Accelerated transfer of CANopen projects into assembly and service

Dr. Heikki Saha

Author
Dr. Heikki Saha
sahat@kotiposti.net

One of the main targets in the industry is to improve operational efficiency by maximizing the re-use of components in as many positions as possible with minimum changes. Typically the target is reached by maximizing the use of nodes according to various CANopen device profiles with application-specific configurations. CANopen defines the system management process and standard design files ([1], [7]). DCF files are designed to be used as information source in configuration downloads and target format in troubleshooting. EDS files are used as templates in both use cases, describing the interface supported by each node.

Components in a distributed system always need information of their position in a target system. Information includes parameters controlling both communication and device behavior. In general, when there may be plenty of similar devices, it is impossible to fully automatically assign them with corresponding target positions without significant constraints. Therefore, the most reliable way is to download configuration node-by-node before the physical installation. It is possible to automatically determine available components, but manual assignment of target positions is required. Isolated control and configuration download applications enable transducer and actuator updates without a need for updating the application, as long as CANopen device profiles are followed.

Uniform way of working is required to enable efficient configuration management in assembly and service. Using more than one approach will multiply the number of required tools and amount of education effort required by assembly and service personnel. Moreover, if multiple approaches are used, the risk of regular problems and support requests increases.

Design requirements

CANopen project is required for each system to support efficient and managed assembly and service. It is most efficient to design systems according to the CANopen system design process resulting CANopen project for the system [7]. Main phases of the system design are definition of singal connections, parameter values and access paths [3]. If a managed process is not used, CANopen project information can also be read from validated prototype and stored as CANopen project by a system design tool. The latter approach is not recommended, because there is a significant risk for unintentionally invalid parameter values.

System design tools use EDS files as standard input for capabilities of the nodes [1] [7]. Interfaces of the application programs can be managed as profile databases to maximize re-usability ([1], [8]). External tools can also be linked to the system design tool with corresponding entries of EDS file [9]. Complete CANopen project consists of DCF file for each node and nodelist.cpj [9] listing the DCF files of the project. In addition to the configuration management, communication database generated by system design tool can be used in diagnostics, rapid control prototyping and simulation. XDD and XDC files cannot be used, because e.g. tool integration is not currently supported [10].

Requirements for download

CANopen systems do not necessarily contain application programmable devices. The most simple control systems can be implemented with standard nodes according to the appropriate device profiles [2]. Therefore, at least configuration management with an external tool needs to be supported to enable uniform way of working with any kind of CANopen system. A clear benefit is, if the same tool can be integrated as part of control system, too. Functional safety standards expect the use of download procedure with download, ▷

References

[1] Saha H., Wikman M., Nylund P., CANopen network design and IEC 61131-3 software design, CAN Newsletter 3/2009, 2009, pp. 52–58

[2] Saha H., Bäck B., Commonality between hydraulic valves driven by general purpose CANopen I/O and hydraulic CANopen drives, Proceedings of 13th iCC, 2012, pp. 11-1 – 11-7

[3] Saha H., Benefits of intelligent sensors and actuators throughout the systems life cycle, The 12th Scandinavian International Conference on Fluid Power, May 2011, Tampere, Finland, I SBN-978-952-15-2517-9, pp. 169–181

[4] CANopen application layer and communication profile, CiA 301, CAN in Automation

[5] CANopen additional application layer functions – Part 3: Configuration and program download, CiA 302-3, CAN in Automation

store, reset and verify and an intuitive user interface to minimize the number of errors [12].

CANopen devices may have any bit-rate and node-ID by default, especially in service actions, where spare parts may also be borrowed from other machines. To enable a uniform handling of any CANopen device, independent of the supported download method for node-ID and bit-rate, a point-to-point connection to a single device is required. Many devices support only LSS switch mode global command or changing bit-rate and node-ID via object dictionary, which cannot be used with more than one node connected.

Node-ID and bit-rate

There are various approaches used for setting node-ID and bit-rate to the devices. The used approach has a strong impact on the overall performance and quality in both assembly and service. The most common approaches are briefly reviewed.

Hardcoded values in application program cannot and need not to be changed, which may sound simple. But, in practice it prevents efficient re-use of the same design in multiple target positions, which violates the re-usability target of modern system design.

The **use of connector pins** was popular approach in the 1990s, but it has several problems. Re-usability is constrained, because supporting the full node-ID space and all bit-rates will reserve too many connector pins. Wiring is also prone to assembly errors and very unreliable in the long term. Typically special tools are required for making such connections inside the connectors. But the most significant problem, especially in service, is that a person with electrics expertise and electric drawings are required. In addition, parameters need to be downloaded, which leads to the use of two different approaches. Expertise is also needed to perform the tasks in a correct order.

Mechanical switches are comparable with connector pins from process and expertise point of view. It is known fact that switches tend to change state unintentionally under vibrations and shocks. Moreover, the use of switches requires at least one additional opening in the housing to provide access to the switches, causing additional sealing challenges and increased size of the node.

Coding plugs has not been so widely used. Supporting the whole node-ID space and all bit-rates will result huge number of different plugs, which will be a logistic nightmare. Also expertise is required, because correct plugs need to be read from electric drawings. The use of constrained number of node-IDs and bit-rates will decrease the re-usability of the nodes. Coding plug connector also increases the size of the node. Missing standard for such plugs may lead to the use of multiple different plug series for different devices, decreasing the logistics efficiency further.

Automatic bit-rate detection sounds attractive, but the use of it requires one node transmitting at fixed bit-rate. If there is more than one node with a fixed bit-rate, a configuration challenge exists again. Instead, it increases complexity of the nodes and node-IDs and other parameters still need to be assigned separately. So, automatic bit-rate detection does not provide generic solution in a system level.

The **use of CANopen object dictionary** for setting bit-rate and node-ID keeps the nodes small and simple [4]. Mechanical unintentional changes are avoided and the overall way of working becomes simple – no electrics or other expertise is needed, because a download tool can read values from DCF files and download them into nodes. In theory there is a risk that an unintentional SDO transaction may corrupt bit-rate or node-ID. To protect against unintentional SDO transactions, manufacturers are using various keywords or other coding methods for the values. The use of such device-specific values introduces another challenge. Therefore the download tools need additional information, which is not stored into DCF files. Activation of the bit-rate and node-ID is not exactly described, but various activation timings are well supported by the current tools using point-to-point connection.

Layer setting services (LSS) provides the same benefits as using the object dictionary and also solves the problems with the object dictionary [6]. LSS defines exactly the procedures needed for setting bit-rate and node-ID, including activation. All parameter values are also standardized, which enables download tools to directly use the bit-rate and node-ID from DCF file without node specific values and procedures.

It can be concluded, that the most simple and efficient approach is to use CANopen selectable node-ID and bit-rate, preferably LSS. All other approaches have significant restrictions or they require special expertise. Node-ID and bit-rate need to be set before installation of the devices – all other parameters may be written in the same time.

[6] CANopen layer setting services (LSS) and protocols, CiA 305, CAN in Automation
[7] CANopen electronic datasheet – Part 1: General definitions and electronic data sheet specification, CiA 306-1, CAN in Automation
[8] CANopen electronic datasheet – Part 2: Profile database specification, CiA 306-2, CAN in Automation
[9] CANopen electronic datasheet – Part 3: Network variable handling and tool integration, CiA 306-3, CAN in Automation
[10] CANopen device description – XML schema definition, CiA 311, CAN in Automation
[11] CANopen device profile for fluid power technology proportional valves and hydrostatic transmissions, CiA 408, CAN in Automation
[12] EN 62061, Safety of machinery. Functional safety of safety-related electrical, electronic and programmable electronic control systems, 2005, p. 201

Application programs are parameter values of domain type objects, when programs are downloaded according to CANopen [5]. Using CANopen conformant program download enables downloading the all required information in a single transaction.

Preparing for download

DCF-files cannot contain the all-necessary information for configuration downloads. In addition to the node-ID and bit-rate object values, hierarchical and human-readable position information cannot be included. Image can be linked by file name, but image name included into control file improves re-usability of the images. Traditionally, all required information is passed to assembly and service as text documents and parameters are written object by object. It is most efficient to download configuration from DCF files to the nodes [3]. Until now, the weakest points have been manually edited control files, manually created from templates. It is most efficient to generate such files directly from CANopen projects.

To enable automatic control file generation, sufficient information needs to be included into CANopen projects. The most important thing is to define, what objects will be downloaded. By default, all objects with access type read-write (RW) are included – write-only (WO) objects are not included, because their values cannot be verified. Access type is fixed and if required, it can be overridden by setting the lowest bit of object flags to one to intentionally prevent the download [7]. System designer shall make sure, that parameter values are defined for all objects to be downloaded. The download tool shall not write to an object, if parameter value is not defined. Missing parameter values indicate that the design may not be ready.

System-subsystem-position-hierarchy is used in the user interface of the example download tool shown in Figure 5. Network name is by default used as both system and subsystem name and node name is used as a position name. If required, the names can be edited manually to more descriptive. The conversion tool adds instructions for the possible modifications as comments into corresponding locations of the control file.

If all objects need not to be or shall not to be written, e.g. factory calibration objects [3], they can be manually removed from the control file. Additional editing is not needed, if target node supports LSS. But if object dictionary is used to set node-ID and bit rate, corresponding objects and values need to be defined because it is impossible to automatically find absolutely correct objects and values from the corresponding DCF file. In some nodes written value differs from the read value. Therefore the written values are defined separately and values used in verification are included as object parameter values.

Case example

Information storage into version control and product data management systems may vary among the system integrators and is not within the scope of this article. File-system-based proof of concept is presented instead.

```
01  @echo off
02  python "C:\...\pco2cpj.py" "%1nodelist.pco"
03  python "C:\...\dcf2cfg.py" "%1nodelist.cpj"
```

Figure 2: An example make.bat with node list transformation and download control file generation

A CANopen project in an example system design tool is presented in Figure 1. Project name is Simple_CANopen and there is a DCF file for each node for storing node configurations. Short node names are used, because they may be used as part of variable names in application programmable devices and variable name length is typically limited. The example tool uses a proprietary node list file for combining DCF files into the project.

File generation can be implemented as a batch file combining several phases. Proprietary node list need to be converted first into a standard file to provide standard CANopen project for further processing. Example system design tool supports a make functionality, which calls the batch file with project path as a command line parameter. Same approach could also be used with tools not supporting make – e.g. an external CANopen tool command, toolbar button or desktop icon may be used instead. An example make.bat is presented in Figure 2.

Example conversion is made in two phases. In the first phase, conversion tool pco2cpj.py in line 2 generates a standard nodelist.cpj from proprietary nodelist.pco. The second conversion tool dcf2cfg.py in line 3 generates the configuration download control file templates from the DCF files. Path to the current project is passed as the first command line parameter. It is possible to include other conversions, e.g. for generation of analyzer setup files or node reference designator plate originals, into the same batch file. Project files are shown in Figure 3, where generated node list and DCF files are highlighted.

Configuration files need to be copied finally to the corresponding folders of the download tool. Dedicated folders for DCF, CFG and image files are used by the example download tool as shown in Figure 4. Fixed ▷

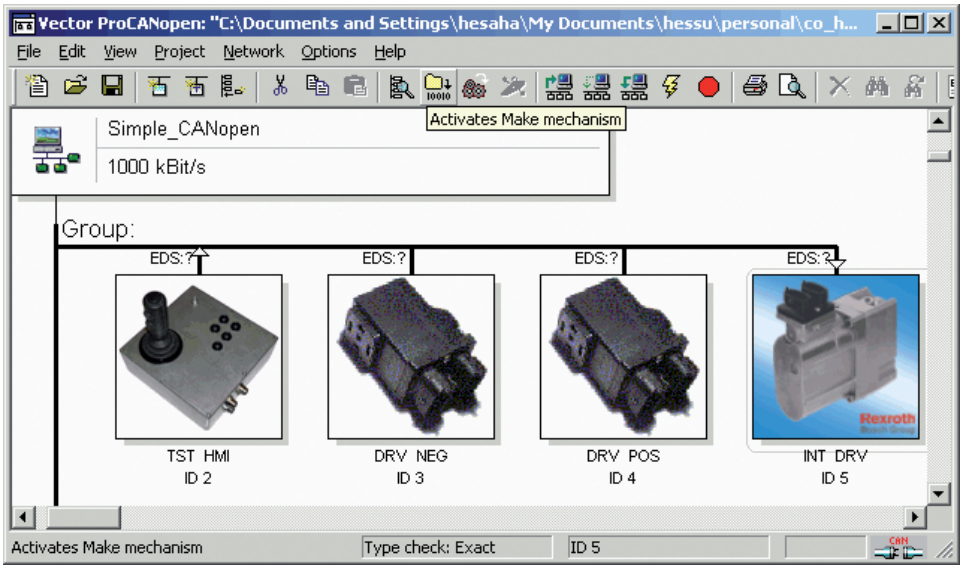

Figure 1: An example CANopen project with highlighted make.bat call

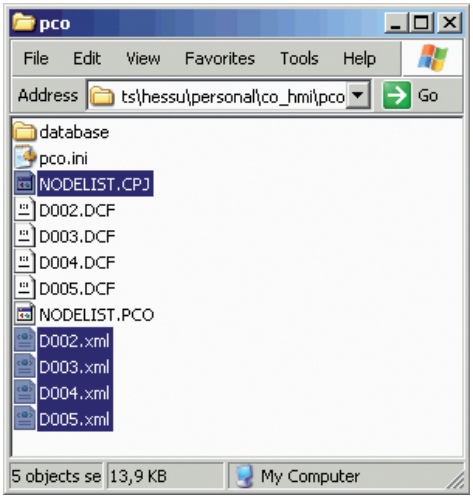

Figure 3: An example CANopen project folder with generated files highlighted

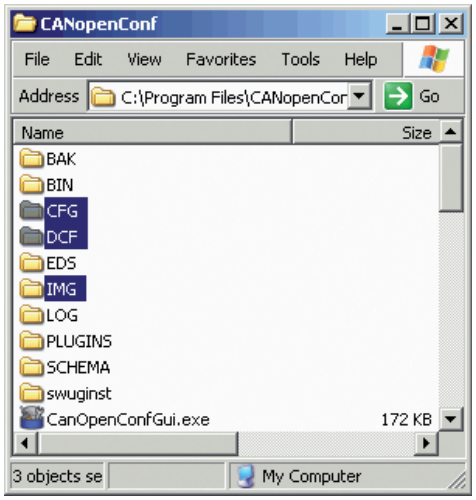

Figure 4: Example CANopen download tool folders with the folders for configuration files highlighted

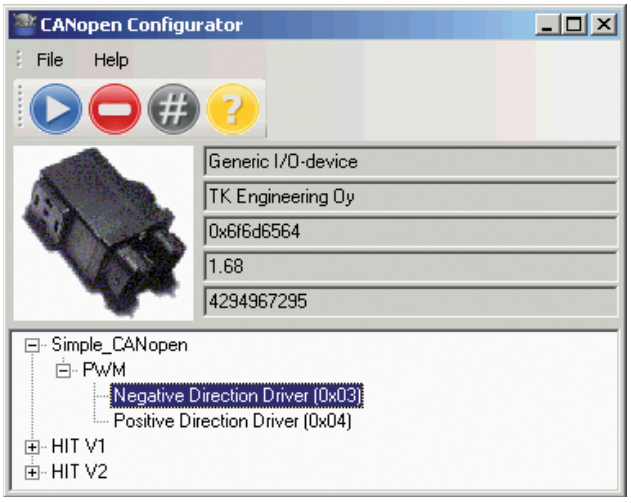

Figure 5: Example of selecting target position for a device connected to the download tool

folders are used, because the configuration download tool may be used in a PC or in a PDA without network connection, and database is too heavy to be used in a PDA. File copy can also be automated, with e.g. previously presented make, but it depends on the storage approach, which is not within the scope of this article.

Configuration downloads are simple. First, operator shall connect the device into the tool, turn power on and start the tool. Then, operator need to select the bit-rate, after which the tool reads the device identity and lists all defined target positions for such device. Finally operator can just select the desired position from the tree view and the tool downloads, stores and verifies the configuration, which may also contain application program(s).

In the example of Figure 5, position names are edited to maximize the understandability of the target positions. Only positions designed for device type and version of currently connected device are shown in the list. Short node names are replaced with long node names, which are more understandable. Also subsystem name was changed manually, because there is not corresponding standard entry available in the DCF file.

Concluding remarks

Incomplete designs cause always problems in assembly and service. When information management is automated, incompleteness of a design is more clearly indicated. All positive results are achieved by following standardized CANopen design process and files and automating the standardized information processing. Based on the evaluation in some real projects, any reasons for deviating from the process were not found.

All CANopen devices can be supported without any constraints. If a device supports standardized SW download interface, also application program(s) can be downloaded together with configuration in a single transaction. The download procedure operates in a plug-to-play principle [3] – plug the device to the download tool, plug the device to the target system and play. Special expertise is not required, operator need to know only where he or she is going to assemble the device. The use of a device image for illustration of target position enables the use of the download tool even by analphabetic persons.

Average reached speed-up for preparing the download was measured to 15x, when some modifications to the control files – e.g. target subsystem name, object list and object details for bit-rate and node-ID – are needed. If the device supports LSS and only target sub-system names need to be changed, speed-up of 60x can be achieved. Absolute modification time for each DCF is less than 30 s in the latter case.

The first challenge was found from CiA 408 device profile [11], where object multiplexing is used in actual value conditioning. Such approach expects the use of multiple values for some objects, which is not supported by current DCF and XDC files. The most attractive solution would be to replace the object multiplexing approach with e.g. the use of array objects.

Second future development could be an improvement of the standard DCF and XDC files. With some additional entries the use of additional download control files could be avoided. At least a separate long name for node and a subsystem name are needed to improve the user interface of the download tool. As long as the use of LSS is not mandatory, details for setting bit-rate and node-ID through the object dictionary are needed.

Third improvement could be introduced in CANopen program download. It is not currently included into the standard, how to activate the so-called boot-loader mode. It is handled by the existing tools, but the complexity of the tools could be reduced by improving the corresponding mechanism in CiA 302-3 [5].