Challenges of CANopen Node ID assignment, avoiding duplicates

Olaf Pfeiffer & Christian Keydel (Embedded Systems Academy)

The Node ID used by CANopen device is a very essential setting. Each node in a CANopen system requires one unique Node ID number by which it can be identified. If it is duplicated, errors will occur and potentially one or multiple nodes will shut down due to collisions on the network. This paper summarizes the typical options available for configuration of the Node ID and introduces the latest enhancement to LSS (Layer Setting Services), allowing for a fast dynamic assignment of Node IDs through the network.

When designing a CAN-based network, one condition has to be avoided at all cost: Devices having their CAN controllers going into BUS OFF state. This most severe of CAN errors logically disconnects the device from the bus and prevents all further communication with it. Recovery from this state requires at least a reset of the CAN controller but often a reset of the complete device. Sometimes, it requires power-cycling the whole network.

There are two major issues that can cause a CANopen device to go into BUS OFF state. The first is one or multiple devices operating at a different CAN speed, or bit rate, than the rest of the network. This invariably will cause some or all nodes that are trying to send messages eventually turning BUS OFF.

The second area of concern is on the logical, or protocol level. Each node connected to a CANopen network must have a unique Node ID between 1 and 127. If a CANopen Node ID exists multiple times in a network, in addition to the inability to distinguish between individual nodes, also physically different devices may try to send CAN messages at the same time that have the same identifier but different data. This thwarts the arbitration mechanism on a CAN bus, also causing devices to switch into BUS OFF state, and is therefore a big no-no.

This paper discusses the available options to address the second issue of Node ID assignment, which has to happen at some point during the development or integration of a CANopen network.

Hard Coded

If a CANopen network is embedded in a machine and all nodes run different software, then this software can use a hard coded Node ID. With power-up and initialization all nodes know their Node ID immediately and can start communicating. The benefit of this method is that the entire issue of assigning Node IDs is kept at the development level. The engineers originally planning the network make this assignment. Persons involved with installation or maintenance do not need to worry about it.

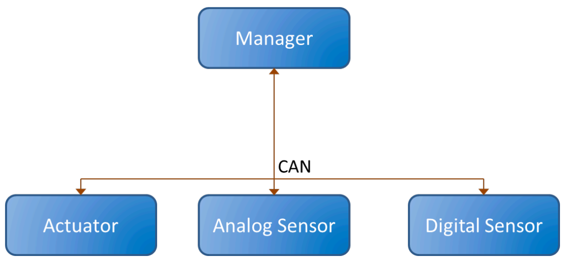

Figure 1: Network with all different devices

On the other hand this also means that the network is less flexible. If at some point a network device needs to be duplicated – the same device twice – then this does not work without a software change and the software would need to be different for both devices.

If several nodes in a system are based on the same hardware/software, then an additional way to set the Node ID is required as otherwise these nodes would automatically have the same Node ID.

Dials or Switches

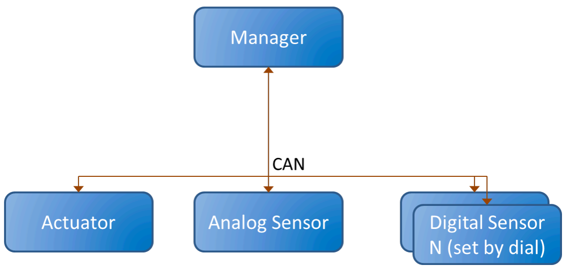

Figure 2: Multiple same nodes in a network with dials

Some CANopen modules and devices have DIP switches or dials to select a Node ID. This means that the person installing or replacing a device in a network must have the knowledge about these settings and how to make them before inserting or adding a module to the network. Although this is a very flexible solution, dials and switches are becoming less popular. Every configuration switch added increases the risk that a technician does something wrong.

Coded into the cabling and connectors

If the connector for the CAN cabling has extra pins available, those can be equipped with different bridges / shorts representing a binary number. The software in the connected module can then read this number and use it to set its own Node ID.

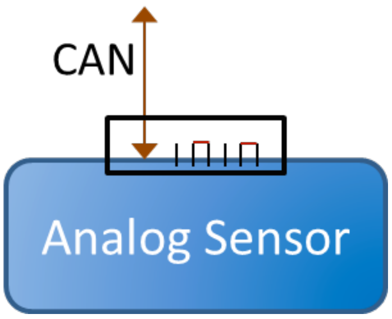

Figure 3: Bridges in the connector

This method moves the intelligence about the Node IDs from the modules to the wiring of the system. As with the hard coded software solution, the Node ID assignment happens at the engineering level. If the cables and connectors all come pre-configured, the installation or maintenance technicians do not need to worry about it.
This solution is typically only used in networks with a limited number of nodes as otherwise the connectors become too big or too expensive.

Stored in non-volatile memory such as EEPROM

Most embedded devices today have some non-volatile memory in which configuration data can be stored. The Node ID setting can be made part of that configuration data. However, this in itself is not a solution to the main problem: When does who decide which node number is assigned to which module? Are the Node IDs pre-set individually during the manufacturing process? Or does the person installing a system need to connect some configuration tool and set them from there?

Dynamically at run-time

Recent improvements in the CANopen Layer Setting Services (LSS) have made the Node ID assignment through the network more efficient and faster. These methods allow an LSS Master to detect unconfigured nodes and assign them a Node ID dynamically. In general, this method also supports plug-and-play at runtime. An unconfigured node can be added at any time, will be detected by the LSS Master and assigned a Node ID.
Early implementations were slow and sometimes required up to a minute for the reliable detection of an unknown, unconfigured node and the assignment of a Node ID to it. With the latest enhancements, called LSS Fastscan with feedback message, which improves on the already enhanced LSS Fastscan [1], this time can now be reduced to less than 100 milliseconds.

In order for this method to work, the participating nodes must implement all Object Dictionary entries of the Identification object 1018h. These are four values of 32bits each: the vendor ID – which is assigned by CAN in Automation, a product code, a revision number and a serial number. Together, these 128bits make up the LSS ID and are used to specifically identify a particular device on the network. Therefore this number must be unique to each node.

Principles of the LSS Fastscan cycle

LSS Fastscan only requires two CAN messages: the LSS Master message (from master to devices) and the LSS Identify message (response from devices to master). The LSS Identify is the same for all devices (same contents), so if multiple devices send it, they do not cause a collision on the network, it can be transmitted from multiple devices at the same time.
The basic mode of operation is a bit-by-bit scan of the 128bit LSS ID. The LSS Master first asks “is there anybody non-configured / in LSS mode?” If one or multiple LSS Identify responses come, this is true. Next the master asks “is there anybody with the highest bit of the LSS ID cleared?” If the LSS Identify response comes, this is true. The master can continue with the next request for the next bit: “is there anybody with the highest TWO bits of the LSS ID cleared?” If no LSS Identify response comes within a defined timeout, the Master assumes a bit to be set, otherwise to be cleared.
It is important that LSS devices that no longer “have a match” of their own LSS ID with the bit pattern requested by the LSS Master messages remain silent. Only LSS devices that have all requested bits matching may still participate in sending LSS Identify responses.
It takes about 128 requests to determine the entire LSS ID, there might be a four extra requests for confirmation of a complete 32 bit value. With a timeout of 10ms, it takes about 1.25 seconds to detect all 128bits of a LSS ID.

Parameters of the LSS Fastscan Identify Master message

As defined by [1] and [2], the eight bytes (0 to 7) of the LSS Master Message (0x7E5) are used as follows with the Fastscan service:
Byte 0, command byte:
Set to 81h, identifies this as a Fastscan Identify message.
Bytes 1-4, IDNumber:
These are 32bits each that are checked versus the Vendor ID, product code, revision number or serial number.
Byte 5, BitChecked:
Defines how many of the bits in IDNumber are currently checked. This is a value in the range of 0 to 31. 31 means that only bit 31 is checked, 30 means that bits 31 and 30 are checked, 29 means that bits 31, 30 and 29 are checked and so on. 0 means that all 32bits are checked.
A value of 80h is an exception and indicates the start of a new scan cycle, all nodes supporting Fastscan reset their internal state machines and respond.
Byte 6, LSSSub:
Defines which part of the 128bit LSS ID is currently checked in the 32bit IDNumber. This is a value from 0 to 3 representing the Vendor ID, product code, revision number or serial number.
Byte 7, LSSNext:
Defines which part of the 128bit LSS ID will be checked towards the 32bit IDNumber in the next cycle. This is a value from 0 to 3 standing for the Vendor ID, product code, revision number or serial number.

Introducing feedback messages

Using these parameters, an LSS Master can determine the LSS ID of an individual node by checking it bit-by-bit as shown in Table 1. However, these parameters can also be used to verify known bits of an LSS ID. So if the LSS Master knows the next group of bits, it can confirm them using just one message.
Instead of only using the single identify message, LSS devices supporting feedback also use feedback messages to

inform the LSS master about the next, upcoming bits of their LSS ID. These are coded into the CAN identifier, so that feedback messages from multiple nodes do not cause a collision on the network. In the example of table [2] 29bit ID messages are used with the lowest 16bit representing the next 16bits of the CAN ID. An alternate method uses 16 different 11bit messages as feedback supplying the LSS master with the next 4bits of a slave's LSS ID. The version with 29bit messages should only be used, if the number of nodes using this method is limited as it causes a higher busload then the single, shared LSS Identify message. Today the 29bit feedback version is in use in networks with up to 16 nodes.

Usage Considerations

Although the dynamic Node ID assignment with LSS Fastscan and feedback might now sound like a perfect solution for many applications, there is still one thing to consider: this method does not provide any clues about the physical location of a device in the network. As an example consider a machine using two identical CANopen sensors for the right side and the left side of a machine. With LSS each sensor is assigned its own, unique Node ID. However, from this information alone we do not know which one is left or right?

In many applications there are mechanisms that allow detecting the physical location; maybe because the two devices have a different configuration (which can be read via CANopen) or by the way the machine is built, for example the right sensor always becoming active before the left.

If such additional information is not available, then some manual configuration is still required. In order to keep the process manageable for an installation or maintenance crew, the recommendation is to only have them add/install one node at the time. The system detects it and can inform the crew "new device detected from manufacturer X, serial number Y" which can then be assigned to be the "left" or "right" sensor. If that information is persistently stored in the Master/Manager, then from that time forward the system knows the physical location of a device with a given serial number.

Conclusion

All Node ID assignment methods mentioned above are in use today, also in combination, wherever for a specific application they are simply the best fit. With the new LSS detection and configuration method being vastly more efficient than previous iterations, a dynamical assignment of Node IDs at power up of the network is becoming more attractive for many applications for which it was previously not considered due to the long detection cycle times.

Table 1: LSS Fastscan cycle without feedback

Msg	ID	Details	Raw Message
1	0x7E5	LSS Fastscan: Checking for new nodes	51 00 00 00 00 80 00 00
2	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
3	0x7E5	LSS Fastscan: Vendor ID - Checking bits 31 - 31	51 00 00 00 00 1F 00 00
4	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
5	0x7E5	LSS Fastscan: Vendor ID - Checking bits 30 - 31	51 00 00 00 00 1E 00 00
6	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
. . .	*continue*		
56	0x7E5	LSS Fastscan: Vendor ID - Checking bits 0 - 31	51 41 53 35 01 00 00 01
57	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
. . .	*continue*		
115	0x7E5	LSS Fastscan: Product Code - Checking bits 0 - 31	51 03 00 47 04 00 01 02
116	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
. . .	*continue*		
176	0x7E5	LSS Fastscan: Revision Nr. - Checking bits 0 - 31	51 06 00 02 00 00 02 03
177	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
178	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 31 - 31	51 00 00 00 00 1F 03 03
179	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 30 - 31	51 00 00 00 80 1E 03 03
. . .	*continue*		
204	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
205	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 9 - 31	51 00 F0 ED FE 09 03 03
206	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
207	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 8 - 31	51 00 F0 ED FE 08 03 03
208	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
209	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 7 - 31	51 00 F0 ED FE 07 03 03
210	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
211	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 6 - 31	51 00 F0 ED FE 06 03 03
212	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
213	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 5 - 31	51 00 F0 ED FE 05 03 03
214	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
215	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 4 - 31	51 00 F0 ED FE 04 03 03
216	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
217	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 3 - 31	51 00 F0 ED FE 03 03 03
218	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 2 - 31	51 08 F0 ED FE 02 03 03
219	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 1 - 31	51 0C F0 ED FE 01 03 03
220	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
221	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 0 - 31	51 0C F0 ED FE 00 03 04
222	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 0 - 31	51 0D F0 ED FE 00 03 04
223	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
224	0x7E5	Configure Node ID - NID: 0x03	11 03 00 00 00 00 00 00
225	0x7E4	Configure Node ID - Success	11 00 00 00 00 00 00 00
226	0x7E5	Store Configuration	17 00 00 00 00 00 00 00
227	0x7E4	Store Configuration - Success	17 00 00 00 00 00 00 00
228	0x7E5	Switch Mode Global - Operation Mode	04 00 00 00 00 00 00 00
229	0x703	Boot up of node 3	00

The trace recording above is a summary of a LSS Fastscan execution. The columns show the CAN message identifier seen on the network (7E5h for LSS Master message and 7E4h for LSS Identify response), the message interpretation with type and details and the raw message contents. At message number 56, 115, 176 and 222 the next 32bits of the LSS ID are detected, as highlighted in the "Raw Message" column. Based on a 20ms timeout this took about 2.78 seconds to execute (until boot up).

Table 2: LSS Fastscan cycle with feedback

Msg	ID	Details	Raw Message
1	0x7E5	LSS Fastscan: Checking for new nodes	51 00 00 00 00 80 00 00
2	0x1F900135	LSS Fastscan: Bits 16 - 31 are 0x0135	
3	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
4	0x7E5	LSS Fastscan: Vendor ID - Checking bits 16 - 31	51 00 00 35 01 10 00 00
5	0x1F915341	LSS Fastscan: Bits 0 - 15 are 0x5341	
6	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
7	0x7E5	LSS Fastscan: Vendor ID - Checking bits 0 - 31	51 41 53 35 01 00 00 01
8	0x1F900447	LSS Fastscan: Bits 16 - 31 are 0x0447	
9	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
10	0x7E5	LSS Fastscan: Product Code - Checking bits 16 - 31	51 00 00 47 04 10 01 01
11	0x1F910003	LSS Fastscan: Bits 0 - 15 are 0x0003	
12	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
13	0x7E5	LSS Fastscan: Product Code - Checking bits 0 - 31	51 03 00 47 04 00 01 02
14	0x1F900002	LSS Fastscan: Bits 16 - 31 are 0x0002	
15	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
16	0x7E5	LSS Fastscan: Revision Nr. - Checking bits 16 - 31	51 00 00 02 00 10 02 02
17	0x1F910006	LSS Fastscan: Bits 0 - 15 are 0x0006	
18	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
19	0x7E5	LSS Fastscan: Revision Nr. - Checking bits 0 - 31	51 06 00 02 00 00 02 03
20	0x1F90FEED	LSS Fastscan: Bits 16 - 31 are 0xFEED	
21	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
22	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 16 - 31	51 00 00 ED FE 10 03 03
23	0x1F91F00D	LSS Fastscan: Bits 0 - 15 are 0xF00D	
24	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
25	0x7E5	LSS Fastscan: Serial Nr. - Checking bits 0 - 31	51 0D F0 ED FE 00 03 04
26	0x7E4	Identify Servant	4F 00 00 00 00 00 00 00
27	0x7E5	Configure Node ID - NID: 0x03	11 03 00 00 00 00 00 00
28	0x7E4	Configure Node ID - Success	11 00 00 00 00 00 00 00
29	0x7E5	Store Configuration	17 00 00 00 00 00 00 00
30	0x7E4	Store Configuration - Success	17 00 00 00 00 00 00 00
31	0x7E5	Switch Mode Global - Operation Mode	04 00 00 00 00 00 00 00
32	0x703	Boot up of node 3	00

The trace recording above shows all messages involved in the scan of a single LSS device using 29bit feedback messages. The 29bit feedback messages reports the feedback value in bits 0 to 15. Bit 16 is a toggle bit alternating with each transfer. Bit 17 is reserved. Bits 18 to 28 contain the pattern 7E4h so that it matches with the 11bit ID of the LSS Identify message. Based on a 20 millisecond timeout this took 210ms to execute (until boot up).

Olaf Pfeiffer
Embedded Systems Academy GmbH
Bahnhofstr. 17
D-30890 Barsinghausen
Tel.: (+49) 5105 / 582 7897
opfeiffer@esacademy.com
www.esacademy.com

Christian Keydel
Embedded Systems Academy GmbH
Bahnhofstr. 17
D-30890 Barsinghausen
Tel.: (+49) 5105 / 582 7897
opfeiffer@esacademy.com
www.esacademy.com

References

[1] CiA 305-1 Work Draft version 2.2.14 from 26 January 2012, chapter 6.7.3 – LSS Fastscan procedure
[2] icc 2008, Pfeiffer, Plug and Play: Node detection and Node ID assignment with the LSS Fast Scan service