MicroControl
Systemhaus für Automatisierung GmbH & Co. KG

TD-03011E

Identifier Usage in CANopen Networks

Author **: U. Koppe**

Published **: January 2003**

This document describes the usage of identifiers in CANopen networks. The paper is intended for CANopen newcomers and system integrators.

In order to understand the various abbrevations for CANopen services, a short introduction is given. More details can be found in the corresponding specification DS-301.

The default identifier setup is explained and examples are given on how to change the identifier values for various services.

Introduction into CANopen Services

NMT	The „**Network Management** (NMT)“ is node oriented and follows a master-slave-structure. The service is used to start, stop, reset and initialize a CANopen node. Through the important nature of the service the highest identifier priority is used. The NMT message has two bytes of data.
SYNC	The „**Synchronisation Object** (SYNC)“ is broadcasted periodically by the SYNC producer (one CANopen node in a network). The SYNC provides the basic network clock. In order to give the SYNC message a short access time to the CAN bus a very high priority identifier is used by default. The SYNC message has no data.
TIME	By means of the „**Time Stamp Object** (TIME)“ a common time frame reference is provided to all devices in the CANopen network. The data length of the TIME message is fixed to six bytes.
EMCY	A „**Emergency Object** (EMCY)“ is triggered by the occurence of a device internal error situation. The data length of the EMCY message is fixed to eight bytes.
PDO	Real-time data transfer is performed by means of „**Process Data Objects** (PDO)“. The transfer of PDOs is performed with no protocol overhead. The data length of a PDO message is variable.
SDO	With „**Service Data Objects** (SDO)“ the access to a device object dictionary is provided. By means of a SDO a peer-to-peer communication channel between two devices is established. The data length of a SDO message is fixed to eight bytes.
NMT-EC	Through „**Network Management Error Control** (NMT-EC)“ failures in a CANopen network can be detected (change of node state). The data length of the two possible message types (Heartbeat / Nodeguarding) is fixed to one byte.
LSS	With „**Layer Setting Services** (LSS)“ it is possibe to change the node ID and the baudrate of a CANopen device. The LSS is described in the specification DSP-305.

Default Identifier Setup

The default identifier setup is described as „Pre-defined Connction Set“ inside the communication profile DS-301. Every CANopen node can be reset to this behaviour with a specific command (Restore Default Parameters, Object 1011_h).

Identifier	Service	Direction	COB-ID calculation	Remarks
000_h	NMT	Receive	n/a	mandatory service
080_h	SYNC	Recv./Trmt.	n/a	change via index 1005_h
081h - 0FFh	EMCY	Transmit	COB-ID = 080_h + Node ID	change via index 1014_h
100_h	TIME	Recv./Trmt.	n/a	change via index 1012_h
181_h - $1FF_h$	PDO1	Transmit	COB-ID = 180_h + Node ID	change via index 1800_h
201_h - $27F_h$	PDO1	Receive	COB-ID = 200_h + Node ID	change via index 1400_h
281_h - $2FF_h$	PDO2	Transmit	COB-ID = 280_h + Node ID	change via index 1801_h
301_h - $37F_h$	PDO2	Receive	COB-ID = 300_h + Node ID	change via index 1401_h
381_h - $3FF_h$	PDO3	Transmit	COB-ID = 380_h + Node ID	change via index 1802_h
401_h - $47F_h$	PDO3	Receive	COB-ID = 400_h + Node ID	change via index 1402_h
481_h - $4FF_h$	PDO4	Transmit	COB-ID = 480_h + Node ID	change via index 1803_h
501_h - $57F_h$	PDO4	Receive	COB-ID = 500_h + Node ID	change via index 1403_h
581_h - $5FF_h$	SDO	Transmit	COB-ID = 580_h + Node ID	mandatory service
601_h - $67F_h$	SDO	Receive	COB-ID = 600_h + Node ID	mandatory service
701_h - $77F_h$	NMT-EC	Transmit	COB-ID = 680_h + Node ID	mandatory service
$7E4_h$ - $7E5_h$	LSS	Recv./Trmt.	n/a	identifiers are fixed

Table 1: Pre-defined Connection Set

The direction in the table is given with respect to the CANopen slave device. The node ID (address) of a CANopen device must be within the range from 1 to 127. The communication object identifier (COB-ID) is directly dependent on the selected node ID.

The services in the highlighted lines (NMT, SDO, NMT-EC) are mandatory services, which are implemented in every CANopen device. The COB-ID of these services can not be changed.

Also the COB-IDs for the Layer Setting Services (LSS) can not be changed. The LSS service is not mandatory for a CANopen device.

Within the pre-defined connection set, the following identifiers are not used:

COB-ID (Identifier values)
001_h - $07F_h$
101_h - 180_h
200_h / 280_h / 300_h / 380_h / 400_h / 480_h / 500_h / 580_h
600_h / 680_h
780_h - $7E3_h$
$7E6_h$ - $7FF_h$

Table 2: Unused identifiers in the Pre-defined Connection Set

Using CANopen devices in Non-CANopen networks

Is it possible to use a CANopen device in a CAN network that runs with another protocol? The answer to that question is: Yes, unless you keep two limitations in mind:

1. The Non-CANopen network shall not use the identifier value 0 (CANopen NMT).

2. The Non-CANopen network shall not use the identifiers for SDO and NMT-EC services. This can be achieved by changing the CANopen device address in most cases.

But how can the CANopen device be set into Operational state in a Non-CANopen network? Simply by configuration of the object $1F80_h$ (NMT Startup). Writing and storing a value of 0x0000 $00C0_h$ to this object will tell the CANopen device to enter automatically „Operational“ state.

Example: Automotive Test Application

A car manufacturer has to acquire many additional temperatures in a new vehicle during the test runs (e.g. brakes, clutch, etc). In most cases it makes sense to feed this information into the existing CAN network, which knows nothing about CANopen. Let us assume the car manufacturer wants to acquire 4 temperatures, where the data is transmitted with the identifier value 312_h periodically. A standard

CANopen device for temperature acquisition can be used for that automotive application if the above mentioned limitations are met.
Before connecting the CANopen temperature module to the CAN network of the vehicle, the following configuration must be applied to the CANopen device:

- Configure the Transmit PDO identifier to the value 321_h
- Configure the Transmit PDO to cyclic transmission, using the desired update rate
- Configure the NMT Startup object to „auto-start“
- Store this configuration

Figure 1: CANopen device for temperature measurement

It is, however, nearly impossible to use CANopen devices in a Non-CANopen network when this network requires a certain protocol (Higher-Layer Protocol). It is, for example, not possible to use a CANopen module in a DeviceNet network.

Changing Identifiers

With the exception of the NMT service, the SDO and the NMT-EC service, identifiers for all other CANopen services can be changed.

The NMT service has always the identifier value 0. It can not be modified by any means. The identifiers for the SDO service and the NMT-EC service are always bound to the node ID of a CANopen slave device (refer to “Pre-defined Connection Set” on page 3.).

The identifier value of a CANopen service is changed by a communication via SDO (peer-to-peer). Details concerning the SDO communication can be found in the DS-301 [1].

An „Expedited SDO“ message has the following structure:

ID	*DLC*	*B0*	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*
	8	CD	Index		SI	Data			

Byte 0 of the message (**CD** = command byte) has the following contents for a 32-bit read or write operation:

SDO Operation by Master	Value
Master reads from Slave	40_h
Master writes to Slave	23_h

Table 3: Command byte values for Expedited SDO

The addressed CANopen slave answers with the identical message structure, where the command byte may have the following values:

SDO Answer from Slave	Value
Successful read response	43_h
Succesfull write response	60_h
Abort operation	80_h

Table 4: Command byte response for Expedited SDO

Bytes 1 and 2 define the index for the parameter, byte 3 defines the sub-index for the parameter.

The byte order for the fields "**Index**" and "**Data**" is least significant byte first (Intel format).

Example: Changing the SYNC identifier

In the following example the identifier value for the SYNC service is changed to 120_h. The CANopen node ID is 2. For calculation of the SDO identifiers refer to “Pre-defined Connection Set” on page 3. The index for the SYNC identifier is 1005_h. This leads to the following SDO write operation:

ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
602_h	8	23_h	05_h	10_h	00_h	20_h	01_h	00_h	00_h

As response the requested CANopen slave will send:

ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
582_h	8	60_h	05_h	10_h	00_h	00_h	00_h	00_h	00_h

Now the CANopen slave will receive SYNC messages with the identifier value 120_h.

Usage of Extended-CAN Frames

Extended-CAN frames (CAN 2.0B) are possible for the following services in a CANopen network:

Service	Default Identifier
SYNC	080_h
EMCY	081_h - $0FF_h$
TIME	100_h
PDO	refer to “Pre-defined Connection Set” on page 3.

Table 5: Services with the possibility of using 29-bit identifiers

Please note that allthough you can use 29-bit identifiers for the PDO service, the total number of PDOs inside a CANopen network is limited to 512. The limitation is caused by the CANopen object dictionary, which allows only up to 512 entries for each PDO direction (receive / transmit).

In order to enable Extended-Frame operation, bit 29 of the COB-ID parameter must be set to 1.

Example: Changing the EMCY identifier (29-Bit)

In the following example the identifier value for the EMCY service is changed to $7F1234_h$. The CANopen node ID is 2. For calculation of the SDO identifiers refer to “Pre-defined Connection Set” on page 3. The index for the EMCY identifier is 1014_h. This leads to the following SDO write operation:

ID	*DLC*	*B0*	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*
602_h	8	23_h	14_h	10_h	00_h	34_h	12_h	$7F_h$	20_h

As response the requested CANopen slave will send:

ID	*DLC*	*B0*	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*
582_h	8	60_h	14_h	10_h	00_h	00_h	00_h	00_h	00_h

Now the CANopen slave will transmit emergency messages with the identifier value $7F1234_h$.

References

[1] CANopen Application Layer and Communication Profile, DS-301, Version 4.02, CAN in Automation, Erlangen/Germany

[2] CANopen Framework for Programmable CANopen Devices, DSP-302, Version 3.1, CAN in Automation, Erlangen/Germany