Approach for node-ID negotiation in CANopen

The increasing complexity and size of CANopen (FD) networks create new challenges on how to decide which node-ID should be used. This article discusses a theoretical approach how devices in a CANopen (FD) network could negotiate their node-ID by themselves.

The assignment of a node-ID to a CANopen device is an essential requirement. Without a node-ID a CANopen device is unable to communicate over services defined in CiA 301 and CiA 1301. Without a valid node-ID in the range of 1 to 127, a CANopen device would never leave the NMT (network management) sub-state reset communication. Since the node-ID has to be unique in the network, there has to be a way to configure the node-ID value. Neither CiA 301 nor CiA 1301 specify a way to configure this value. There are various solutions on the market how this can be achieved.

Some manufactures use hardware solutions on their devices, such as dip- or rotary-switches, or encode plugs. Others, use some kind of proprietary software configuration. All of these solutions need more or less knowledge about the system where the devices are integrated. Another way to configure the node-ID is specified in the document CiA 305, CANopen layer setting services (LSS). This specification describes services and protocols for identifying CANopen devices and assigning node-IDs, which can be used by a so called LSS manager (formerly named LSS master) or some configuration tool.

The CANopen application profile for building door control, CiA 416, also specifies a procedure for claiming node-IDs by the devices themselves. But besides that, this procedure is slow, it is also patented. The following theoretical approach for negotiation of node-IDs introduced by the engineers of Emotas Embedded Communication should be a starting point how to overcome most of the disadvantages of the other solutions.

Systems without a superordinated manager

Some systems are designed very flexible. That means that a system could be built with different combinations of devices, exactly meeting the needed requirements.

Sometimes, it could be enough that only two devices are needed, sometimes much more, and sometimes there are multiple of the same device type in a network. An example of such kind of systems are heating systems based on CANopen. They can include multiple sensors and controllers depending on location and size where the system is installed.

When creating the devices for such a system, it is not clear which kind of device could be unique in the system, so it can operate as a CANopen manager. The CANopen manager is the device with NMT manager functionality plus additional functionality, for example, the LSS manager.

One way to assign node-IDs without an LSS manager is when integrating the system. But this leads to a static system, without a real plug-and-play possibility. And, in case some parts are added, or have to be replaced, an unused node-ID has to be assigned to the replaced part, which might be impossible for a service technician.

Another example are battery clusters, which are capable of plug and play. The only devices in this network are the batteries themselves. They are physically all of the same device type and have the same software. In such a system, the node-ID is irrelevant to the system, but is required for CANopen communication.

Systems with a superordinated manager

In generic CANopen systems with a superordinated manager, there are also possibilities that the actual distribution of the node-IDs is not relevant for the functionality of the system.

Examples are the CANopen application profile for energy management systems, CiA 454, or the CANopen application profile for special-purpose car add-on devices, CiA 447. These specifications use the LSS Fastscan procedure, defined in CiA 305, to detect devices and to assign node-IDs to them. With LSS Fastscan devices can be only detected one after another, and in the worst case of completely unknown devices it will take 128 messages to verify every single bit of the CANopen LSS address. With a response-timeout of 10 ms, it would take 1,28 s to detect one single device.

New approach

The idea behind this approach is that, in systems which do not depend on the node-ID distribution, the devices negotiate their node-ID themselves. The idea is not new, for example in the document J1939-81 of the SAE (Society of Automotive Engineers), there is a description of a so-called address claim procedure. In this procedure, the devices in a CAN-based network negotiate the addresses (node-IDs), depending on values of the Name field. The problem here is that this procedure uses the 8-byte data field of the CAN frame. This led to the fact, that, in case two or more devices are claiming the same address, it could lead to collisions on the CAN network. In our approach, we are trying to avoid this by not using the data field: All the information exchanged between devices are encoded in the identifier field.

The main requirements for this procedure are that the software of the devices is able to send and receive CAN data frames in classical extended frame format (CEFF). Because most modern CAN FD controllers support sending and ▷

receiving this kind of data frames, this approach works in CANopen FD as well. It is also important that the software can distinguish between extended and basic frame formats.

Usage of the CAN-Identifier

The CAN-Identifier (CAN-ID) of an extended data frame has 29 bits available. For our approach we only need 13 bits. The least significant 13 bits are used for the node-ID negotiation. The 13 bits are divided into two fields, the least significant 8 bits are used as data field, and the most significant 5 bits as data code. Table 1 illustrates the usage of the CAN-ID.

Table 1: 29-bit CAN-ID usage

Bit 28 to 13	Bit 12 to 8	Bit 7 to 0
reserved	Data code	Data field

Using 13 bit of the 29 bits of an extended CAN-ID means that 44 % of the bits are required, but the following

$$\frac{2^{13}}{2^{29}} = \frac{8.192}{536.870.912} = 0{,}00001525$$

formula shows that only 0,0015 % of all possible extended CAN-IDs are reserved for this service:

The data code defines the meaning of the data field content. If the most significant bit is set to 1, the data field contains parts of the CANopen object 1018_h, identity object. For this procedure to work, all four sub-indices are required. Table 2 shows which part of the identity object 1018_h is transmitted with which data code.

Table 2: Data code correlation with 1018_h

Data code	1018_h data
10000_b	Sub-index 1, bit 0 to 7
10001_b	Sub-index 1, bit 8 to 15
10010_b	Sub-index 1, bit 16 to 23
10011_b	Sub-index 1, bit 24 to 31
10100_b	Sub-index 2, bit 0 to 7
10101_b	Sub-index 2, bit 8 to 15
10110_b	Sub-index 2, bit 16 to 23
10111_b	Sub-index 2, bit 24 to 31
11000_b	Sub-index 3, bit 0 to 7
11001_b	Sub-index 3, bit 8 to 15
11010_b	Sub-index 3, bit 16 to 23
11011_b	Sub-index 3, bit 24 to 31
11100_b	Sub-index 4, bit 0 to 7
11101_b	Sub-index 4, bit 8 to 15
11110_b	Sub-index 4, bit 16 to 23
11111_b	Sub-index 4, bit 24 to 31

Due to the binary coding of the data code, it is very easy to implement the algorithm for accessing the value of the identity object. Bit 3 and 4 are the sub-index number - 1, and bit 0 and 1 are the byte number of this sub-index.

Two data code values are dedicated for the flow control of the process and are listed in Table 3.

Table 3: Flow control data code

Data code	Name
00001_b	ReqUsedNodeId
00010_b	ActUsedNodeId

The data code ReqUsedNodeId is used to start a new process, whereby the data field contains 0. The resulting message with extended CAN ID 100_h will trigger all devices with a valid node-ID to respond with ActUsedNodeId and their own node-ID in the data field. Figure 1 shows the basic concept of asking the network for the node-IDs, which are already used in the network.

Negotiation process

The negotiation process is only performed by nodes with an invalid node-ID. All nodes with a valid node-ID are ignoring the messages with the most significant bit in the data code field set to one. If a device participating in the procedure, receives a negotiation message, it compares the value according to Table 2 with its own equivalent. In case the received value is higher than its own value, the device continues the negotiation.

In case the received value is smaller as its own equivalent, the negotiation is stopped on this device. And, the device has to wait until the running negotiation is done. It can detect this by setting a timeout (e.g. 100 ms), and restart this timeout every time it receives a negotiation message. In case a device is able to transmit all 16 messages, which are needed to transmit all 128 bits of the identity object, it will then announce the preferred node-ID to the network by using ActUsedNodeId data code message.

Figure 2 shows a complete negotiation cycle with a timeout value of 100 ms before and after the actual negotiation and an additional 5-ms delay between every negotiation message.

Node-ID conflicts

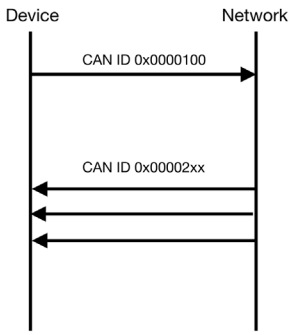

Figure 1: Initializing negotiation (Source: Emotas)

In case a device successfully negotiated a node-ID, but another device indicates by the ActUsedNodeId message the very same node-ID, the receiving device sends out the same ActUsedNodeId, sets its pending node-ID to the invalid node-ID, and enters the NMT state reset communication. Because the other device received this message, it loses its node-ID too. Both devices can restart the negotiation process again. The losing device of this negotiation shall alter the preferred node-ID for the next cycle to a node-ID not yet present in the system.

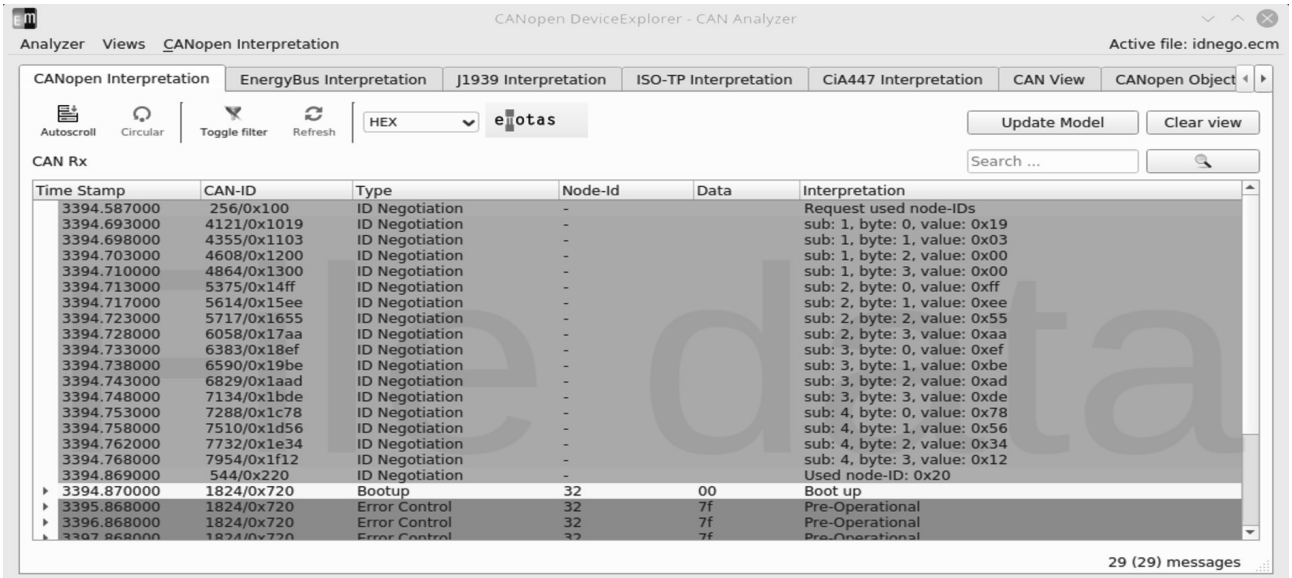

Figure 2: Complete negotiation cycle (Source: Emotas)

Table 4: CAN Trace simulation example (Source: Emotas)

Time (s)	CAN-ID	ide	l	d
0,000000	256_d / 100_h	EXT	0	–
0,105000	4121_d / 1019_h	EXT	0	–
0,106000	4121_d / 1019_h	EXT	0	–
0,110000	4355_d / 1103_h	EXT	0	–
0,111000	4355_d / 1103_h	EXT	0	–
0,115000	4608_d / 1200_h	EXT	0	–
0,115000	4608_d / 1200_h	EXT	0	–
0,119000	4864_d / 1300_h	EXT	0	–
0,121000	4864_d / 1300_h	EXT	0	–
0,125000	5375_d / $14FF_h$	EXT	0	–
0,126000	5375_d / $14FF_h$	EXT	0	–
0,130000	5614_d / $15EE_h$	EXT	0	–
0,130000	5614_d / $15EE_h$	EXT	0	–
0,135000	5717_d / 1655_h	EXT	0	–
0,136000	5717_d / 1655_h	EXT	0	–
0,140000	6058_d / $17AA_h$	EXT	0	–
0,141000	6058_d / $17AA_h$	EXT	0	–
0,145000	6383_d / $18EF_h$	EXT	0	–
0,146000	6383_d / $18EF_h$	EXT	0	–
0,149000	6590_d / $19BE_h$	EXT	0	–
0,155000	6829_d / $1AAD_h$	EXT	0	–
0,161000	7134_d / $1BDE_h$	EXT	0	–
0,165000	7201_d / $1C21_h$	EXT	0	–
0,170000	7491_d / $1D43_h$	EXT	0	–
0,174000	7781_d / $1E65_h$	EXT	0	–
0,181000	8071_d / $1F87_h$	EXT	0	–
0,279000	576_d / 240_h	EXT	0	–
0,280000	1856_d / 740_h	–	1	00_h
0,282000	256_d / 100_h	EXT	0	–
0,283000	576_d / 240_h	EXT	0	–
0,387000	4121_d / 1019_h	EXT	0	–
0,392000	4355_d / 1103_h	EXT	0	–
0,397000	4608_d / 1200_h	EXT	0	–
0,402000	4864_d / 1300_h	EXT	0	–
0,406000	5375_d / $14FF_h$	EXT	0	–
0,412000	5614_d / $15EE_h$	EXT	0	–
0,417000	5717_d / 1655_h	EXT	0	–
0,422000	6058_d / $17AA_h$	EXT	0	–
0,426000	6383_d / $18EF_h$	EXT	0	–
0,432000	6590_d / $19BE_h$	EXT	0	–
0,437000	6829_d / $1AAD_h$	EXT	0	–
0,442000	7134_d / $1BDE_h$	EXT	0	–
0,447000	7423_d / $1CFF_h$	EXT	0	–
0,451000	7491_d / $1D43_h$	EXT	0	–
0,456000	7781_d / $1E65_h$	EXT	0	–
0,462000	8071_d / $1F87_h$	EXT	0	–
0,561000	577_d / 241_h	EXT	0	–
0,562000	1857_d / 741_h	–	1	00_h
1,280000	1856_d / 740_h	–	1	$7F_h$
1,562000	1857_d / 741_h	–	1	$7F_h$

Table 4 shows a CAN Trace simulation example of two nodes negotiating their node-IDs. The difference in the 1018_h parameter revision number causes the one node to stop the negotiation and to restart again after the first one is ready.

The main goal is still to find a reliable, robust, and fast solution to assign node-IDs in plug-and-play systems to extend the network during its lifecycle. This theoretical approach should only be a basis for discussion on how to achieve this. With the discussed process it is possible to assign node-IDs to devices without the need of a dedicated LSS manager. The usage of extended CAN-IDs makes it possible that only 16 messages are needed, and that the process does not create CAN network collisions.

Author

Alexander Philipp
Emotas Embedded Communication
phi@emotas.de
www.emotas.de