CANopen Application Note

Introduction

If you setup a CANopen network, maybe you come to the point where you face a difficulty:
How to find the right command to setup the device?
And how does the structure of the command have to look?

This application note is an appendix to the Kübler CANopen encoder manuals and can hopefully explain the questions and support you find the right way to setup your encoder.

Table of content:

Stefan Schubert
Service & Applikations Center 26.05.2009

SDO Access

The SDOs allow direct access to the object dictionary of the CANopen device. This access is simple and well arranged. An SDO access always starts from the superimposed control (host), which sends to the CANopen device either a write instruction in order to change a parameter of the object dictionary, or a read instruction in order to read out a parameter. The host receives an answer for each instruction. This answer either contains the readout value or – in the case of a write instruction – serves as a confirmation. The identification of the message for the CANopen device is effected by means of the COB-ID. The structure of the instructions or the answers depends on the data type of the object to be read or written, because 1, 2, or 4 data bytes have to be sent or received.

Write Access, Data Transfer from Host to Slave
Each access to the object dictionary is checked by the slave for validity. Any write access to non-existing objects, to read-only objects or with a non-corresponding data format are rejected and answered with a corresponding error message.

The host sends:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD	Index LSB	MSB	Sub-Index	data LSB			…MSB

CMD determines the direction of data transfer and the size of the data object:
23 hex Sending of 4-byte data (bytes 4...7 contain a 32-bit value)
2B hex Sending of 2-byte data (bytes 4, 5 contain a 16-bit value)
2F hex Sending of 1-byte data (byte 4 contains an 8-bit value)

Index 16-bit value; index of the object to be written into (in the object dictionary)
Sub-Index 8-bit value; sub-index of the object to be written into (in the object dictionary)
Data 8-bit, 16-bit, or 32-bit value

The slave answers:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
RES	Index LSB	MSB	Sub-Index	reserved			

RES Response of the slave:
60 hex Data sent successfully
80 hex Error, bytes 4...7 contain the error code according to the standard

Index 16-bit value; index of the object addressed by the host telegram
Sub-Index 8-bit value; sub-index of the object addressed by the host telegram
Reserved Is not used, or error message (depending on RES)

Example:
Writing into the operating parameter (Index 6000, Sub-Index 00) value = 1h (count direction=ccw)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2B	00	60	00	01	00	00	00
Response:	60	00	60	00	00	00	00	00

Stefan Schubert
Service & Applikations Center

Read Access, Data Transfer from Slave to Host
Any read access to non-existing objects is answered with an error message.

The host sends:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD	Index LSB	MSB	Sub-Index	reserved			

CMD determines the direction of data transfer:
40 hex read access (in any case)

Index 16-bit value; index of the object to be read (in the object dictionary)
Sub-Index 8-bit value; sub-index of the object to be read (in the object dictionary)
Reserved Is not used

The slave answers:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
RES	Index LSB	MSB	Sub-Index	data LSB			…MSB

RES Response of the slave:
43 hex Bytes 4...7 contain a 32-bit value
4B hex Bytes 4, 5 contain a 16-bit value
4F hex Byte 4 contains an 8-bit value
80 hex Error, bytes 4...7 contain the error code according to the standard

Index 16-bit value; index of the object addressed by the host telegram
Sub-Index 8-bit value; sub-index of the object addressed by the host telegram
Data Data or error message (depending on RES)

Example:
Reading of the status word (Index 6000, Sub-Index 00)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	40	00	60	00	00	00	00	00
Response:	4B	00	60	00	01	00	00	00

Abort code Description

0503 0000h	Toggle bit not alternated.
0504 0000h	SDO protocol timed out.
0504 0001h	Client/server command specifier not valid or unknown.
0504 0002h	Invalid block size (block mode only).
0504 0003h	Invalid sequence number (block mode only).
0504 0004h	CRC error (block mode only).
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	Object does not exist in the object dictionary.
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to a hardware error.
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value.
0800 0000h	general error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).

Network Management – NMT Service

All CANopen devices can be controlled by means of the network management. A special identifier (000h) is reserved for this purpose. It is used to send instructions to one or all servo amplifiers. Each instruction is composed of two bytes - the first byte includes the instruction code and the second byte the node address of the operated CANopen device.

Structure of the message of the superimposed control:

Identifier	Byte 0	Byte 1
000h	Instruction code	Node address

With the node address zero, all nodes in the network can be activated at the same time. The CANopen devices do not confirm the NMT instructions.

The „pre-operational" status is provided for the configuration phase of the network (SDO). For the work with PDOs, the node must have accepted the communication status „**operational“**. Generally, the network is booted by the host and put in the "operational" status by means of the „NMT-Start“ instruction.

Code	Name	Mode
01	Start Remote Node	Operational mode
02	Stop Remote Node	Pre-operational mode
80	Enter Pre-Operational	Pre-operational mode
81	Reset Node	
82	Reset Communication	

Example:
Start node number 63 (0x3f) (switch to operational mode)

Identifier	Byte 0	Byte 1
000h	Instruction code	Node address
000h	01	3F

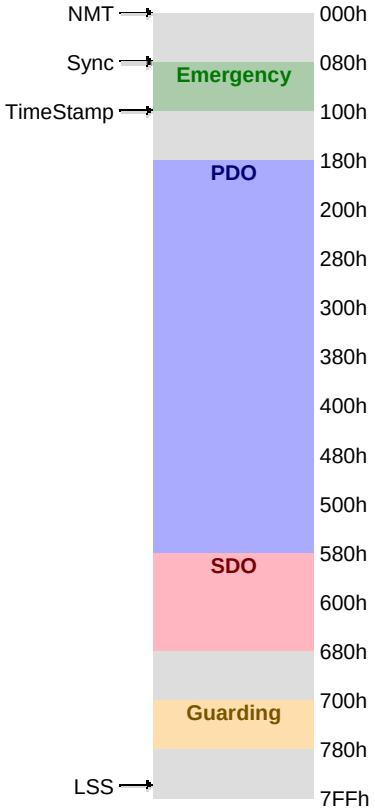
COB-ID

In order to reduce the amount of configuration required with simple network structures (1: n communication relationships between a control device and several lower order devices), CANopen supports a predefined allocation of message identifiers (Predefined Connection Set). This set of predefined identifiers supports one emergency message per node, synchronization and time stamp messages, one SDO-connection per device, the NMT-messages for node control and node monitoring and up to 4 transmit and 4 receive PDOs per device.

In a CANopen network it is possible to differentiate between a maximum of 127 nodes. These nodes share the 11-bit identifier space.

First a differentiation is made between network- and device-related functions. One CAN identifier is reserved for each of the network-related functions (e.g. NMT node control), one identifier per device is required for each device-related functionality (e.g. emergency messages, PDOs), since it must be possible to differentiate between the same functions on different devices. More important functions are assigned a higher priority COB-ID. For future extensions and for historical reasons, some message identifiers are not assigned. It is thus possible to operate systems with a higher order control node and up to 127 slave nodes without reconfiguration.

The following diagram shows the resulting partitioning of the CAN identifier space and the priority assigned to each COB-ID:



Stefan Schubert
Service & Applikations Center

The following table shows the identifier allocation of the Predefined Connection Set:

Communication object	COB-ID(s) hex	Slave nodes
NMT node control	000	Receive only
Sync	080	Receive only
Emergency	080 + NodeID	Transmit
TimeStamp	100	Receive only
PDO	180 + NodeID 200 + NodeID 280 + NodeID 300 + NodeID 380 + NodeID 400 + NodeID 480 + NodeID 500 + NodeID	1. Transmit PDO 1. Receive PDO 2. Transmit PDO 2. Receive PDO 3. Transmit PDO 3. Receive PDO 4. Transmit PDO 4. Receive PDO
SDO	580 + NodeID 600 + NodeID	Transmit Receive
NMT node monitoring (node guarding/heartbeat)	700 + NodeID	Transmit
LSS	7E4 7E5	Transmit Receive

SDOs and PDOs are always used in pairs (i.e. to transmit and to receive), where the rule is that the node on the lower (and therefore higher priority) COB-ID transmits and on the higher (i.e. lower priority) COB-ID receives.

Object 6000h Operating Parameters:

This object shall contain the functions for Code sequence, Commissioning diagnostic control and Scaling function control and also manufacture specific parameters

Bit 0: Code sequence: 0 = increasing when turning clockwise (cw)
1 = increasing when turning counter-clockwise (ccw)
Default: Bit = 0

Bit 2: Commissioning: Diagnostic control: 0 = disable
1 = enable
Default: Bit = 0

Bit 2: Scaling Function: 0 = disable
1 = enable
Default: Bit = 1

Bit13: Speed Format: 0 = RPM
1 = Units /second
Default Bit = 0

Bit14: Startup Mode: 0 = after Boot up Pre-Operational,
1 = after Boot up Operational mode
Default Bit = 0

Bit15: Event Mode: 0 = Position output acc. to TPDO 1800h
1 = Output on each change of position
Default Bit = 0

Bit	Function	Bit = 0	Bit = 1	Bit numbering
0	Code sequence	**CW**	CCW	0x0001
1	Commissioning Diagnostic Control	**Disabled**	Enabled	0x0002
2	Enable scaling	Disabled	**Enabled**	0x0004
3	Measuring direction	Forward	Reverse	0x0008
4…11	Reserved for further use			
12	Manufacturer specific parameter	Not used	Not used	
13	Speed Format	**RPM**	Units/sec	0x2000
14	Startup automatic in OP-Mode	**Disabled**	Enabled	0x4000
15	Event Mode Position	**Disabled**	Enabled	0x8000

Bold=default

Example:
You want make the following functions:

Code sequence increasing when turning counter-clockwise	0x0001
Scaling function enabled	0x0004
After Boot up operational mode	0x4000

	0x4005

The summery of the bit numbering (here 0x4005) is the data word you have to write into the object 6000

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2B	00	60	00	05	40	00	00
Response:	60	00	60	00	00	00	00	00

Programming Examples:

Index 1010
Save all bus objects with index 1010 sub-index 01

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	10	10	01	73	61	76	65
Response:	60	10	10	01	00	00	00	00

Index 1011
Load factory default with index 1011 sub-index 01

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	11	10	01	6C	6F	61	64
Response:	60	11	10	01	00	00	00	00

Index 180x
Set the transmission mode to event mode (to 0xFF) with index 1800 sub-index 02

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	02	18	02	FF	00	00	00
Response:	60	02	18	02	00	00	00	00

Set the event timer to 500ms (0x1F4) with index 1800 sub-index 05 to

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2B	02	18	05	F4	01	00	00
Response:	60	02	18	05	00	00	00	00

Index 2100
Set baud rate (to 0x05) with index 2100 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	00	21	00	05	00	00	00
Response:	60	00	21	00	00	00	00	00

0 = 10 kBit/s; 1 = 20 kBit/s; 2 = 50 kBit/s; 4 = 125 kBit/s; 5 = 250 kBit/s; 6 = 500 kBit/s; 8 = 1000 kBit/s

Index 2101
Set node address (to 0x3F) with index 2101 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	01	21	00	3F	00	00	00
Response:	60	01	21	00	00	00	00	00

Index 2102
Switch termination resistor off (to 0x00) with index 2102 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	02	21	00	00	00	00	00
Response:	60	02	21	00	00	00	00	00

termination on = 1; termination off = 0

Index 2105
Save all bus objects with index 2105 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	05	21	00	73	61	76	65
Response:	60	05	21	00	00	00	00	00

Index 6000
set ccw counting direction (to 0x01) with index 6000 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2B	00	60	00	01	00	00	00
Response:	60	00	60	00	00	00	00	00

Index 6001
Set MUR with (to 0x1000) index 6001 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	01	60	00	00	10	00	00
Response:	60	01	60	00	00	00	00	00

Index 6002
Set TMR (to 0x10000) with index 6002 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	02	60	00	00	00	01	00
Response:	60	02	60	00	00	00	00	00

Index 6003
Set preset value (to 0x00) with index 6003 sub-index 00 to 0x00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	03	60	00	00	00	00	00
Response:	60	03	60	00	00	00	00	00

Index 6200
Set event timer (to 0xa) with index 6200 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2B	00	62	00	0A	00	00	00
Response:	60	00	62	00	00	00	00	00

Example how to map the position, speed and acclaration in the transmision PDO3

Index 1A02

Set the numer of entries to 3 (3 objects can be mapped) with index 1A02 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	02	1A	00	03	00	00	00
Response:	60	02	1A	00	00	00	00	00

Open array with index 1A02 sub-index 00

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	02	1A	00	00	00	00	00
Response:	60	02	1A	00	00	00	00	00

Mapping of object 6004 sub 0 with 32 bit data legth to mapping object 1 with index 1A02 sub-index 01

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	02	1A	01	20	00	04	60
Response:	60	02	1A	01	00	00	00	00

Mapping of object 6030 sub 1 with 16 bit data legth to mapping object 2 with index 1A02 sub-index 02

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	02	1A	02	10	01	30	60
Response:	60	02	1A	02	00	00	00	00

Mapping of object 6040 sub 1 with 16 bit data legth to mapping object 3 with index 1A02 sub-index 03

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	02	1A	03	10	01	40	60
Response:	60	02	1A	03	00	00	00	00

Close array with index 1A02 sub-index 00 to 03

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2F	02	1A	00	03	00	00	00
Response:	60	02	1A	00	00	00	00	00

Example how to change the COB-ID from standart (at PDO1 180 + node-ID) to 0x40B

Index 1800

Set current COB-ID invalid with index 1800 sub-index 01

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	00	18	01	8B	01	00	80
Response:	60	00	18	01	00	00	00	00

Write new COB-ID with index 1800 sub-index 01 to 40B

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	00	18	01	0B	04	00	80
Response:	60	00	18	01	00	00	00	00

Set new COB-ID valid with index 1800 sub-index 01

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	23	00	18	01	0B	04	00	00
Response:	60	00	18	01	00	00	00	00

Detail of a trace:

ID	Dir	DLC	Data	Interpretation
73F	Rx	1	00	Boot up message from encoder
63F	Rx	8	23 11 10 01 6C 6F 61 64	Load factory defaul
5BF	Rx	8	60 11 10 01 00 00 00 00	answer from encoder
0	Rx	2	81 3F	Reset node
73F	Rx	1	00	Boot up message from encoder
63F	Rx	8	2F 00 21 00 05 00 00 00	Set to 250kHz baude rate
5BF	Rx	8	60 00 21 00 00 00 00 00	answer from encoder
63F	Rx	8	2F 01 21 00 3F 00 00 00	Set node adress to 3f
5BF	Rx	8	60 01 21 00 00 00 00 00	answer from encoder
63F	Rx	8	2F 02 21 00 00 00 00 00	termination off
5BF	Rx	8	60 02 21 00 00 00 00 00	answer from encoder
63F	Rx	8	23 05 21 00 73 61 76 65	save all bus objects
5BF	Rx	8	60 05 21 00 00 00 00 00	answer from encoder
			power off/on encoder comes back with the new bus settings	
73F	Rx	1	00	Boot up message from encoder
63F	Rx	8	2B 00 62 00 0A 00 00 00	set event time to 10ms
5BF	Rx	8	60 00 62 00 00 00 00 00	answer from encoder
63F	Rx	8	2B 00 60 00 01 00 00 00	set ccw counting direction
5BF	Rx	8	60 00 60 00 00 00 00 00	answer from encoder
63F	Rx	8	23 10 10 01 73 61 76 65	save all
5BF	Rx	8	60 10 10 01 00 00 00 00	answer from encoder
0	Rx	2	01 3F	switch to operational mode
1BF	Rx	4	BA 4C 00 00	position values
1BF	Rx	4	BA 4C 00 00	position values
1BF	Rx	4	BA 4C 00 00	position values
1BF	Rx	4	BA 4C 00 00	position values
0	Rx	2	80 3F	switch to preoperational mode