Online workshop meeting minutes  
2023-03-08

1. Participants

<table>
<thead>
<tr>
<th>CiA members (present)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Kenneth Tindell</td>
</tr>
<tr>
<td>Dr. Pavel Pisa</td>
</tr>
<tr>
<td>Konrad Groschowski</td>
</tr>
<tr>
<td>Michal Mosdorf</td>
</tr>
<tr>
<td>Seweryn Ścibior</td>
</tr>
<tr>
<td>Brent G. Gardner</td>
</tr>
<tr>
<td>Zoltan Bardos</td>
</tr>
<tr>
<td>James Holley</td>
</tr>
<tr>
<td>Vinh Huynh</td>
</tr>
<tr>
<td>Eric Penny</td>
</tr>
<tr>
<td>Jochen Schnitger</td>
</tr>
<tr>
<td>Kyle Williams</td>
</tr>
<tr>
<td>Jemej Heloian</td>
</tr>
<tr>
<td>Richard Hubbard</td>
</tr>
<tr>
<td>Shannon Lippincott</td>
</tr>
<tr>
<td>Vrasank Shkla</td>
</tr>
<tr>
<td>Helmut Renner</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CiA members (apologized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael Hilzendegen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CiA guests (present)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jean Dalenq</td>
</tr>
<tr>
<td>Wojciech Stodulny</td>
</tr>
<tr>
<td>Alberto Valverde</td>
</tr>
<tr>
<td>Sandi Habinc</td>
</tr>
<tr>
<td>Fabio Malatesta</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Laura Seoane</td>
</tr>
<tr>
<td>Isabella Shi</td>
</tr>
</tbody>
</table>

**CiA secretary**

<table>
<thead>
<tr>
<th>Holger Zeltwanger</th>
<th>CAN in Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reiner Zitzmann</td>
<td>CAN in Automation</td>
</tr>
</tbody>
</table>

**Competition caution:**
Participants in this meeting do not enter into any discussion or conduct that may infringe applicable antitrust laws. This applies not only to discussions in formal meetings but also to informal discussions taking place in the sideline of formal meetings.

2. **Presentations**

Mr. Zeltwanger opened on behalf of CiA e. V. the workshop. He introduced briefly to the workshop. The following companies provided presentations (attached):

- Airbus Defense & Space
- CAN in Automation
- Czech Technical University Prague
- Frontgrade Gaisler
- N7 Space

3. **Discussion and further steps**

The participants demanded in the discussion a follow-up workshop providing information about the CANopen FD application layer and the CAN XL lower layers and the CAN XL ecosystem.

CiA is going to schedule such an event within the next month. Invitations and details will be provided to the workshop participants.

**Holger Zeltwanger**

2023-03-17

**Annexes:**

- Airbus Defense & Space
- CAN in Automation
- Czech Technical University Prague
- Frontgrade Gaisler
- N7 Space
**CiA workshop**

**CAN in (outer-space) satellites**

- Welcome and introduction
- Presentations by several parties
- Discussion about hardware requirements
- Discussion about higher-layer protocol requirements
- Discussion about profile specification demands
- Summary: Next steps, establishing of technical groups

*a* Airbus Defense & Space, Benchmark Space Systems, ESA, Frontgrade Gaisler, N7 Space, Technical university Prague, etc.
Holger Zeltwanger

◆ Electronic engineer (university of applied science)
◆ Technical journalist (editor)
◆ Initiator of the nonprofit CiA (CAN user) association
◆ Member of the CiA Board of Directors since 1992
◆ Convenor of ISO TC22 SC31 WG3 (in-vehicle networks)
◆ Convenor of ISO TC22 SC31 WG4 (network applications)
◆ Convenor of DIN NA052-00-31-53 AK (J1939)
◆ Expert in ISO groups (e.g. TC22 SC31 WG2, TC23 SC19 WG1/5, TC299 WG6)
◆ Expert in IEC groups (e.g. TC22 SC22G, TC47 SC47A)
◆ Expert in DIN groups (e.g. DIN 4630, DIN 14700/4)
◆ Member of SAE J1939 committee (and task forces)
◆ Member of SAE J2284 (CAN) task force
◆ Member of SAE J2602 (LIN) task force
Welcome and introduction

◆ Participants from
  - Airbus Defense & Space
  - Benchmark Space Systems
  - Canis Automotive labs
  - Czech Technical University Prague
  - ESD
  - ESA (European Space Agency)
  - Frontgrade Gaisler
  - N7 Space
  - NASA J. Glenn research center
  - National Institute for Aerospace Technology (Spain)
  - Novium Designs
  - ESD
  - Skylabs
  - Texas Instruments
  - Vector

◆ General note
  Presentation slides will be distributed as well as brief meeting minutes.
**CAN in outer-space applications**

- CAN in space conference series organized by ESA (European Space Agency) in cooperation with partners (e.g. Sitael and Cobham) reported about the status of applying CAN-based networks in satellites.

- Most of the launched CAN-based satellites used radiation-tolerant EIA 485-based transceiver circuitry. There were also first ISO 11898-2 compliant CAN high-speed (HS) transceiver applied.

- Some CAN-based networks embedded in satellites used CANopen as higher-layer protocol. ESA has developed an according implementation guideline (ECSS-E-ST-50-15).
# CAN technology generations

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classical CAN&lt;sup&gt;a&lt;/sup&gt;</th>
<th>CAN FD&lt;sup&gt;b&lt;/sup&gt;</th>
<th>CAN XL&lt;sup&gt;c&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data field length</td>
<td>0 byte to 8 byte</td>
<td>0 byte to 64 byte</td>
<td>1 byte to 2048 byte</td>
</tr>
<tr>
<td>Priority (CAN DLL) identifier</td>
<td>11 bit or 29 bit</td>
<td>11 bit or 29 bit</td>
<td>11 bit (priority ID field)</td>
</tr>
<tr>
<td>Addressing</td>
<td>(11 bit or 29 bit)</td>
<td>(11 bit or 29 bit)</td>
<td>32 bit (acceptance field)</td>
</tr>
<tr>
<td>Service data unit type</td>
<td>n.a.</td>
<td>n.a.</td>
<td>8 bit (SDT field)</td>
</tr>
<tr>
<td>Virtual CAN network ID</td>
<td>n.a.</td>
<td>n.a.</td>
<td>8 bit (VCID field)</td>
</tr>
<tr>
<td>Bit stuffing</td>
<td>dynamic</td>
<td>dynamic (fixed in CRC field)</td>
<td>dynamic in arbitration field fixed in data phase fields</td>
</tr>
<tr>
<td>CRC polynomials</td>
<td>15 bit</td>
<td>17 bit or 21 bit</td>
<td>PCRC: 13 bit; FCRC: 32 bit</td>
</tr>
<tr>
<td>Error signaling</td>
<td>on</td>
<td>on</td>
<td>Configurable (on/off)</td>
</tr>
<tr>
<td>FAST mode switch</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Configurable (on/off)</td>
</tr>
<tr>
<td>Bit-rate ratio (high/slow)</td>
<td>Not applicable</td>
<td>up to 16 times</td>
<td>up to 40 times</td>
</tr>
</tbody>
</table>

Supported frame formats:

<sup>a</sup> CBFF (classical base frame format), CEFF (classical extended frame format);  
<sup>b</sup> FBFF (FD base frame format), FEFF (FD extended frame format);  
<sup>c</sup> XLFF (XL frame format)
**CAN physical layer (PMA) options**

- **EIA 485-based PMA (physical medium attachment) transceiver**: Not standardized transceiver circuitry used historically in classical CAN network approaches;

- **Legacy CAN transceiver**: Featuring high-speed (HS) transmission with NRZ (non-return to zero) coding with bit rates from 40 kbit/s to 1 Mbit/s depending on the network length;

- **Improved CAN transceiver**: Featuring also HS transmission with NRZ coding; the data(phase) bit-rate is up to 2+ Mbit/s using not optimized network topologies and up to 5 Mbit/s using networks with two nodes (point-to-point network);

- **CAN SIC transceiver**: SIC (signal improvement capability) transmission with NRZ coding is possible up to 8 Mbit/s, even when using not optimized network topologies;

- **CAN SIC XL transceiver**: Optional SIC XL (dual-mode) transmission with PWM (pulse-width modulation) coding up to 10+ Mbit/s using hybrid topologies (in this case, arbitration bit-rate is limited to 0.667 Mbit/s). Linear bus topology with short not terminated stubs enable data(phase) bit rates up to 20 Mbit/s.

**NOTE** In CAN XL networks you can run all frame formats using NRZ coding. In this case, you need to configure the arbitration bit-rate, the CAN FD data(phase) bit-rate, and the CAN XL data(phase) bit-rate.
CAN hardware (MCU with on-chip CAN protocol controller and CAN transceiver) need to be radiation-tolerant.

- Do we need test method specifications for the total-ionizing dose (TID) and single-event effects (SEE)?
- Redundancy of CAN networks and CAN interfaces could be an issue regarding interoperability of devices. Is there a dedicated specification for satellites required?
- Which CAN protocol generations are suitable and are desired for future satellite applications?
- Which PMA technologies are demanded for future satellite applications?
There is an implementation recommendation/guideline for CANopen (ECSS-E-ST-50-15) available by ESA, which has been updated. There are some dedicated CANopen protocol stacks for satellite applications available.

- Is it also demanded to migrate to CANopen FD?

- Additional CiA profile specifications can improve the interoperability of devices. Is there a need for such profile specifications? Perhaps an implementation guideline for CiA 401 is sufficient.
Questions and answers
**Discussion: Hardware requirements**

- Radiation-tolerance requirements and test methods
- Bit-timing configuration recommendations
- Physical interface implementation guidelines
- Network redundancy specification
- CAN device interface redundancy specification
- Cable requirements and test procedures
- Connector pin-assignment recommendations
- etc.
Discussion: HLP requirements

- Classic CANopen implementation guideline for satellite applications
- CANopen FD implementation guideline for satellite applications
- Additional specifications and system design guidelines
- etc.
Discussion: Profile specification demand

- Dedicated profile specifications for satellite applications
- Other demands regarding process data, configuration parameter, and diagnostic information
CiA workshop: CANopen-connectable devices suitable for satellites
Overview of use of CAN bus at Airbus Defence and Space
Agenda

SatCom Payload Serial Bus
Platform Modules Communication Bus
Implementation
CAN bus Adoption in space applications
CPU based CAN nodes is an emerging trend
Rosalind Franklin (ExoMars) Rover
SatCom Payload Serial Bus
Context
SatCom Payloads are made with several RF units
Operability of these units is done thanks to data bus
Typical RF units are associated to a batch of data (RF Gain, temperature, voltage, current…)
These parameters are cyclically monitored (typically every second)
RF units configuration can be changed through telecommand

During decades Airbus used proprietary serial bus : LSSB
With drawbacks : limited data rate and number of nodes, non optimized harness (5 pairs per bus), proprietary definition leading to specific development effort on suppliers side.

CAN has replaced LSSB :
Improved data rate (250kb/sec),
# node up to 64,
single pair per bus,
relying on established standard.
Payload Serial Bus

**Higher Level Protocol**

In the frame of Neosat and Artes 5.2 Protocol has been agreed with TAS, TESAT and Airbus DS

Master/Slave + Time Triggered

Message coding is CANopen

---

<table>
<thead>
<tr>
<th>Message type</th>
<th>Direction</th>
<th>PDO</th>
<th>Data</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM Request</td>
<td>Master -&gt; Slave</td>
<td>RPDO1, 2, 3, 4</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Data Transmission</td>
<td>Slave -&gt; Master</td>
<td>TPDO1, 2, 3, 4</td>
<td>Yes</td>
<td>Shall always be initiated by TMReq</td>
</tr>
<tr>
<td>Unconfirmed Command</td>
<td>Master -&gt; Slave</td>
<td>RPDO1, 2, 3, 4</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>
Payload Serial Bus

**Bus Management**

Slave implementation:
Slave Nodes are very often CPU-less units
Cost Driven $\rightarrow$ simplest implementation, simplest validation
- No CPU added to manage CAN bus
- Single CAN bus Controller with bus selection allowed in order to minimize logic resources

Master implementation:
Master CAN bus manager is also hardware implemented
Telemetry polling via frames:
- Cyclic
- Generic $\rightarrow$ Shrink Validation Time
- Deterministic

Slave Nodes send messages only on Master Request
Payload Serial Bus

**Eurostar Neo implementation**

CAN bus definition:
- 64 nodes
- 40 meters
- 250 kb/sec

- P/L buses managed directly by OBC
- 4 buses available
- Complete P/L data is poll in every 1 sec

**CAN bus definition:**
- 64 nodes
- 40 meters
- 250 kb/sec
- P/L buses managed directly by OBC
- 4 buses available
- Complete P/L data is poll in every 1 sec
Platform Modules Communication Bus

Aim and characteristics

- Eurostar Neo Telecom platform now in production
- DHS units use CAN bus to connect I/O modules to the processor

- Data rate: 1Mb/sec
- Most distant nodes: <= 15 meters
- Nodes: <= 44

- CAN bus is more and more used as “backplane” communication bus
Protocols
Protocols

Thalès Alenia Space and ADS has defined a common Applicable Document (AD) in the frame of Neosat Program.

This AD is known by many suppliers, used on several platforms: Spacebus (TAS), Eurostar Quantum, Onesat, G2SB1 (ADS)

Message format are CAN open, but no OD or other CAN open resources are used.

Node Implementation is often hardware with no configurability.

Master Slave protocol
No spontaneous message from the slaves
Master manages the bus in a time triggered approach

This protocol is also defined in ECSS ECSS-E-ST-50-15C CAN section 9 Minimal implementation of the CANopen protocol for highly asymmetrical control applications

Recently ESA driven ADHA project revisited the Neosat AD adding:
Buffered Telemetry Request
Patch and Dump Operations
Time Distribution and Synchronization
Implementation
CAN bus Transceivers

- Rad Hard CAN bus transceivers are available

- Intersil
  - ISL71026M, 14 Ld TSSOP package
  - ISL72026, ISL72027, ISL72028 SMD 5962-15228, FP8 Hermetic package

- Frontegrade (Cobham)
  - UT64CAN333x SMD 5962-15232, FP8 hermetic Package

- Texas Instrument
  - SN55HVD233-SP SMD 5962L1420901VXC, FP8 hermetic Package

Transceivers Variants:
- Loop back mode: allows to isolate driver and receiver from the bus for diagnosis
- Vref output: provides a voltage reference for split matching
- Low power shut down: allows to reduce TCVR consumption when not used
- Auto-baud loop back: allows to isolate the driver from the bus but not the receiver
CAN bus Controllers

Most of Satcom nodes are implemented with FPGA or ASICS
- COTS IP Core are used: E.g. INICORE, CAST
- Logic Synthesis must consider SEU mitigation. Note that FSM of COTS IP Core are not necessarily hardened against SEU.

µcontroller and µprocessor are also used:
- Fronte grade: GR712, UT32M0R500
- TI: TMS570LS1227
CAN bus Adoption in space applications
Several units implement CAN bus interface:
- Up/down flexible converters
- SSPA
- Channels amplifiers
- Stable Oscillators
- Telemetry transmitters
- Telecommand receivers
- Centralised Power Supply Unit (CPSU)
- Radiation monitor, Hosted Payload Computer

From Several suppliers:
- Tesat, Thales Alenia Space
- Kongsberg, Beyond Gravity
- NEC Space Technologies, Ltd.
- Airbus DS, Sener, Stellant
- ....
- CAN bus is already flying on Eurostar 3000, Eurostar Neo, Quantum, and soon on Onsesat and Galileo
CPU based CAN nodes is an emerging trend
CPU based CAN nodes is an emerging trend

Two different CAN nodes implementing GR712 Microprocessor are used in Satcom hosted payloads

- Instrument interface and processing electronic will fly on E3000 platform
- ICARE Radiation monitor flies on Eurostar Neo platform

- Avionics and payloads units using TMS570LS1227 or UT32M0R500 are under development
Rosalind Franklin (ExoMars) Rover
Rosalind Franklin (ExoMars) Rover

CAN Bus implementation:
- 1 P/F + 1 P/L bus
- 1000 kbps
- RS 485 (DS16F95)
- Higher level protocol: CANopen ECSS ECSS-E-ST-50-15C (HurriCANe/CCIPC)

Platform:
- Lengths: 7m between most distant nodes, 8.5m total.
- 10 nodes per bus

Payload:
- Lengths: ~3.8 m between most distant nodes.
- 8 nodes per bus

Status:
- “ESA’s ExoMars rover is confirmed technically ready for launch, and a fast-track study is under way to determine options for bringing the mission to Mars.”
Thank you for your attention
CTU CAN FD
Origin at
Czech Technical University in Prague

Open source CAN FD IP core
https://canbus.pages.fel.cvut.cz/

2023-03-08
Ondrej Ille <ondrej.ille@gmail.com>
Pavel Pisa <pisa@fel.cvut.cz>
CTU CAN FD IP Introduction

- CTU CAN FD IP ecosystem
  - ISO11898-1 2015 compliant IP core in VHDL 93
  - ISO18645-1 2016 compliance test-suite
  - VHDL design with no vendor-specific libraries required, yet RAM for buffers and Rx FIFO automatically inferred by Xilinx and Intel tools
  - Extensive semi-randomized test-suite in simulation with PSL functional coverage
  - SocketCAN Linux driver – Part of mainline Linux kernel
  - QEMU user-space model – Part of QEMU mainline
Features

- RX buffer FIFO with 32 - 4096 words (1-204 CAN FD frames with 64 bytes of data)
- 2-8 TXT buffers (1 CAN FD frame in each TXT buffer)
- 32-bit slave memory interface (APB, AHB, RAM-like interface)
- Support of ISO and non-ISO CAN FD protocol
- Timestamping and Time triggered transmission
- Interrupts
- Loopback mode, Bus monitoring mode, ACK forbidden mode, Self-test mode, Restricted operation mode
TX Buffer States and Transitions

- **TX_PRIORITY**, up to 8 nibble-aligned fields
- Write allowed at any time
- Allows to maintain FIFO, priority queue or multiple queues

Legend:
- Transmission result
- SW command
- Accessible for SW
- Inaccessible for SW
Current usage

- Many FPGA based applications
  - Custom automotive testers (HIL units)
  - Embedded devices
  - Aerospace measurement units (fault tolerance extensions, parity bits in design and logic SEU protection by synthesis)
  - Industrial machinery
- Current integration
  - Xilinx Zynq 7000 – MZ_APO board
  - Intel EP4CGX15 based DB4CGX15 PCIe board
  - Cyclone V 5CSEMA4U23C6 based DE0-Nano-SoC Terasic board
CAN FD Latency Tester Application

Gateway DUT

CAN bus A

can0

Can1

can2

Can3

CAN bus B

Ethernet

Console + reset

Eth

Eth1
USB
PC

To LAN

MicroZed APO Board - LaTester
History

- 2015: IP written at Czech Technical University
- 2018: Linux driver written
- 2018: IP RTL open-sourced under MIT
- 2019: Complete re-write of the RTL (to target ASIC)
- 2020: ISO16845 test suite and C++ model started (VIP)
- 2020: License changed on HW and TB
- 2021: Compliance with ISO11898 reached
- 2022: Linux driver accepted in mainline kernel
Current developers

- **Ondrej Ille** (self-funded)
  - RTL, TB and VIP (compliance suite)
- **Pavel Pisa** (self-funded, partially CTU)
  - Linux driver maintainer
- **Martin Vasilevski** (CTU, student)
  - Linux driver extensions for timestamping
- **Jiří Novák** (CTU)
  - Windows driver and tester for SkodaAuto
Licensing status

- HW, TB and VIP copyright held by Ondrej Ille
- Linux driver licensed under GPLv2
- Currently searching for other sustainable model
Future development

- RTL, HW and TB
  - Split design to two clock domains (2023)
  - Flip to simulator with code-coverage (2023)
  - VHPI support in VIP (2023)
  - ISO11898-1 certification (following years)
  - Deployment in mass ASIC production (following years)
  - CAN XL support (long term goal in future)
  - TT CAN support (long term goal in future)
Future development cont.

- Linux driver
  - Maintenance
  - Multi-cast support
  - QEMU cosimulation with RTL simulation
- QEMU
  - Maintenance of user model
- NuttX CTU CAN FD driver
  - We have already implemented driver for ESP32C3, imxRT, updated to CAN FD SAMV7x
Future development cont.

- RTEMS CAN drivers
  - CTU CAN FD driver (Pavel Pisa mentor in multiple RTEMS GSoC, the last one prototype of unified CAN driver)

- Test on nanoXplore
  - NX1H35AS-EK available at PiKRON s.r.o. from previous European Space Agency De-Risk project
  - Extension board for motion control etc. available
    https://gitlab.com/pikron/projects/sumintadc/deliverables
Compliance Testing Funding

• Problems of current development
  – Need to collect funding for work and payment of official compliance testing on RTL elevel and then final products

• Option to form foundation
  – Purpose: Fund, develop and certify the IP core.
  – Members would have:
    • Right to use latest HW / TB / VIP for commercial purposes.
    • Access to technical discussion in the project development (possibly technical steering comitee in future)
    • Support upon agreement
    • Prioritized fixes of possible RTL / driver issues.
Thank you

Overview article:
https://can-newsletter.org/uploads/media/raw/a9abe317ae034be55d99fee4410ad70e.pdf
Project pages: https://canbus.pages.fel.cvut.cz/
Our portfolio of CAN based products

CAN in satellites
CiA workshop

Fabio Malatesta
Product Marketing Engineer
Overview

• Introduction
• GRLIB VHDL IP library
• CAN 2.0 -> GRCAN
• CANFD -> GRCANFD
• Software support
• CAN-FD transceivers
• Conclusions
Introduction

CAN has increasingly been used in space applications to replace conventional spacecraft bus architectures

• Reduce the amount of wires and connectors required
• Flexibility
• Low power consumption
• Arbitration and error detection
GRLIB - VHDL IP LIBRARY

- A complete SoC design environment:
  - IP Cores
  - AMBA on-chip bus with plug & play
  - Scripts to support implementation tools
  - Template designs for prototyping boards
IP core building blocks

- Synthesizable processor cores and system peripherals
- Described in VHDL code
- Available in GRLIB VHDL library
- Open GRLIB community
- Excel sheet for SoC area estimation

**NOEL Processor Family**
- NOEL-V

**LEON Processor Family**
- LEON5
- LEON4
- LEON3

**Peripherals**
- Memory controllers
- On-chip interconnect
- Communication interfaces
- Encryption and compression
- Error detection and correction
- Spacecraft data handling functions
- Verification
- Auxiliary functions
- Test functions

All peripherals can be used with NOEL and LEON families.
GRCAN

• Compatible with CAN 2.0 (base and extended formats)
• DMA engine to autonomously fetch and store frames through an AMBA AHB master interface
• AMBA APB slave interface for configuration registers
• Silicon implementations: GR740 and GR716A

GRCAN is a legacy IP. For new designs, the GRCANFD core is recommended.
GR716A LEON3FT Microcontroller

Baseline features

• Single-core LEON3 SPARC V8 processor, Rad-hard, Fault Tolerant
• 16-bit instruction set: LEON-REX for improved code density
• Support for many different standard interfaces (SPI, I2C, SpaceWire, 1553, CAN and more)
• On chip rad-hard DAC, ADC, LDO and PLL
• LEON Technology – re-use of Development and Software ecosystem
• Flight models available

• For more information -> GR716 webpage
**Value proposition**

- Highest performance, wide range of interfaces
- Quad-Core LEON4: SPARC V8, Rad-hard and Fault Tolerant
- Designed as ESA’s Next Generation Microprocessor, NGMP
- LEON Technology – re-use of Development and Software ecosystem
- SEU errors corrected without software interruption
- QML Q/V qualified
- Excellent performance/watt ratio
  - Very low power, < 3 W (core typical)
  - Performance 1800 DMIPS (1000 MIPS)

**Interfaces**

- 8-port SpaceWire router
- Redundant CAN
- 2x Ethernet
- Parallel PCI
- MIL-STD-1553

For more information -> [GR740 webpage](#)
CANFD introduced in 2012 to meet the requirements of higher bandwidth in modern applications

- Maximum payload of 64 bytes
- Optional bit-rate switch (typ. up to 8 Mbit/s)
- CAN and CAN-FD nodes can still coexist in the same network

New GRLIB IP to offer the benefits of the FD extension to the space industry: GRCANFD
GRCANFD

- Fully compatible with ISO 11898-1:2015
- DMA engine to autonomously fetch and store frames through a generic bus master interface
  - Wrappers available for both AMBA AHB 2.0 and AXI4
- AMBA APB slave interface for configuration registers
- FIFOs protected by fault-tolerant features
- Upcoming silicon implementations: GR716B and GR765
• The CAN-FD controller implements the functionality related to MAC and PL (PCS) sub-layers of the protocol

• Main functionalities:
  • Transmission, reception and acknowledgment of frames
  • Arbitration control
  • CRC calculation and verification
  • Error detection and signaling
  • Bit segments generation and synchronization
  • Transmitter delay compensation
GRCANFD - DMA

- Configuration registers and interrupts generation
- DMA engine with separate TX and RX channels, each with a dedicated FIFO of configurable size
- The TX channel fetches the frames from an external buffer and stores them into the FIFO. New frames are fetched as soon as the codec transmits a complete frame
- The RX channel filters and stores the frames received by the codec into the FIFO. The frames are then written to external buffer and new frames can be locally stored
- External buffers located in on-chip or off-chip memories
GRCANFD – More features

- CAN bus redundancy
- TX and RX channels controlled independently
- Frame acceptance filters
- TX and RX SYNC filters (interrupt generation)
- Single-shot mode
- Listen-only mode
- Internal or external loopback
- Transmitter delay compensation of up to 2 data bit-times
- Overload frame generation when RX FIFO is full
- CANOpen
GRCANFD – CANOpen

- CANOpen Minimal Set Protocol as per the ECSS-E-ST-50-15C specification, section 9
- GRCANFD can only be a slave node
- PDO commands
  - Provide R/W access to on chip-bus
  - Enables processor boot via CAN
- Heartbeat commands
  - If nominal bus is dead, switch automatically to redundant
- SYNC commands
- (Optional) separate DMA interface for CANOpen
**Some updated features compared to GR716A**

- Max clock frequency increased to 100 MHz
- 2-port SpaceWire router instead of two SpaceWire controllers
- CAN-FD with CANOpen controller instead of CAN controllers
- FPGA supervisor
- Real time accelerators to offload the main LEON3 for simple tasks
- Ethernet controller
- Upgraded analog interfaces to better support DC/DC controller and motor control applications

• For more information -> GR716 webpage
Baseline Features

- Fault-tolerant octa-core architecture
  - LEON5FT SPARC V8 or NOEL-V RV64GCH
  - Dedicated FPU and MMU, 64 KiB per core L1 cache, connected via multi-port interconnect
- Target technology: STM 28nm FDSOI
- 1 GHz processor frequency - 26k DMIPS
- 2+ MiB L2 cache, 512-bit cache line, 4-ways
- DMA controllers
- DDR3 interface with dual x8 device correction capability
- 8/16-bit PROM/IO interface
- (Q)SPI and NAND memory controller interfaces
- Secure Element, providing Secure (authenticated) boot (TBD)
- eFPGA ~30k LUT (TBD)
- High-pin count – LGA1752 package – allows using more interfaces simultaneously

In development
No guarantee of product launch
Software support
**CAN software drivers and simulators**

**Application software:**
Drivers and examples for
- Linux
- RTEMS
- VxWorks
- bare-metal

**Simulation:**
- TSIM3
CAN Transceivers
UT64CAN333X Transceiver series

- Compatible with ISO 11898-2 and ISO 11898-5 standards
- Single +3.3 V supply voltage
- +5 V tolerant digital I/O
- 10 kbps to 8 Mbps bit-rates
- QML-Q and QML-V qualification
- Total Ionizing Dose: 100 krad (Si)
- SEL Immune: ≤ 141 MeV-cm²/mg
- Evaluation board
Conclusions
Conclusions

• We provide a broad offering of CAN products:
  • IP Cores
  • Software drivers
  • Microcontrollers & Microprocessors
  • CAN transceivers
• For further information visit www.gaisler.com
• Contact me at fabio.malatesta@gaisler.com
N7 SPACE

CANopen Software Library for ECSS-E-ST-50-15C compliant
On-Board Implementation

Konrad Grochowski

CiA workshop "CAN in satellites"

© 2023 N7 Space sp. z o.o.
N7 Space

• Warsaw based company devoted to space systems software development

• Key activities:
  • Critical software development for real-time embedded systems
    • LEON and ARM based systems
    • Software qualification based on ECSS standards
  • Development and applications of MBSE tools
    • Formal architecture and data modelling
    • Rich experience with TASTE and Capella
  • Contribution to EGS-CC (EU Ground Systems Common Core) ecosystem development and integration

© 2023 N7 Space sp. z o.o.
Project overview

• In 2021 N7S finished ESA project: ECSS-E-ST-50-15C Protocol On-Board SW Implementation

• Goal: implement and validate reusable space-grade software library compatible with ECSS CANopen extension standard
  • ECSS Criticality Category B
  • Validated on representative hardware

• There is an abundance of mature CANopen tools
  • But no ECSS-E-ST-50-15C compliant version is easily available
  • ECSS introduces specific features:
    • redundancy
    • new format for time distribution
    • software quality requirement
How?

• Used existing and field tested open-source library
• Changes were contributed to the open-source project so they become „core”

• Used lely-core (https://opensource.lely.com/canopen/)
**Why lely-core?**

- Among few libraries suggested by ESA after Agency’s research (2020)

<table>
<thead>
<tr>
<th>Feature</th>
<th>CANopenNode</th>
<th>openCANopen</th>
<th>lely-core</th>
<th>CanFestival</th>
</tr>
</thead>
<tbody>
<tr>
<td>CiA301 compliance</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>All basic data types</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>All extended data types</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PDO synchronous</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PDO asynchronous</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PDO write</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PDO read</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SDO services</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SDO server</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SDO client</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SYNC</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TIME</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>EMCY</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NMT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NMT state machine</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Object dictionary</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Additional details**

<table>
<thead>
<tr>
<th>License used</th>
<th>GPL2</th>
<th>ISC</th>
<th>Apache 2.0</th>
<th>LGPL2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCF/EDS support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Does not require OS</td>
<td>✓</td>
<td>X</td>
<td>✓/X</td>
<td>✓/X</td>
</tr>
<tr>
<td>In active development</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Provides own tests</td>
<td>✓</td>
<td>✓/X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Provides auxiliary tools</td>
<td>None</td>
<td>Interface enumerator</td>
<td>DCF to C tool, sniffers</td>
<td>OD generator</td>
</tr>
</tbody>
</table>

© 2023 N7 Space sp. z o.o.
Why *lely-core*?

- Good code and process quality
  - Tests
  - Coding standard based on *Linux kernel coding style* and checked by SonarCloud + Coverity
  - Platform independent code (simple HAL)
  - Well documented API
- Used and maintained by Lely Industries – a robotic company, member of the CiA
  - More than 20,000 robots since 2018
  - Lely supported the activity
- Active community
- Implements: CiA 301, 302, 305, 306, 309 and 315
Pre-qualification results

- Added ECSS specific features to the *lely-core*
  - Redundancy
  - Space specific time representation

- Made *lely-core* Criticality Category B complaint
  - Removed dynamic allocation and external dependencies
  - Unit-test 100% coverage (line/branch/decision)

- Validated the library on the selected representative hardware
  - ATSAMV71Q21 with Xplained Ultra Evaluation Kit
    - SAMV71Q21RT existing radiation tolerant version of the MCU
  - Validation test suite executed on a dedicated Hardware Test Bench (HWTB) prepared by BD SENSORS
Pre-qualification approach

• Requirements and test scenarios based directly on ECSS and CiA standards
• Unit-tested function-by-function
• Scripts and environment configuration are part of the public repositories
• Many tasks executed on publicly available Continuous Integration systems
  • Unit testing
  • Static analysis
• Everything should be a „single click” process
Pre-qualification approach

- 61 GitLab CI jobs, including
  - Unit testing
  - Valgrind
  - cppcheck
  - Clang-tidy
  - Coverity
  - SonarCloud
- 2.5k unit-tests, executed in various library configurations (40k tests executions)
- Dedicated Jenkins pipeline with automatic execution of all validation tests on the HW
Project deliverables

• CANSW – CANopen SW Library
  • https://gitlab.com/lely_industries/lely-core
  • https://gitlab.com/n7space/canopen/lely-core

• CTESW – CANSW Test Environment
  • https://gitlab.com/n7space/canopen/test-environment

• CTSSW – CANSW Test Suite
  • https://gitlab.com/n7space/canopen/test-suite

• ECSS compliant documentation datapacks tailored for criticality B qualification
  • Distributed by N7 Space
Example – complete NMT service setup

```c
// necessary #include directives
#include "dcf_nmt.h" // from dcf2dev

co_nmt_t* setup_nmt(co_dev_t* device, can_net_t* network) {
    co_nmt_t* nmt = co_nmt_create(network, device);
    assert(nmt != NULL);

    co_nmt_set_hb_ind(nmt, &hb_ind, NULL);
    co_nmt_set_st_ind(nmt, &st_ind, NULL);

    co_nmt_cs_ind(nmt, CO_NMT_CS_RESET_NODE);
}

int main() {
    co_dev_t* device = dcf_nmt_init(); // from dcf2dev
    alloc_t* alloc = create_allocator();
    can_net_t* network = can_net_create(alloc);
    can_net_set_send_func(network, &send_func, NULL);

    co_nmt_t* nmt = setup_nmt(device, network);

    set_current_time(network);

    while (!end_condition_met()) {
        receive_message(network);
        set_current_time(network);
    }

    co_nmt_destroy(nmt);
    can_net_destroy(network);
    return 0;
}

void hb_ind(co_nmt_t* nmt, co_unsigned8_t id, int state, int reason,
            void* data) {
    printf("HB: id=<%d> state=<%d> reason=<%d>
", id, state, reason);
}

void st_ind(co_nmt_t* nmt, co_unsigned8_t id, co_unsigned8_t st,
            void* data) {
    printf("ST: id=<%d> st=<%d>
", id, st);
}
```

Heartbeat and status indication functions (callbacks) added as an example.

Few lines for setting up the most complex CANopen service – NMT.

Simple application example of a working CANopen master. (It reboots all the nodes in the network and monitor their status).
DCF – Device Configuration File

• CiA 306
• Human readable format

```
[DeviceInfo]
VendorName=
VendorNumber=0
ProductName=
ProductNumber=0
RevisionNumber=0
OrderCode=
BaudRate_10=0
BaudRate_20=0
BaudRate_50=0
BaudRate_125=0
BaudRate_250=0
BaudRate_500=0
BaudRate_800=0
BaudRate_1000=0

[MandatoryObjects]
SupportedObjects=3
1=0x1000
2=0x1001
3=0x1018

[OptionalObjects]
SupportedObjects=0

[ManufacturerObjects]
SupportedObjects=0

[1000]
ParameterName=Device type
DataType=0x0007
AccessType=ro

[1001]
ParameterName=Error register
DataType=0x0005
AccessType=ro

[1018]
SubNumber=5
ParameterName=Identity object
ObjectType=0x09

[1018sub0]
ParameterName=Highest sub-index supported
DataType=0x0005
AccessType=const
DefaultValue=0x4

[1018sub1]
ParameterName=Vendor-ID
DataType=0x0007
AccessType=ro

[1006]
ParameterName=Communication cycle period
DataType=0x0007
AccessType=rw
DefaultValue=500000

[1016]
SubNumber=2
ParameterName=Consumer heartbeat time
ObjectType=0x09

[1280]
SubNumber=4
ParameterName=SDO client parameter
ObjectType=0x09

[1280sub0]
ParameterName=Highest sub-index supported
DataType=0x0005
AccessType=const
DefaultValue=0x03
```
The tool is used by every test in the Test Suite.
Summary

• Matured industrial open-source library (lely-core) pre-qualified
• Unit-tests 100% coverage on lines, branches and decisions
• HWTB based on SAMV71 (ARM) board with dual CAN bus support
• Test Suite containing set of CAT-B validation tests for the software library
• Complete CAT-B compliant documentation and software data pack was created
  • Including Qualification Guideline
Future

• Planned future activities include enhancing Test Suite for validating software on other ARM and LEON based MCUs

• Commercial library deployment is already in progress

• N7 Space can provide support with
  • Library adaptation to new MCUs
  • Development of dedicated HWTB allowing to run the test suite
  • Support to delta qualification and execution of the validation test suite in the target environment
CANopen On-Board Protocol Stack

ECSS criticality B pre-qualified CANopen library for use in space environment.

Main Features

- Complete CANopen software stack implementation, compliant with ESA 301, ESA 305 and ECSS-E-ST-50-10C standards.
- Highly customizable and configurable open-source library targeted for critical embedded systems.

- Validation test suite and documentation which includes:
  - ECSS-Q-ST-80C standards;
  - ECSS-E-ST-50-10C validation tests;
  - 100% line, branch and M3D3C coverage by unit-tests.

- C language API of the library provides pure C99/CLI11 interfaces, object-oriented design, static memory allocation, no external libraries dependencies. C++ wrappers (for non-critical software).

- Complete implementation

- Object dictionary code generator

Konrad Grochowski
kgrochowski@n7space.com
+48 22 299 20 50
https://n7space.com

© 2023 N7 Space sp. z o.o.