Getting Started with CANopen

Version 1.1
2008-05-29
Application Note AN-AON-1-1102

Restrictions	Public Document
Abstract	This application note explains items which need to be considered when creating a CANopen device or system. The Manager, Systems Engineer, and Software Engineer are the primary audience for this discussion. A basic understanding of CAN and CANopen is required.

Table of Contents

1.0 Overview 2
2.0 Required Resources 2
2.1 Knowledge 2
2.2 Development Tools 2
3.0 Hardware Requirements 2
3.1 CAN Activation 3
3.2 LEDs 3
4.0 Device Functionality 3
4.1 Using Device Profiles 3
4.2 Process Data 4
4.3 Estimating the Number of PDOs 4
4.4 Fixed or Variable Mapping 4
4.5 Transmission Types 5
4.5.1 Event-Driven Transmission 5
4.5.2 Synchronous Transmission 5
4.5.3 Polling Via Remote Request 5
4.6 Establishing the Object Dictionary 5
4.7 Network Management 6
4.7.1 Guarding 6
4.7.2 Heartbeat 6
4.8 Saving Device Configurations 6
4.9 Layer Setting Services 7
5.0 Conclusions 8
5.1 Hardware Issues 8
5.2 Communication Issues 8
5.3 Object Dictionary Organization 8
5.4 Nonvolatile Storage 8
6.0 Questions & Answers 8
7.0 Additional Resources 9
8.0 Contacts 9

1.0 Overview

CANopen is a standardized, highly flexible, and highly configurable embedded network architecture. A large number of CANopen features are optional, thus giving the device or system developer a great amount of flexibility during implementation. This document will assist the user in understanding CANopen's available features and their applicability when developing a CANopen device or system.

2.0 Required Resources

2.1 Knowledge

Since CANopen is a higher-layer protocol based on CAN (Data Link Layer, Layer 2), knowledge of CAN is necessary, especially CAN's communication mechanisms and error handling. Basic knowledge of the following topics should be acquired:

- Dominant/Recessive logic and bit-wise arbitration
- Message frame types (architecture, setup, usage)
- Error handling

Once the basics of the CAN protocol are understood, knowledge of these CANopen features is required:

- Communication mechanisms (PDO, SDO)
- Object Dictionary (interface between application and communication)
- Network Management
- Error handling

2.2 Development Tools

A good set of tools is recommended when developing a CAN or CANopen-based system. In the beginning, it is a good idea to design and simulate the network architecture (prototyping tools are available). An oscilloscope is appropriate for the initial operating phase of a CAN or CANopen device. Once the transceiver has been connected and the bus is physically laid out correctly, then the tool must focus on the CAN or CANopen message frame level. This is where a bus analysis and monitor is needed. The following table is a summary of the types of tools that are available.

Abstraction Level	**Tool Type**
Physical Layer	Oscilloscope, CAN-based oscilloscope
Network (or message level)	Simulator, analysis/monitoring, and testing tools
Application Level (PDO, SDO, NMT)	Device configuration and testing tools

Table 1 – Types of Development Tools

3.0 Hardware Requirements

There are no special hardware requirements for a CANopen device, but there are things to consider, depending on the nature of the application and implementation. One main concern is how the node ID and baud rate will be set for a device. In most situations, switches can be used. If this is not possible, then an alternative way is necessary – for example, configuration with a one-to-one relationship, Layer Setting Services (LSS), plug coding.

3.1 CAN Activation

A CANopen device always has a physical interface according to ISO11898 (High-Speed CAN Physical Layer). The next questions are which baud rates should be supported and how the physical CAN connection should be made. The document *CiA DS 301 - CANopen Application Layer And Communication Profile* lists all baud rates supported by CANopen.

Baud Rate	Comment
1 Mb/s	The bus length is relatively short. With electrical isolation, the length is in the 10-meter range.
800 Kb/s	Many CAN Controllers do not support this rate when certain oscillator frequencies are used.
500 Kb/s	No restrictions
250 Kb/s	No restrictions
125 Kb/s	No restrictions
50 Kb/s	Newer transceivers have an approximate lower limit of 50 - 60 Kb/s.
20 Kb/s	Newer transceivers have an approximate lower limit of 50 - 60 Kb/s.
10 Kb/s	Newer transceivers have an approximate lower limit of 50 - 60 Kb/s.

Table 2 – CANopen Baud Rates

At a minimum, 125 Kb/s, 250 Kb/s, and 500 Kb/s should be supported during normal operation. CANopen does not require specific connectors. There are recommendations in *CiA DR 303-1 - CANopen Cabling And Connector Pin Assignment*.

3.2 LEDs

If a device has display indicators (LEDs), the colors can be selected according to *CiA DR 303-3 - CANopen Indicator Specification*. This document explains that CANopen-specific states can be displayed with two LEDs or one bicolor LED.

4.0 Device Functionality

When CANopen is implemented in a device, many of the CANopen features are optional. Note that if all optional features were left out, the device would have very little useful functionality. The intention of this section is to help you make decisions that make the functionality of a device easier. Ultimately, the goal is to achieve a good balance between the functionality that will be implemented and the amount of resources used.

4.1 Using Device Profiles

Device profiles describe the functionality of a particular device class within the CANopen standard. Each device must include the minimum functionality that is defined for that particular device class. Many profiles are available for several different device classes. The user should know if a device class already exists for the device to be developed. If one exists, then the corresponding specification must be referenced. In most cases, only the minimum functionality is defined. There are also rules regarding the state machine for the particular application, as well as the standard configuration of Process Data Objects (PDO). If a device class closely matches the device under development, then the profile specification can be used. If the device cannot be associated to an existing device class, then the device needs to be developed according to *CiA DS 301 - CANopen Application Layer And Communication Profile*. The advantages of a standardized device profile are now lost (interoperability).

4.2 Process Data

In order to determine the necessary CANopen functionality, it is essential to know the entire amount of process data that will be required. Process data is the data that is to be transmitted in a PDO. Process data transmitted in either a Send or Receive PDO is evaluated separately in each case. The user must determine how much data should be transmitted with the PDO mechanism. The number of PDOs required can be calculated directly from this number. The rate at which the data is presumed to change must be calculated as well. Critical data should be specifically labeled.

4.3 Estimating the Number of PDOs

When the amount of process data is known, then it is relatively easy to determine the number of PDOs required. Every PDO can transmit up to 8 bytes of process data. The easiest way to proceed is to fill the PDOs step-by-step with process data. Once all process data has been placed in the appropriate PDOs, the number of PDOs is then set. Since this method does not always end with the best result, a few boundary conditions should be considered. The goal should be to implement the smallest possible number of PDOs in one device.

A device often has special modes of operation. Specific process data can only be exchanged depending on the mode of operation. It is easy to determine how the process data can be assigned to the different modes of operation using a table, which often results in the number of PDOs required. Note that a new configuration for the PDO is usually needed whenever the mode of operation changes. Each PDO should transmit uninterrupted and discrete process data. When continuous data is being transmitted, communication is mostly synchronous or cyclic. Continuous data that changes either slowly or quickly should be placed in different PDOs. Discrete data that is event-driven can be assigned to one PDO (or several). Data that has similar update rates can be found in one PDO.

PDOs can also be quite extensively configured. It is possible to change the meaning of the information transmitted independent of run-time requirements (mapping). The "mapping" functionality naturally requires internal resources, notably execution time and memory. One question may be "Where is this flexibility worthwhile?” Certainly not for a device that only has 8 digital input ports. Here, one single Transmit PDO (TPDO) with fixed mapping can be used, but there is no general rule. When a device has several configuration options, variable mapping usually cannot be avoided.

Theoretically, every CANopen device can have up to 512 PDOs per communication direction. It is conceivable that modules will be in a centralized system architecture that must send and receive many PDOs. If this is the case, keep in mind that CAN is a serial arbitration system. This means the update rate is relatively low when large quantities of data are being transferred.

4.4 Fixed or Variable Mapping

As mentioned earlier, mapping determines the data to be transferred in a PDO. PDO mapping is managed in an internal mapping table. According to *CiA DS 301*, this table can contain up to 64 entries. This table size allows every bit in the data field of a CAN message to be described. However, keep in mind that you must run through every entry in the variable mapping table for every PDO received. The meaning of every bit for the application can only be conclusively determined this way. This also applies for PDOs that have to be sent. This means that interpreting the PDO is what results in a relatively high processor load.

In practice, bitwise mapping is rarely used. The majority of implementations use a single byte as the smallest resolution for mapping, so that the mapping table has at most 8 entries.

Variable mapping guarantees the required PDO flexibility in CANopen. There are also a number of simple CANopen devices that can get by with fixed mapping. This is usually possible when a device sends or receives a single PDO that is not completely filled with application data.

4.5 Transmission Types

Setting the transmission type sometimes influences the amount of device resources used. The device developer decides which transmission types are available for a PDO. The different transmission types are:

4.5.1 Event-Driven Transmission

Event-driven transmission is closest to the nature of the CAN protocol. A TPDO is triggered (or sent) when an event occurs which is linked to the data that is to be transmitted. When this happens, it is usually a change in the data that leads to triggering a TPDO.

TPDOs can also be directly linked to a timer. The expiration of the timer leads to the message being sent, regardless of changes in the data or other events. Thus, the timer is activated cyclically.

Since events can occur on different devices within a small window of time, temporary high bus loads may occur during event-driven communication. Event-driven communication with an RPDO means that the received data is processed immediately.

4.5.2 Synchronous Transmission

Synchronous transmission of a TPDO is triggered with the "SYNC" message. The SYNC message is pre-defined and a SYNC producer is required. Due to the serial nature of CAN, equidistance between SYN messages cannot be guaranteed. Because of this, another mechanism (or system) should be used to fulfill these higher synchronization requirements. Synchronous transmission is for continuously changing data, since there is no dependence on transmitting every individual value. If discrete values are transmitted synchronously, then the synchronization time must be chosen in such a way that changes to the data can be detected with certainty.

4.5.3 Polling Via Remote Request

A TPDO can be polled with a Remote Request. This type of communication adds a considerable amount of overhead to the system and offers no real advantage over the other transmission types.

4.6 Establishing the Object Dictionary

The Object Dictionary allows device properties to be accessed over the network. This means all changeable data must also be accessed the same way. In general, any communication property selected has a corresponding entry in the Object Dictionary. For example, when a PDO is implemented, the corresponding configuration entries (mapping and configuration parameters) must also be made available in the Object Dictionary. These entries are mandatory.

The structure of the Object Dictionary makes it possible for logical entries to be combined, like inter-related data being ordered into one object. A good example is a control unit with corresponding parameters. The control unit is addressed with a main index and the individual parameters accessed with alternate sub-indices. Every object in the Object Dictionary must contain the following:

- Object index and sub-index (each index pair within a device must be unique)
- Entry size (usually results from the data type)
- Value range
- Read and write attributes
- Mapping (determine whether or not the object can be transmitted with one PDO)

The Object Dictionary must also be described in electronic form (electronic data sheet or EDS) according to *CiA DSP-306 - CANopen Electronic Data Sheet (EDS) For CANopen*. A proper EDS is required for effectively using configuration tools. Commercially available editors are available to assist in EDS construction and testing (according to the standard).

4.7 Network Management

CANopen defines two different mechanisms for device supervision or network management. These mechanisms are called Guarding and Heartbeat.

4.7.1 Guarding

When Guarding is used, the device waits for a special cyclical message to be sent and replies by transmitting the current status of the device. Guarding is triggered by one node in the network, and this node is called the NMT Master. Because of this requirement, it is not possible for a non-NMT Master module to check and verify the existence of another module in the network.

Since this inflexible network management style leads to relatively high overhead, Heartbeat was designed, and it is used in all modern CANopen implementations.

4.7.2 Heartbeat

There are two types of devices in this management scheme: a Heartbeat Producer and Heartbeat Consumer. The Heartbeat Producer sends a cyclic message containing its current status. The Heartbeat Consumer contains a specific Object Dictionary configuration, allowing it to monitor this Heartbeat message within a given period of time. If this message is not sent, the Heartbeat Consumer records this event. Heartbeat is far more flexible than Guarding, since it is not bound to a Master-Slave hierarchy.

In general, Heartbeat is preferred with newer CANopen implementations. However, a certain amount of care should be taken to ensure the Heartbeat Consumer has enough available resources to monitor within the required amount of time.

4.8 Saving Device Configurations

CANopen is very flexible because it can have various levels of configurability. Configurations are implemented by accessing Object Dictionary entries via the network. Nonvolatile storage is required if it is desirable for the device configuration to be available after the device has powered up again. If nonvolatile storage for configurations is not possible, then the configuration must be implemented every time the device is powered up. Storage makes practical sense for the following types of data:

- Application parameters (e.g., controller characteristics, characteristic curves, limit values)
- Communication parameters (for example, PDO-relevant parameters)

While the amount of storage for the necessary application parameters cannot be estimated, the requirements for the communication parameters can. When doing this, care must be taken to ensure that the largest amount of configuration data for the PDO is calculated.

The amount of mapping data for a PDO with byte-wise mapping is:

  1 x 1 byte (sub-index 0)

<u>+ 8 x 4 bytes (mapping entries)</u>

  33 bytes

The amount of data required to describe the communication properties for a Receive PDO (RPDO) is:

   4 bytes (identifier)

+ 1 byte (transmission behavior)

   5 bytes

The amount of data required to describe the communication properties for a Transmit PDO (TPDO) is:

   4 bytes (identifier)

+ 1 byte (transmission behavior)

+ 2 bytes (inhibit time)

+ 2 bytes (PDO timer)

   9 bytes

If the device has 4 PDOs for each communication direction and that configuration must be saved, then the following amount of storage is required:

   2 x 4 x 33 bytes (mapping table)

+ 4 x 5 (RPDO communication parameters)

+ 4 x 9 (TPDO communication parameters)

   320 bytes

4.9 Layer Setting Services

Since a CANopen network requires a consistent baud rate, and since all device identifiers are derived from the node ID, CANopen provides a service called Layer Setting Services (LSS). LSS is used for setting the baud rate and node ID via the network. Two nodes on the same network should not transmit the same CAN identifier.

The deciding factor for implementing LSS is whether setting the baud rate and node ID is a one-to-one session between the configuring node and the node to be configured, or whether all devices in the network are to be configured in an already existing network. When there is a one-to-one session the configuration is trivial, since there is only one device in the network that needs to be configured. Using LSS for the complete installed network is much more complex. When the node ID is being set in this situation, only one device can be in configuration mode at one time. The device to be configured must also be clearly identifiable on the network so that it can be selected. This also applies when devices are identical in functionality. In this case, a serial number and a vendor ID can differentiate the nodes. The combination of both serial number and vendor ID must be universally unique. Because of this, a unique serial number must be assigned to a device where LSS will be used during the manufacturing process.

5.0 Conclusions

When developing a CANopen device or system, the following questions need to be answered:

5.1 Hardware Issues

- How should the physical interface be created? Galvanic electrical isolation or not?
- Are there going to be switches for setting the baud rate and the node ID?
- Should the LEDs be controlled according to CiA's recommendations?

5.2 Communication Issues

- Does a device profile exist which could be applied to the device under development?
- How much process data needs to be transmitted?
- How many PDOs are needed?
- Should variable or fixed mapping be used?
- Which PDO transmission types should be supported?
- What Network Management style should be used? Heartbeat or Guarding?

5.3 Object Dictionary Organization

- How should the process data be organized in the Object Dictionary?
- What parameters from the CiA DS 301 should be included in the Object Dictionary?

5.4 Nonvolatile Storage

- Will the device configuration be saved in nonvolatile memory?
- What memory capacity is required?

6.0 Questions & Answers

Where can I find additional information about CANopen?

The first step should be to contact CAN in Automation (CiA) International Users and Manufacturers Group (http://www.can-cia.com), which has information and the actual CANopen specification.

Is Certification Available For CANopen Devices?

CiA has established a conformance test and offers device certification. However, this test is only for the device itself and doesn't include a test for interoperability.

Where Can I Get A Vendor ID?

The vendor ID is acquired from the CiA.

7.0 Additional Resources

Additional information can be found in the following:

VECTOR APPLICATION NOTES

AN-ION-1-1100 Introduction to the CANopen Protocol

AN-AON-1-1101 Introduction to the CANopen Documentation Family

AN-ION-1-1103 Master and Slave in the CANopen World

AN-ION-1-1104 Setting Baud Rate and Node IDs in a CANopen System

AN-AND-1-100 Business Introduction to the CAN Protocol

AN-AND-1-101 Technical Introduction to the CAN Protocol

AN-AND-1-102 Beginner's Guide to CAN Development

AN-AND-1-106 Basic CAN Bit Timing

AN-AND-1-108 Glossary of CAN Protocol Terminology

AN-AND-2-135 Advanced CAN Bit Timing

AN-ANI-1-115 Common High Speed Physical Layer Problems

8.0 Contacts

Vector Informatik GmbH Ingersheimer Straße 24 70499 Stuttgart Germany Tel.: +49 711-80670-0 Fax: +49 711-80670-111 Email: info@vector-informatik.de	**Vector CANtech, Inc.** 39500 Orchard Hill Pl., Ste 550 Novi, MI 48375 USA Tel: +1-248-449-9290 Fax: +1-248-449-9704 Email: info@vector-cantech.com	**VecScan AB** Lindholmspiren 5 402 78 Göteborg Sweden Tel: +46 (0)31 764 76 00 Fax: +46 (0)31 764 76 19 Email: info@vecscan.com
Vector France SAS 168 Boulevard Camélinat 92240 Malakoff France Tel: +33 (0)1 42 31 40 00 Fax: +33 (0)1 42 31 40 09 Email: information@vector-france.fr	**Vector Japan Co. Ltd.** Seafort Square Center Bld. 18F 2-3-12, Higashi-shinagawa, Shinagawa-ku J-140-0002 Tokyo Tel.: +81 3 5769 6970 Fax: +81 3 5769 6975 Email: info@vector-japan.co.jp	**Vector Korea IT Inc.** Daerung Post Tower III, 508 Guro-gu, Guro-dong, 182-4 Seoul, 152-790 Republic of Korea Tel.: +82-2-2028-0600 Fax: +82-2-2028-0604 Email: info@vector-korea.com