Satellite (cont'd)

- Hot redundant CAN bus configuration (CAN-TS v1.2 specific)
- 3U satellite defined over 300 TM/TC channels
- On-board CAN bitrate: 125 kbps (up to 2000 CAN msgs per second)
- Rotating buffer of up to 20 keep-alive messages per subsystem
    - 1 keep alive message per 2 seconds per subsystem
- Time synchronization: 1 time sync message per second (ms resolution)
- Effective mass transfers data rates of more than 56kbps (max data bandwidth 62 kbps with CAN2.0B)
    - FW upgrades, logs, raw and payload data access

skylabs

CAN-TS vs CANopen

Memory consumption (picoRTU - PicoSkyFT-L processor)**:**

- CAN-TS (redundancy, retransmission supported)
    - Program memory size: 5 kB
    - Per instance block transfer data memory consumption:
        - set block: 538
        - get block: 533
- CANopen (CANfestival, modified to conform to ECSS-E-ST-50-15C)
    - Program memory size: 24 kB
    - Object dictionary : 10 kB (data memory)
    - SDO context: ~1 KB (depends on max. block transfer size, in our case 889 B)

Pros for each protocol:

- CAN-TS:
    - Appropriate for constrained devices
    - Stateless, except SB and GB
- CANopen:
    - Standardized, a lot of tools/implementations available
    - Interoperability
    - Larger block transfers possible (better bus utilization)

CAN-TS is open source

Reference implementation

- Published and officially announced today @ CANinSpace 2019!

- Server side CAN-TS implementation for memory constrained devices available @ https://github.com/skylabs-si/CANTS-MCU
  - GitHub example PicoSkyFT Evaluation Board (SKY-9213)
  - CAN to USB interface (CAN2USB V2)

- Released under BSD license:
  - Allows commercial use, distribution and modifications
  - No liability

- Stay tuned, Client side CAN-TS implementation will be soon available in cross platform C++ implementation.

GitHub URL

PicoSkyFT Evaluation Board (SKY-9213)

skylabs

Thank you

SkyLabs d.o.o.

Koroška cesta 53D

SI-2000 Maribor

info@skylabs.si

CAN-TS vs CANopen

support slides

	CAN TS	CANopen	Notes
Complexity	Simple	Complex	
Transport protocols	Time sync (TS) Unsolicited telemetry (UTM) Telemetry (TM) Telecommand (TC) Set block (SB) get block (GB)	Time Stamp Object (TIME) Process Data Object (PDO) Service Data Object (SDO)	Similar concepts: TIME and TS, PDO and UTM, SDO block transfer with SB and GB, SDO expedited transfer with TM and TC
Management protocols		Synchronization Object (SYNC) Network Management (NMT) Emergency Object (EMCY)	TC vs. NMT Un. TM vs EMCY/SYNC
Liveness monitoring	Application layer (UTM on destination 1)	Part of CANopen (guarding, heartbeats)	
CAN bus redundancy switching	Part of specification, application layer, similar to ECCS-E-ST-50-15C	Not in CiA 301, but in ECCS-E-ST-50-15C	
Standardization	None, open source available	CiA 301, ECSS-E-ST-50-15C for space applications	
Data location	Anywhere (accessed through callbacks)	Object dictionary	
Unsolicited telemetry handling	Application layer has to send it	Stack can be configured to send PDOs automatically	