Self-configuring CANopen controller

An approach to further simplify using CANopen devices by providing a self-configuring, minimal, easy-to-use CANopen Manager and NMT master.

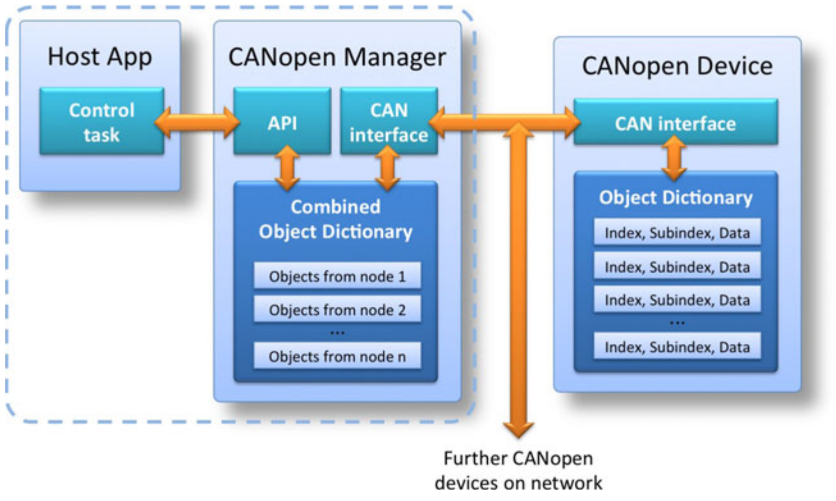

Figure 1: The typical setup - All data objects from all connected devices are combined in the manager's Object Dictionary (Photo: EmSA)

For newcomers to CANopen, the initial hurdle to write a control program for a CANopen system has been relatively high. A system integrator must often go through a lengthy CANopen configuration process. In CANopen, all devices have a unique node ID and a local Object Dictionary containing all parameters that can be communicated. The Object Dictionary entries are addressed by an Index and Subindex value, similar to a lookup table. The manual of a CANopen slave device will typically list and document all available Object Dictionary entries (= parameters).

To successfully connect CANopen NMT slave devices to a network, you need to configure many of them first. Since this is typically the CANopen manager's duty, you need to configure it accordingly as well. This may include a PDO (Process Data Objects) mapping process. Using EDS (Electronic Data Sheet) files, the CANopen Manager learns about these settings and provides local object dictionary entries for every parameter that can then be accessed by the control program.

Figure 1 illustrates the typical setup: all data objects from all connected devices are combined in the manager's Object Dictionary. In many applications this means that the Index and Subindex values for the data objects change. A parameter that is at Object Dictionary entry [6200 01_h] in a CANopen slave device might be placed at a location like [2002 01_h] in the Manager. These re-arrangements can be quite confusing. Also, to access any data objects not part of PDOs, a host would have to access the CANopen Manager's SDO manager function to use SDO read or write cycles. This is part of the CANopen standard, but not straightforward to use in practice.

In contrast, the CANopen knowledge required to write a control program with the new CANopenIA-MGR by EmSA is minimal: All parameters of all slave devices can be immediately accessed without having to know their location in the manager.

This is illustrated in Figure 2. The Manager's local Object Dictionary is no longer used to mirror process data from the slave devices. Instead, the host application can directly reference the Index and Subindex values as used in the slave devices.

With this knowledge, one can start writing a CANopen control program. All data coming in from the CANopen devices is passed on to the control program with the parameters source node ID, source index and subindex. No need ▷

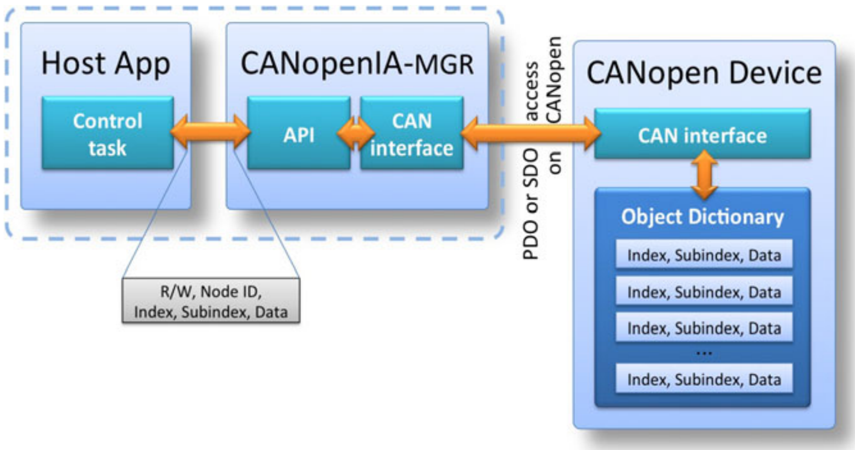

Figure 2: Read or write request to a remote Object Dictionary of a node on the network (Photo: EmSA)

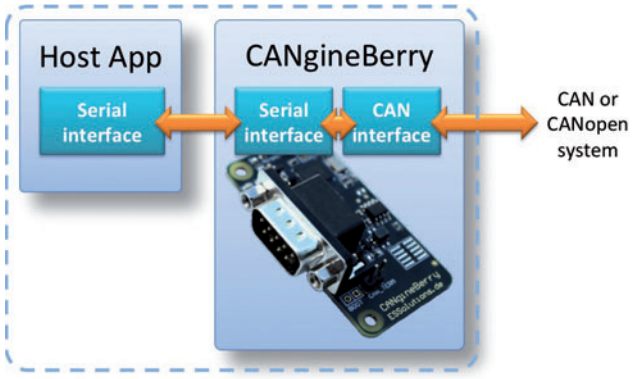

Figure 3: Stand-alone versions of the CANopen Manager communicated to the host via UART (Photo: EmSA)

to consider any "mapping", just receive the data along with the information from which node it comes and which parameter it is. The same concept is used for sending data to the CANopen devices. The control program simply passes the destination node ID and the destination Index and Subindex with the data on to the CANopenIA-MGR process. The CANopenIA-MGR then decides automatically which communication method (for those who know these details: PDO or SDO) is best used to transmit the data.

A generic read and write from the host to all slaves is also possible at all times. All parameters transmitted through PDOs are locally buffered whereas for those not in PDOs, the manager automatically creates the necessary SDO read or write access to the slave devices. This self-configuring CANopen controller scans the network on every power-up to learn all PDO mappings from all slave devices. While this adds a delay of a few seconds on every power-up, this ensures that the controller always makes sure that its internal PDO configuration matches the devices connected to the network.

It also informs the host application about important system events. These include Bootup of nodes, Heartbeat activation or loss, Emergency messages and others. It keeps a local copy of each identification object, so device type and vendor ID can be quickly accessed without generating additional traffic on the CANopen network.

The self-configuring CANopen controller is currently available as a library or dedicated hardware: as a Windows DLL (Dynamic Link Library) for PCAN interfaces from Peak System or as CANgineberry, an active CAN/CANopen module for the Raspberry Pi or other embedded host computers.

The CANgineberry addresses the shortcomings of many "CAN shields" that are passive, have no own intelligence, and require the host computer to handle all CAN communication message by message. In worst case, a CAN system can have more than ten thousand individual messages per second. Sometimes the real-time requirements are below 10 ms for some responses which is difficult to achieve with a Linux or Windows based host and a passive approach. The communication to the host system uses a regular serial channel (TTL-UART), so no special driver is required as UART support is typically part of all operating systems. The CANopenIA-MGR firmware is included with the CANgineberry delivery and can handle up to 32 CANopen devices.

The CANopen Controller Library for PCAN interfaces of Peak-System can handle up to 127 devices. It is a Windows DLL and therefore the timing accuracy and response time greatly depends on the host system and its tasks. However, there are many CANopen control applications where occasional delays of a few tens of milliseconds are perfectly acceptable. Programming examples are provided for C++ and Javascript, examples for other programming languages are available upon request. ◀

Author

Olaf Pfeiffer
EmSA (Embedded Systems Academy)
info@esacademy.com
www.esacademy.de

CiA

CAN in Automation

The non-profit CiA organization promotes CAN and CAN FD, develops CAN FD recommendations and CANopen specifications, and supports other CAN-based higher-layer protocols.

Join the community!

- ▶ Initiate and influence CiA specifications
- ▶ Receive information on new CAN technology and market trends
- ▶ Have access to all CiA technical documents also in work draft status
- ▶ Participate in joint marketing activities
- ▶ Exchange knowledge and experience with other CiA members
- ▶ Get the CANopen vendor-ID free-of-charge
- ▶ Get credits on CANopen product certifications
- ▶ Get credits on CiA training and education events
- ▶ Benefit from social networking with other CiA members
- ▶ Get credits on advertisements in CiA publications

For more details please contact CiA office at headquarters@can-cia.org

www.can-cia.org