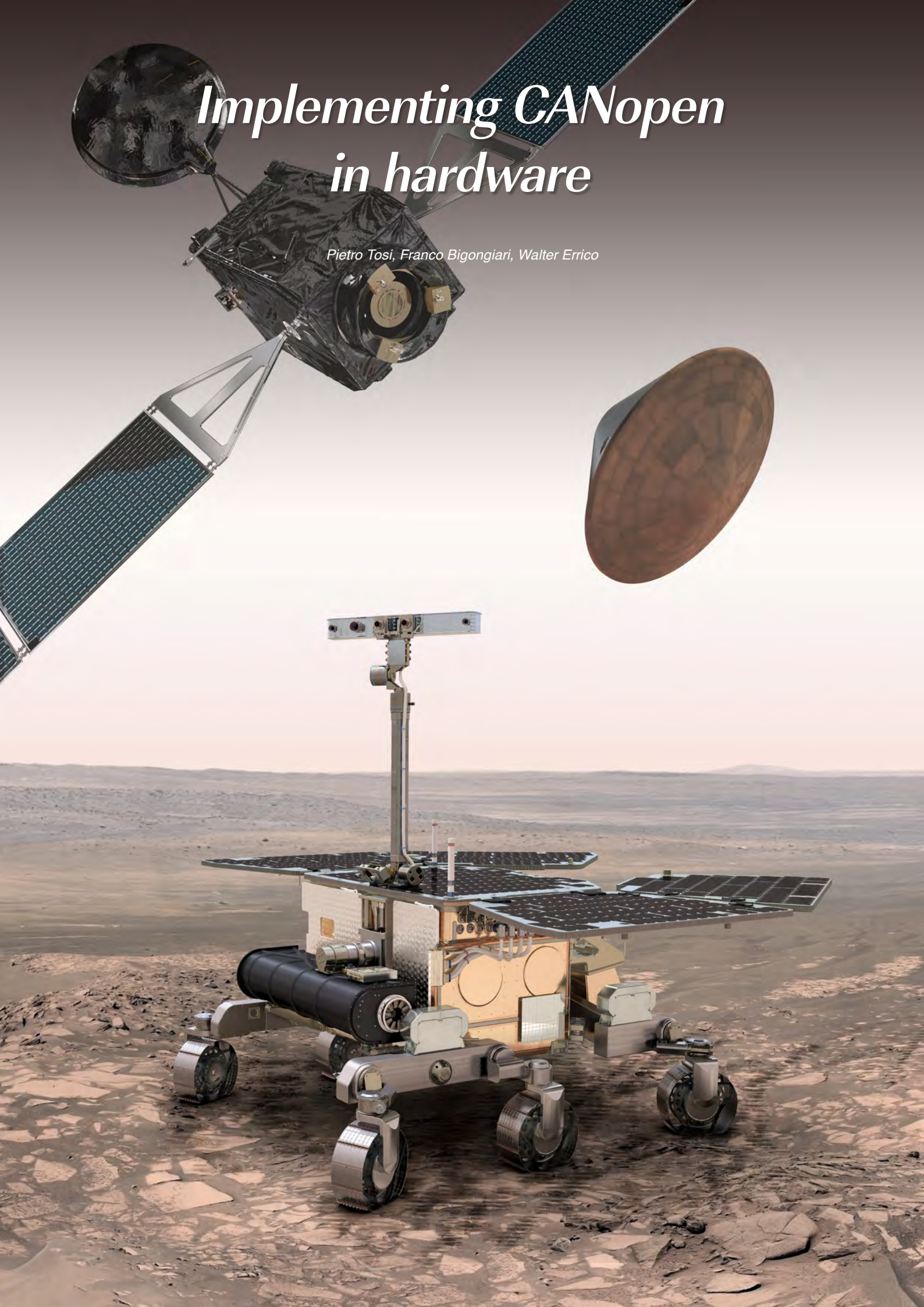
Implementing CANopen in hardware

Pietro Tosi, Franco Bigongiari, Walter Errico



The CCIPC is a hardware solution for CANopen implementation in charge of managing CAN physical, data link and application layers simultaneously. It is capable of managing the main CANopen protocol aspects (NMT, PDOs, SDO, etc.) with additional features as bus redundancy and error correction and detection (EDAC) algorithms for high-reliable communication. The CCIPC has been designed with a lot of configurable and customizable parameters and it can be implemented both in ASIC and FPGA technologies.

Architecture of the CANopen core

Within a CAN network, the CCIPC takes the role of a CANopen slave node. CCIPC implements the following CANopen services:

- PDO handler (synchronous and asynchronous RPDO and TPDO)
- SDO handler (expedited, segmented and block)
- Network Management
- Synch consumer
- Heartbeat consumer and producer

The CCIPC also supports a specific service, called Redundancy Manager, in charge of executing redundancy algorithm and managing CAN network multiplexing.

Figure 1 presents the CANopen Controller IP core architecture. The CAN controller module is the manager of the CAN interface and it's able to transmit, receive and filter CAN messages.

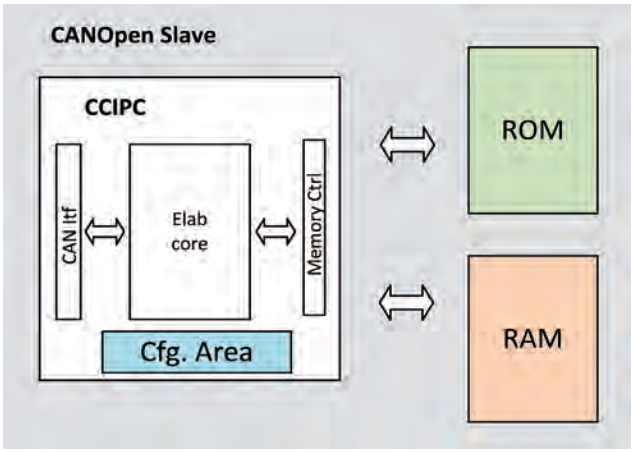


Figure 2: CANopen slave node

The Bus manager module is part of the Network Manager block; it is represented as a separate block in order to explicitly show the CCIPC capability to implement the Bus Redundancy management protocol. The two CAN queue blocks (MsgIn and MsgOut Queue) are used to temporaryly store messages received and messages to be transmitted, using a FIFO mechanism. The Network Manager is the module that controls and coordinates all system functionalities. It performs the Network Management and the Heartbeat Error control services. The event handler module is in charge of enabling the different CANopen functionalities (SDO, RPDO and TPDO) according to the state of its input trigger signals. The OD (object dictionary) handler is the module that has direct access to the CANopen Object Dictionary area in order to implement the different CANopen communication services (PDO and SDO protocols).

The Object Dictionary area has been split in two main subsets:

- Internal area (communication parameter), which stores the PDO communication parameters area;
- External area (mapping parameter & application objects), which includes the PDO mapping parameters area and the manufacturer-specific profile area (2000_h to $5FFF_h$).

The Index pointer RAM internal memory block contains the tables of pointer, keeping the cross references between the ▷

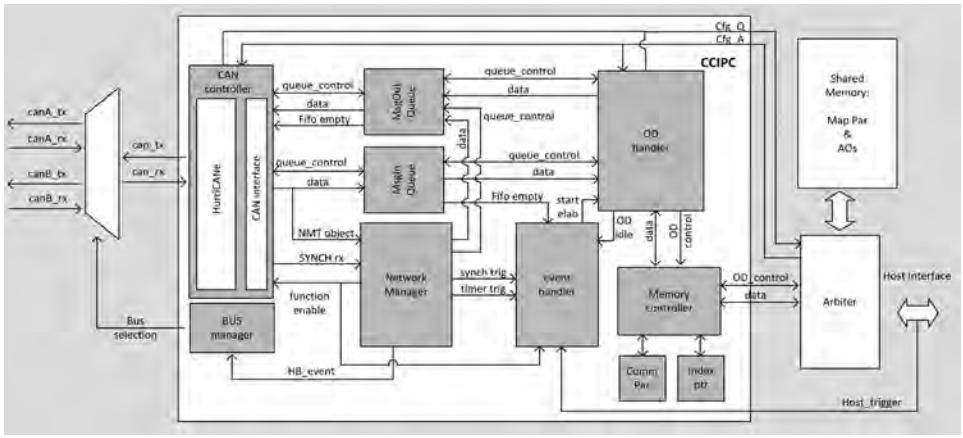


Figure 1: Architecture of the CCIPC

Authors

Pietro Tosi

Franco Bigongiari

Walter Errico

Sitael SpA
S. P. 231 km 1+300
IT-70026 Modugno (BA)
Tel.: +39-050-9912116
Fax: +39-050-9910249
info@sitael.com

Link
www.sitael.com

Introduction

It has been a long time since the CAN network was adopted as one of the possible spacecraft onboard networks for control and remote sensing data communication. To promote the use of CAN and to be more similar to structured data networks like MIL-STD-1553B, in the framework of ESA ExoMars program (launches scheduled for January 2016 and May 2018) and under a contract with Turin premises of Thales Alenia Space Italia has developed a CANopen Controller IP Core (CCIPC).

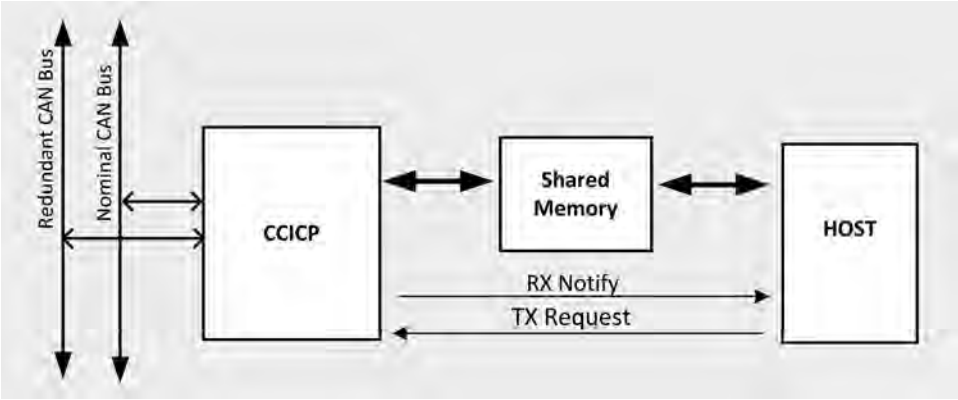


Figure 3: CCIPC host interface

Figure 4: CCIPC graphical user interface

OD application objects and their physical addresses in the external shared-memory. The memory controller is the unit that handles the access on the core's internal and external memory areas. An auxiliary arbiter module allows sharing of the external memory between CCIPC and the host device.

Host interfaces

The CCIPC is designed to cover all aspects of the protocol, exposing the CANopen object dictionary to the user as a shared-memory area with additional received messages notification (interrupt) signals and transmission request triggers.

In order to work as a CANopen slave node in accordance with the Network Management (NMT) "reset node" and "reset communication" services, the CCIPC needs to access two external memory areas, hereafter simply called ROM and RAM.

The ROM area is assigned to store the default values for TPDO/RPDO communication/mapping parameters and for application objects that have to be loaded after power-on or when the "reset node" or "reset communication" requests are received. Depending on the specific instance and in case of a FPGA implementation, this area can be realized with different solutions:

- Internal implementation as a LUT circuit (for example in case of 1-2 node with application objects limited to a few hundred bytes),
- Internal RAM macro-cell. A dedicated internal FPGA memory has to be preloaded before releasing the CCIPC Rst_n signal (e.g. the case of an external microprocessor or a serial link connected to a remote server/memory devices),
- External EEPROM memory. This is a simple (and costly) solution achievable with an external non-volatile memory,

The RAM area is assigned to keep TPDO and RPDO Mapping Parameters and Application object data. This area is shared between the CCIPC core and the External Device. Depending on the number of Node supported and Application Object defined, this area can be implemented exploiting the internal macro-cells (in case of implementation on FPGA) or as part of an external RAM device.

The CCIPC also includes an additional memory area, called configuration area that collects all CCIPC configuration and status registers.

Three different interfaces are available to realize the memory sharing between the CCIPC and its Host Device depending on the different CAN network user categories (CPU-less, micro-controller and CPU) and also different OD structure size:

- The generic I/O interface is the more general solution, where an external arbiter module is in charge of managing the sharing of the memory device between the CCIPC and host device according to request/lock signals coming from the two interfaces;

EB(0) := D(0) ^ D(1) ^ D(2) ^ D(4) ^ D(5) ^ D(7) ^ D(10) ^ D(11) ^ D(13) ^ D(16) ^ D(20) ^ D(21) ^D(23) ^ D(26) ^ D(30) ;

EB (1) := D(0) ^ D(1) ^ D(3) ^ D(4) ^ D(6) ^ D(8) ^ D(10) ^ D(12) ^ D(14) ^ D(17) ^ D(20) ^ D(22) ^ D(24) ^ D(27) ^ D(31) ;

EB (2) := D(0) ^ D(2) ^ D(3) ^ D(5) ^ D(6) ^ D(9) ^ D(11) ^ D(12) ^ D(15) ^ D(18) ^ D(21) ^ D(22) ^ D(25) ^ D(28) ;

EB (3) := D(1) ^ D(2) ^ D(3) ^ D(7) ^ D(8) ^ D(9) ^ D(13) ^ D(14) ^ D(15) ^ D(19) ^ D(23) ^ D(24) ^ D(25) ^ D(29) ;

EB (4) := D(4) ^ D(5) ^ D(6) ^ D(7) ^ D(8) ^ D(9) ^ D(16) ^ D(17) ^ D(18) ^ D(19) ^ D(26) ^ D(27) ^ D(28) ^ D(29) ;

EB (5) := D(10) ^ D(11) ^ D(12) ^ D(13) ^ D(14) ^ D(15) ^ D(16) ^ D(17) ^ D(18) ^ D(19) ^ D(30) ^ D(31) ;

EB (6) := D(20) ^ D(21) ^ D(22) ^ D(23) ^ D(24) ^ D(25) ^ D(26) ^ D(27) ^ D(28) ^ D(29) ^ D(30) ^ D(31) ;

Figure 5: Error detection and correction algorithm

- The AMBA network interface is designed for complex System on Chip (SoC) solutions. It is composed of an AMBA AHB master module used to request read or write operations on the external memory device and an AMBA AHB slave module to access the CCIPC configuration & status area;
- The direct interface is used for small CCIPC solutions in which application objects are directly mapped into FPGA or ASIC, exploiting its register and pins resources

Additional features

The first feature regards the CCIPC behavior when a failure occurs in a CAN network. CCICP utilizes the Heartbeat protocol receiving and transmitting Heartbeat messages to detect failures on the net. When CCIPC does not receive the Heartbeat message from the NMT master node, it generates a “Heartbeat event”. This event is used by CCIPC to implement the “Bus Redundancy Management” protocol. Through this protocol, CCIPC is able to control two CAN networks (nominal and redundant). CCIPC periphery is furnished with a bus selection flag that allows defining which is the CAN network that is currently active.

Two parameters are available to control the redundancy protocol:

- T-toggle counter defines the maximum number of Heartbeat events before switching the bus;
- N-toggle counter defines the maximum number of network toggling before stopping the redundancy process.

Another characteristic of the CCIPC is the implementation of the EDAC algorithm, based on the Hamming code shown in Figure 5, in order to detect and correct single-bit errors and detect double-bit errors in internal and external memory devices.

Graphical user interface

A CCIPC-oriented GUI (graphical user interface) leads the user during the configuration phase of the main CCIPC configuration parameters and also to edit the object dictionary compatible with the EDS (electronic data sheet) standard.

The following items can be set:

- System frequency in the range of 10 to 16 MHz;
- CAN network bit-rate (1 Mbit/s, 500 kbit/s, 250 kbit/s, 125 kbit/s)
- Host Interface;
- PDO communication & mapping parameters;
- Application objects;
- CCIPC configuration objects:
  - Synchronous Window Length;
  - Heartbeat consumer and producer time;
  - Redundancy Manager parameters.