Configuring and monitoring CANopen devices in mobile applications

Uwe Koppe (MicroControl GmbH & Co. KG)

CANopen is a widely used network in mobile applications. Due to the limited capabilities of mobile PLCs the configuration and monitoring of CANopen devices becomes often a challenging task. This paper focuses on the configuration process and security issues for a CANopen network.

Running a CANopen network (figure 1) is typically a two-step process. After power-up the PLC starts the configuration phase, where parameters in all CANopen devices are set according to the Device Configuration File (DCF). In phase 2 all CANopen nodes are started (i.e. switched to Operational state) and PDO communication is started on the bus.

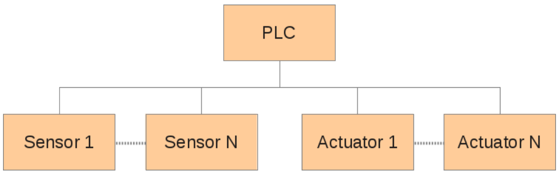


Figure 1: CANopen network structure

Although the “NMT start-up” process is described in detail in the CiA 302-2 specification /2/ (figure 2), we have never seen it to its full extend in CANopen networks, especially for mobile applications.

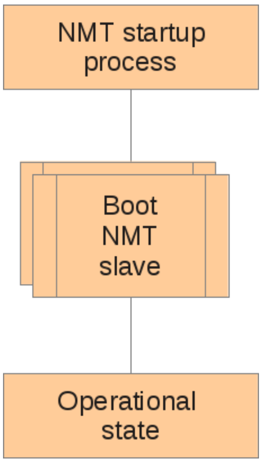


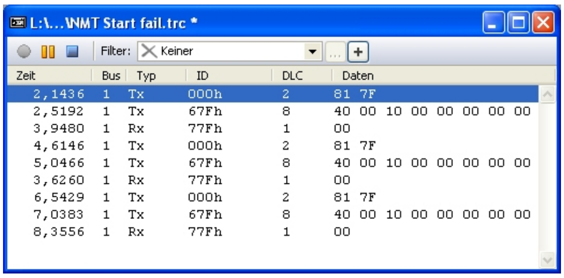
Figure 2: Simplified NMT start-up

Getting the correct device state

The first step in the configuration phase is to get all CANopen nodes in the correct device state, which is typically the Pre-Operational state. This can be achieved by sending a *Reset communication* command with node-ID set to 0 or an individual addressing to each CANopen device. Care has to be taken for the last option: NMT burst typically have unpredictable side-effects. For that purpose the *NMT inhibit time* (object 102A$_h$) has to be taken into account.

Tip 1: Avoid bursts of NMT messages

After the NMT *Reset communication* command has been sent, a device specific delay has to be taken into account. We observe an increasing number of NMT manager implementations which do not check for the boot-up message, but start SDO traffic immediately after the NMT command (figure 3).



```
Zeit     Bus  Typ   ID     DLC  Daten
2,1436   1    Tx    000h   2    81 7F
2,5192   1    Tx    67Fh   8    40 00 10 00 00 00 00 00
3,9480   1    Rx    77Fh   1    00
4,6146   1    Tx    000h   2    81 7F
5,0466   1    Tx    67Fh   8    40 00 10 00 00 00 00 00
3,6260   1    Rx    77Fh   1    00
6,5429   1    Tx    000h   2    81 7F
7,0383   1    Tx    67Fh   8    40 00 10 00 00 00 00 00
8,3556   1    Rx    77Fh   1    00
```

Figure 3: Respect time for initialization

Tip 2: Allow an individual time period for each CANopen device and test for the boot-up message within this period.

Device Identification

The next important step inside the initialization phase is the identification of the device. This means the objects 1000h (Device type) and 1018h (Identity) have to be compared with the DCF values. There are two reasons not omitting the device identification. First, over the life cycle of a machine, the manufacturer of a sensor or actor will probably change the device firmware or even the hardware. From our experience this will have an impact on the application software, leading to a machine failure in worst case scenario. Typical effects are:

- modification of communication timings, e.g. SDO timeout
- modification of the object dictionary (changed parameter behavior)

Keep in mind that the software for the mobile application has been tested with a defined firmware version of all CANopen modules, a firmware update of a CANopen modules would require a re-testing.
The second reason for device identification is service situations, when a malfunction CANopen node is replaced by a spare part (but occasionally not the correct one).

In addition to the mentioned objects 1000h and 1018h it is a good choice to read the optional objects 1009h (manufacturer software version) and 100Ah (manufacturer hardware version), which extend the device signature.

Index	Name
1000h, Sub 0	Device type
1018h, Sub 1	Vendor ID
1018h, Sub 2	Product code
1018h, Sub 3	Revision number
1018h, Sub 4	Serial number (optional)
1009h, Sub 0	Manufacturer hardware version (optional)
100Ah, Sub0	Manufacturer software version (optional)

Table 1: Device identification objects

Tip 3: Always perform a device identification

Device Configuration

After module identification, the device configuration can start. Again, the CiA 302 specification /2/ is a rich source, which is only rarely practiced.

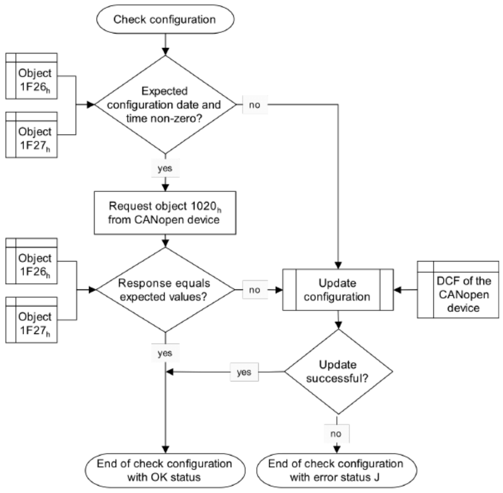

Check configuration
Object 1F26$_h$
Object 1F27$_h$
Expected configuration date and time non-zero?
no
yes
Request object 1020$_h$ from CANopen device
Object 1F26$_h$
Object 1F27$_h$
Response equals expected values?
no
Update configuration
DCF of the CANopen device
yes
yes
Update successful?
no
End of check configuration with OK status
End of check configuration with error status J

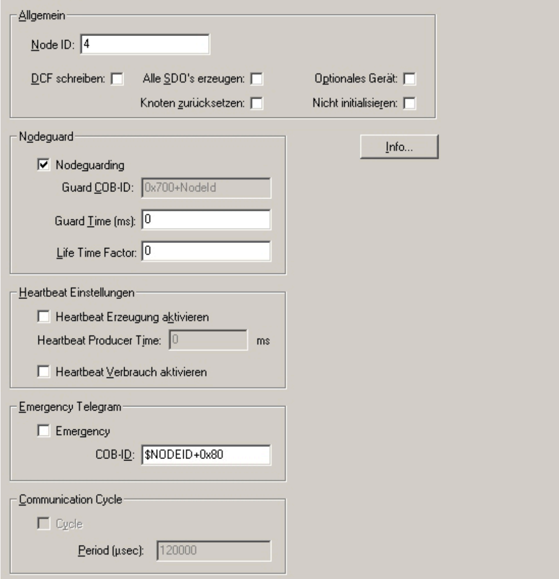
Figure 4: Check configuration flowchart

Whenever possible, request the object 1020h (Verify configuration) from the CANopen device (figure 4) and update the configuration with the DCF. The advantage of the “Check Configuration” procedure is a significantly reduced boot time. If the device does not support object 1020h, never assume that it has the default parameters loaded (Pre-defined connection set and application parameters). Other values than the default ones could have been stored in the device. In addition, we observe during tests with the current CANopen Conformance Test (CCT) that often the real default values differ from those defined in the EDS file. As a conclusion, a “Restore Default Parameters” command is required. All objects which have an impact on the communication (e.g. PDO communication and mapping) and the application have to be written afterwards.

Tip 4: Never assume any default parameters. Try to use the “Check configuration” procedure if supported.

The “Remote Frame issue”

Users of a PLC just want to monitor the NMT state of a CANopen device, they do not care for the used CANopen service. Unfortunately, many PLC manufacturers still offer Node Guarding as default service (figure 5) and this service uses the “banned” CAN Remote Frames for communication.


Allgemein
Node ID: 4
DCF schreiben: □ Alle SDO's erzeugen: □ Optionales Gerät: □
Knoten zurücksetzen: □ Nicht initialisieren: □

Nodeguard
☑ Nodeguarding
Guard COB-ID: 0x700+NodeId
Guard Time (ms): 0
Life Time Factor: 0

Heartbeat Einstellungen
□ Heartbeat Erzeugung aktivieren
Heartbeat Producer Time: 0 ms
□ Heartbeat Verbrauch aktivieren

Emergency Telegram
□ Emergency
COB-ID: $NODEID+0x80

Communication Cycle
□ Cycle
Period (µsec): 120000

Figure 5: Node Guarding set as default

The application note AN 802 /3/ explains in detail why Remote Frames should not be used in CAN networks. This statement is valid for the Node Guarding as well as for PDO communication. As result, make sure that the PLC uses “Heartbeat” for monitoring the NMT state.

Tip 5: Never use Node Guarding or request PDOs with a Remote Frame. Use Heartbeat for NMT state monitoring.

In addition, the Heartbeat service uses a producer / consumer model. This offers the possibility that CANopen NMT slave devices can supervise the state of other nodes. The object 1029h (Error behavior) can be used to achieve a specific device reaction.

Table 2: Objects for NMT state monitoring

Index	Name
1016h	Consumer heartbeat time
1017h	Producer heartbeat time
1029h	Error behavior

The “Bitrate issue”

Mobile applications often require a high protection class (IP66 or higher), which contradicts the use of switches for setting node-ID and bitrate. For configuration of both parameters the CANopen Layer Setting Service (LSS) is available. The bitrate configuration of a CANopen device should always be made before it is installed inside the network. Modification of the bitrate “on the fly” is not recommended and can lead to a bus-off condition which can only be solved after all devices have been uninstalled from the network. A number of vendors offer CANopen devices with Auto-Bitrate Detection, which have the advantage that no specific software is required for configuration. Devices which allow setting of node-ID and bitrate via objects in the manufacturer specific area should never be used in mobile applications. These devices do not pass the CANopen Conformance Test and are a reliable source for trouble in the field.

Tip 6: Use Auto-Bitrate Detection instead of LSS bit-timing service. Never use devices with objects for node-ID and bit-timing.

EDS Administration

The distribution and administration of EDS files can be a challenging process, especially for PLCs with limited resources. The EDS file is parsed by the PLC to generate a list of all objects in a CANopen device together with their default values. The parser typically does not complain about wrong syntax of the EDS file or missing entries. As a result, the application software might work with incomplete or incorrect data.

The situation with bad EDS files has become better in the last couple of years, but the advice is still to check all required EDS files with the freely available EDS checker software /4/.

Tip 7: Test all required EDS files

Switching to Operational State

Having successfully survived the configuration phase, its now time to switch into the NMT Operational state. Please still remember tip 1: do not use a burst of NMT messages when switching from Pre-Operational to Operational state.
From our experience, most application programmers assume in this phase that PDO communication is running as expected on the bus and *all* PDO data is exchanged. Only a few number of system designers perform an exact calculation of the average bus-load in advance. Take a CANopen network with a bitrate of 125 kBit/s and a SYNC cycle period of 10 ms as example. The given SYNC cycle time allows 9 PDOs (with a DLC of 8 bytes) to be transferred (figure 6).

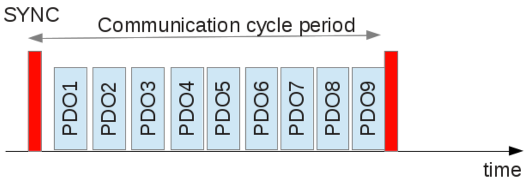


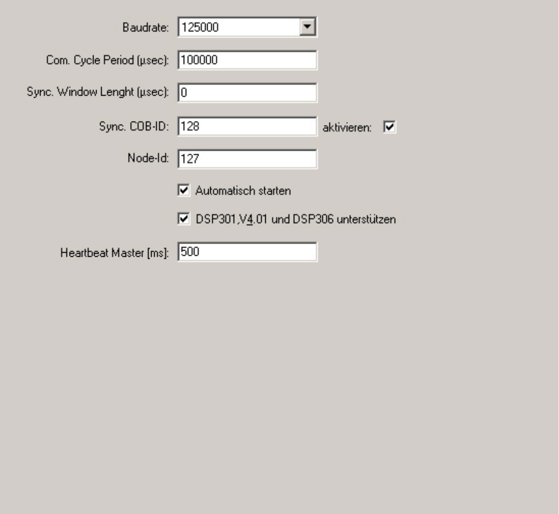
Figure 6: SYNC cycle period

Configuring more than 9 PDOs in this example for SYNC transmission will not lead to an error in any CANopen device. The PDOs simply are not transmitted due to their lower priority identifiers.

Tip 8: Calculate the average bus-load in advance

The advantage of SYNC based PDO communication instead of event-driven communication is not only the precise calculation of bus-load, also the reaction time on device failures is much shorter compared to heartbeat monitoring.

Care has to be taken when the SYNC cycle period is configured. All CANopen timing parameters are a multiple of 1 millisecond, with one exception: the “Communication cycle period” (object 1006h) is a multiple of 1 microsecond. It is a common mistake to write a value less than 1000 in the configuration dialog of the PLC software.



Baudrate: 125000

Com. Cycle Period (µsec): 100000

Sync. Window Lenght (µsec): 0

Sync. COB-ID: 128 aktivieren: ☑

Node-Id: 127

☑ Automatisch starten

☑ DSP301,V4.01 und DSP306 unterstützen

Heartbeat Master [ms]: 500

Figure 6: Communication cycle period setup

The wrong configuration can only be discovered with an external CAN monitoring tool.

Tip 9: Check the cycle time of the SYNC producer

Handling Emergency messages

For the CANopen service of Emergency (EMCY) messages we observe two kinds of implementations in the field: either they are fully ignored or the occurrence of an EMCY identifier causes a complete application software shutdown by the PLC. Both implementations are not leading to productive results.

For testing of application error conditions (e.g. sensor break on an input, short-circuit of an output) the EMCY service is the first choice, rather than polling information via the SDO service. EMCY messages have a high-priority identifier (Pre-defined connection set), which can be modified by the application software.

In order to reduce the bus-load - and also the EMCY buffer on the PLC - it is possible to program a EMCY inhibit time on the CANopen module.

Index	Name
1014h	COB-ID EMCY
1015h	Inhibit Time EMCY
1028h	EMCY consumer

Table 2: Objects for EMCY service

Tip 10: Enable support of the EMCY service in the application software and make sure the objects for the EMCY service are set to the desired values

Just lurking on the EMCY identifier is not enough, the application software has to evaluate the message contents. Imagine a situation where a CANopen module is powered on as first device in the network. Because of the missing acknowledgment on CAN layer 2 the device will move into error passive state, which will cause an EMCY message with error code 8120h (CAN in error passive mode). This is typically not a reason to stop the application software. This example shows that EMCY messages have to be grouped into several error classes with different reactions.

*Tip 11: Evaluate the EMCY message contents and write appropriate handlers for the possible error code*s.

Diagnosis

A CAN layer 2 diagnosis is a missing feature in all known PLCs on the market. For a programmer of a PLC the wish list includes:

- Monitoring of error frames
- Monitoring of device NMT state
- Monitoring of PDO raw data

The only workaround is a CAN / CANopen monitoring tool, which should also support reading and writing DCF information. Especially for the software support team the latest feature is a “must have”, since it allows an examination of the device configuration in the field.

Improving network security

In NMT Operational state the PLC should not only be responsible for exchange of PDO data. If done by mistake or by intention – a man-in-the-middle attack with NMT or SDO messages on the remotes nodes has to be discovered. It is only possible to have one NMT manager and one SDO manager in the network at the same time. Adding an additional configuration tool or a second PLC for redundancy requires support of the NMT Flying master process.

For simple mobile applications, an alien NMT or SDO message shall lead to an immediate NMT stop remote node command.

Tip 12: Observe all network traffic during operation, especially alien NMT and SDO messages.

Conclusion

The configuration process of CANopen devices in mobile applications should be as close as possible to the existing standard CiA 302. CANopen devices that have passed the CANopen Conformance Test allow a shorter integration time and don't show “hidden features” in the field. All hints given in this paper are based on a long time experience with CANopen networks in mobile applications and reflect the most common pitfalls.

Uwe Koppe
MicroControl GmbH & Co. KG
Lindlaustr. 2c
53842 Troisdorf
+49 2241 25659-0
+49 2241 25659 - 11
koppe@microcontrol.net
www.microcontrol.net

References

[1] CiA 301, CANopen application layer and communication profile

[2] CiA 302, Additional application layer functions, Part 2: Network management

[3] AN 802, V1.1.0: CANopen application note – CAN remote frames: Avoiding of usage

[4] EDS checker software, Version 2.2.3, http://can-cia.org/index.php?id=downloads-eds-checker