CAN in Automation e. V.

CANopen

Electronic Data Sheet Specification for CANopen

CiA Draft Standard Proposal 306

Version 1.0

Date: 31.05.2000

History

Date	Changes
May 2000	Initial revision based on Appendix 12 of CiA DS-301 V3.0; Summary of major changes: - clarification and more detailed specification - adjustments to errata sheet and DS-302 - denotation - compact storage - module concept

Table of Contents

1 Scope

The usage of devices in a communication network requires configuration of the device parameters and communication facilities. CANopen defines a standardised way to access these parameters via the object dictionary.

For handling of the complexity of CANopen systems Software Tools are required. This reduces the complexity of the planning, configuration and analysis process and significantly increases the security of the system.

For this purpose Software Tools need an electronic description of the CANopen devices. To allow the usage of manufacturer independent Tools, this document defines a standardised file format – called Electronic Data Sheet EDS.

Furthermore some derived file formats are specified. The DCF describes a concrete incarnation of a device configuration. The MDS describes modules of devices with a modular structure.

2 References

/1/ CiA DS-301, CANopen - Application Layer and Communication Profile, Version 4.0 June 1999
/2/ CiA DSP-302, Framework for Programmable CANopen Devices, Version 2.0 November 1998
/3/ CiA DR-303-4, LSS – Layer Setting Services and Protocol, in preparation

3 Definitions and Abbreviations

CAN	Controller Area Network
CiA	CAN in Automation international users and manufacturers group e.V.
COB	Communication Object. (CAN Message) A unit of transportation in a CAN Network. Data must be sent across a Network inside a COB.
COB-ID	COB-Identifier. Identifies a COB uniquely in a Network. The identifier determines the priority of that COB in the MAC sub-layer too.
DCF	Device Configuration File
DIN	Deutsches Institut für Normung
EDS	Electronic Data Sheet
ISO	International Standardisation Organisation
LSS	Layer Settings Specification
MDS	Module Data Sheet
NMT	Network Management. One of the service elements of CANopen Application Layer in the CAN Reference Model. It performs initialisation, configuration and error handling in a CANopen network.
OSI	Open Systems Interconnection
PDO	Process Data Object
SDO	Service Data Object

4 Electronic Data Sheet

In order to give the user of a CANopen device more support the device's description should be available in a standardised way. This gives the opportunity to create standardised tools for:

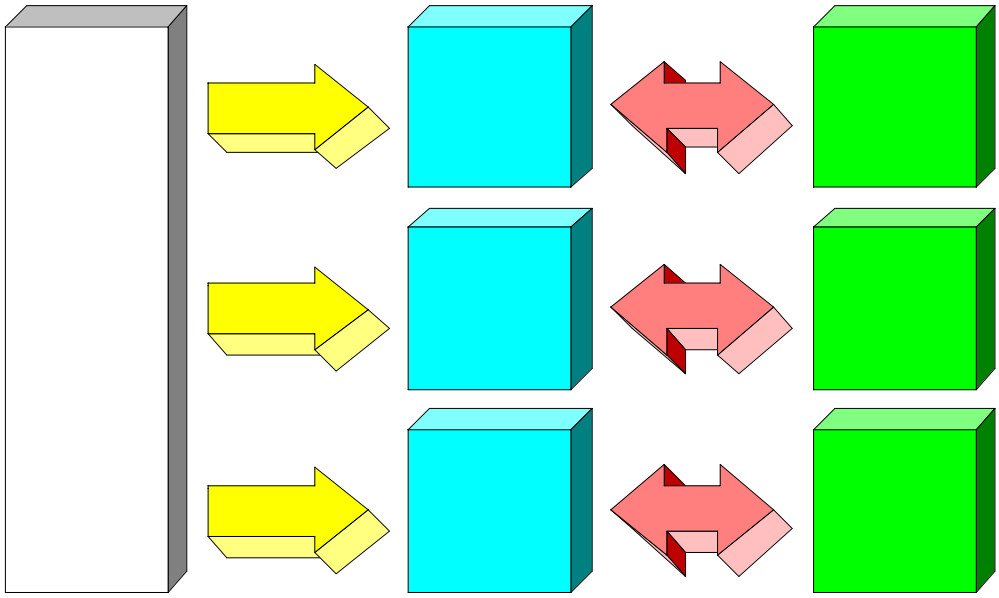
- configuration of CANopen devices,
- designing networks with CANopen devices,
- managing project information on different platforms.

Therefore two types of files are introduced to define a CANopen device with electronically means.

An EDS can be used to describe the:

- Communication functionality and objects as defined /1/ and Application Frameworks DS-3xx
- Device specific objects as defined in the device profiles DS-4XX.

The EDS is the template for a device „XY" of the vendor „UV". The DCF describes the incarnation of a device not only with the objects but also with the values of the objects. Furthermore a value for the baudrate of a device and for the Node-ID are added.



An EDS should be supplied by the vendor of a particular device. If a vendor has no EDS available for his CANopen devices a default EDS might be used. The default EDS comprises all entries of a device profile for a particular device class. The user has to be aware, that the description is different from the concretely implemented features of that device, what might cause severe problems!

4.1 Basic Structure

The files are ASCII-coded, the ANSI character set shall be used.

The lines can be ended by a LF character or by a CR / LF combination. The total length of a line must not exceed 255 characters.

The EDS contains several sections, each of which consists of a group of related entries. The sections and the entries are listed in the following format:

```
[section name]
keyname=value
```

In this example, [section name] is the name of a section. The enclosing brackets ([]) are required, and the left bracket must be in the leftmost column. Section names are not case sensitive.

The keyname=value statement defines the value of each entry. A keyname is the name of an entry. It can consist of any combination of letters and digits, and must be followed immediately by an equal sign (=). The keyname is not case sensitive. If the keyname consists only of digits, it is interpreted as a string, not as a number. This means, that the entry 10=xxx is not the same as 0xA=xxx and 0xA=xxx is not the same as 0x0A=.... The same applies to section names.

The value is a string, which can be interpreted depending on the entry (see below).

You can include comments and empty lines in EDS files. You must begin each line of a comment with a semicolon (;). It must be in the leftmost column.

The sections can appear in any order inside the file. Inside each section, the entries can appear in any order.

If not specified otherwise, all sections and entries in this paper are mandatory. In order to support future extensions it is allowed to include additional sections and additional entries inside the sections. Anyhow the Conformance Test Tool will recognise this with warning messages.

4.2 Entry Value Interpretation

The interpretation of the values depends on each specific entry. Some general rules are defined:

Leading and trailing white space is trimmed. The line

```
keyname=value
```

is interpreted the same way as

```
keyname=    value
```

Integer numbers can be written as decimal numbers, hexadecimal numbers or octal numbers. Hexadecimal numbers are preceded by 0x. Octal numbers start with a leading 0 (not followed by x). If the entry contains a number the following lines are identical:

```
keyname=10
keyname=0xa
keyname=0x0a
keyname=0xA
keyname=0x000A
keyname=012
```

If an entry hasn't a value, this is denoted by End-Of-Line after the equal sign (empty entry). A missing entry is interpreted the same way as an entry without value.

String values are stored without quotes.

Octett Strings and raw data of domains are stored as sequence of hexadecimal bytes (without leading 0x). Bytes with a high nibble of 0 must be stored with the leading 0. If the data does not fit within one line, it may be stored in a separate file (refer to chapter 5.2.1).

Example for octett string:

DemoSeq=01a1053c45aabbccddeeff

Bitstrings are stored as a sequence of 0 and 1.

Example:

BitDemo=11001010000111

For entries of one of the integer types it is allowed to use a formula. This gives the chance to describe values, that depend on other values. For example, the COB-ID of the default PDOs depend on the Node-ID of the device.

The syntax of the formula is given by the following EBNF description:

```
IntEntryValue = $NODEID { ”+” number }
```

For concrete devices, $NODEID will be replaced by the actual Node-ID of the device. The $NODEID must appear at the beginning of the expression. Otherwise the line is interpreted as without a formula. Actually it is only possible to use an offset to this Node-ID in the formula. More complex expressions are not allowed.

4.3 File Information

The EDS contains information about itself. This is useful for version control management. This information is stored in the section `[FileInfo]`.

The following keywords are used:

`FileName`	file name (according to OS restrictions),
`FileVersion`	actual file version (Unsigned8),
`FileRevision`	actual file revision (Unsigned8),
`EDSVersion`	Version of Specification (3 characters) in the Format “x.y”. If the entry is missing, this is equal to “3.0”.
`Description`	file description (max 243 characters),
`CreationTime`	file creation time (characters in format „hh:mm(AM|PM)“),
`CreationDate`	date of file creation (characters in format „mm-dd-yyyy“),
`CreatedBy`	name or description of file creator (max 245 characters),
`ModificationTime`	time of last modification (characters in format „hh:mm(AM|PM)“),
`ModificationDate`	date of last file modification (characters in format „mm-dd-yyyy“),
`ModifiedBy`	name or description of the modification (max 244 characters).

Example:

```
[FileInfo]
FileName=vendor1.eds
FileVersion=1
FileRevision=2
EDSVersion=4.0
Description=EDS for simple I/O-device
CreationTime=09:45AM
CreationDate=05-15-1995
CreatedBy=Zaphod Beeblebrox
ModificationTime=11:30PM
ModificationDate=08-21-1995
ModifiedBy=Zaphod Beeblebrox
```

4.4 General Device Information

The EDS contains device specific information about

- vendor name,
- vendor ID,
- device name,
- device code,
- version information,
- LSS-information (parts of the LSS-address),
- device abilities.

This can be found in the section `[DeviceInfo]`.

The following keywords are used:

`VendorName` vendor name (max 244 characters)

`VendorNumber` unique vendor ID according to identity object sub 1 (Unsigned32)

`ProductName` product name (max 243 characters)

`ProductNumber` product code according to identity object sub 2 (Unsigned32)

`RevisionNumber` product revision number according to identity object sub 3 (Unsigned32)

`OrderCode` order code for this product (max 245 characters)

`BaudRate_10` supported baud rates (Boolean, 0 = not supported, 1 = supported)

`BaudRate_20`
`BaudRate_50`
`BaudRate_125`
`BaudRate_250`
`BaudRate_500`
`BaudRate_800`
`BaudRate_1000`

`SimpleBootUpMaster`
simple boot-up master functionality (Boolean, 0 = not supported, 1 = supported),

`SimpleBootUpSlave`
simple boot-up slave functionality (Boolean, 0 = not supported, 1 = supported),

`Granularity` this value gives you the granularity allowed for the mapping on this device - most of the existing devices support a granularity of 8 (Unsigned8; 0 - mapping not modifiable, 1-64 granularity)

`DynamicChannelsSupported`
according to DS-302 this entry describes the facility of dynamic variable generation. If value is not 0, the additional section `DynamicChannels` exists. Details are given in CiA DS-302 and CiA DS-405.

`GroupMessaging` according to DS-301 Annex A this entry describes the facility of multiplexed PDOs. (Boolean, 0 = not supported, 1 = supported) Details are given in DS-301.

`NrOfRXPDO` number of Receive PDOs supported. (Unsigned8)

`NrOfTXPDO` number of Transmit PDOs supported. (Unsigned8)

`LSS_Supported` support of LSS functionality (Boolean, 0 = not supported, 1 = supported)

For compatibility reasons, the entries `ProductVersion`, ProductRevision, `LMT_ManufacturerName`, `LMT_ProductName`, `ExtendedBootUpMaster` and `ExtendedBootUpSlave` are reserved.

Example:

```
[DeviceInfo]
VendorName=Nepp Ltd.
VendorNumber=156678
ProductName=E/A 64
ProductNumber=45570
RevisionNumber=1
OrderCode=BUY ME - 177/65/0815
LSS_Supported=0
BaudRate_50=1
BaudRate_250=1
BaudRate_500=1
BaudRate_1000=1
SimpleBootUpSlave=1
SimpleBootUpMaster=0
NrOfRxPdo=1
NrOfTxPdo=2
```

4.5 Object Dictionary

In this logical part of the EDS the following information can be found:

1. which objects of the object dictionary are supported,
2. limit values for parameters,
3. default values.
4. data types
5. additional information

The description of the objects take place in separate parts corresponding to:

- mandatory objects,
- optional objects,
- manufacturer specific objects.

4.5.1 Mapping of dummy entries

Sometimes it is required to leave gaps in the mapping of a device. This means that e.g. a device only evaluates the last two data bytes of a PDO of 8 bytes length. The first six bytes should be ignored (perhaps they are evaluated by another device). In this case the mapping of this device must contain dummy entries for these first six bytes.

The indices from the data type area of the object dictionary are used for this purpose. The user of a device has to know which data type can be used for creating dummy entries and which not (indeed only the length of the dummy object is important).

The section `DummyUsage` is used for describing dummies. The entries follow this scheme:

```
Dummy<data type index (without 0x-prefix)>={0|1}
```

Example:

```
[DummyUsage]
Dummy0001=0
Dummy0002=1
Dummy0002=1
Dummy0003=1
Dummy0004=1
Dummy0005=1
Dummy0006=1
Dummy0007=1
```

This means that the device will support the mapping of the data types Integer8/16/32 and Unsigned8/16/32.

4.5.2 Object Descriptions

4.5.2.1 Object lists

The Object Dictionary is structurally divided into three parts:

- Mandatory Objects in `[MandatoryObjects]` contains only the mandatory objects. These are at least the objects 1000H and 1001H. For devices, that have implemented Version 4.0 of CANopen, additionally the object 1018H.
- Optional Objects in `[OptionalObjects]` contains all other objects of the area 1000H-1FFFH and 6000H-FFFFH.
- Manufacturer Specific Objects in `[ManufacturerObjects]` contains all manufacturer specific objects (located in 2000H-5FFFH).

Each of these sections contains a list of the supported objects. Each list contains the entry

`SupportedObjects` - number of entries in the section (Unsigned16)

The entries are decimal numbered beginning with number 1. This way the last entry gives the number of available entries.

```
[OptionalObjects]
SupportedObjects=10
1=0x1003
2=0x1004
3=0x1005
4=0x1008
5=0x1009
6=0x100A
7=0x100C
8=0x100D
9=0x1010
10=0x1011
```

4.5.2.2 Object description

Each of the listed objects has to be described in an own section. The section names are all constructed following the same scheme. The section name is constructed according to

[<Index>]

using the hexadecimal values for Index and Sub-Index without the leading „0x“ and without further leading 0.

In a section the following keywords may exist:

SubNumber	number of sub-indices available at this Index (Unsigned8), not counting Sub-Index FFH. This allows the description of sub-objects as defined below. This entry is empty or can be missing, if no sub-objects exist.
ParameterName	parameter name (up to 241 characters)
ObjectType	This is the object code
DataType	This is the Index of the data type of the object in the object dictionary
LowLimit	Lowest limit of object value (only if applicable).
HighLimit	Upper limit of object value (only if applicable).
AccessType	Access type for this object (String „ro“ - read only, „wo“ - write only, „rw“ - read/write, „rwr“ - read/write on process input, „rww“ - read/write on process output, „const“ - constant value)
DefaultValue	default value for this object,

PDOMapping	Flag, if this object can be mapped into a PDO (Boolean, 0 = not mappable, 1 = mappable).
ObjFlags	Optional entry for assignment of special behaviour. See below.

Objects with object code VAR:

HighLimit and LowLimit are optional. DefaultValue is optional. SubNumber is not supported.

Objects with object code ARRAY or RECORD

HighLimit, LowLimit, DefaultValue are not supported. SubNumber is mandatory. DataType, AccessType, PDOMapping are not supported. For exceptions according to compact descriptions refer to chapter 4.5.2.4.

Example:

```
[1000]
ParameterName=Device Type
ObjectType=0x7
DataType=0x0007
AccessType=ro
DefaultValue=
PDOMapping=0
```

To described a structured object (object has Sub-Indexes) each Sub-Index is described in an own section. The section name is constructed according to the rule

[<Index>(sub<Sub-Index>)]

using the hexadecimal values for Index and Sub-Index without the leading „0x“ and without further leading 0.

Example:

```
[1003]
SubNumber=2
ParameterName=Pre-defined Error Field
ObjectType=8


[1003sub0]
ParameterName=Number of Errors
ObjectType=0x7
DataType=0x0005
AccessType=ro
DefaultValue=0x1
PDOMapping=0

[1003sub1]
ParameterName=Standard Error Field
ObjectType=0x7
DataType=0x0007
AccessType=ro
DefaultValue=0x0
PDOMapping=0
```

Application hint:

In principle it is possible, that a list of sub-object does not have consecutive Sub-Indexes. The value of Sub-Index 0 always stores the highest Sub-Index implemented. In contrast, the EDS entry SubNumber contains the number of Sub-Index implemented, including the Sub-Index 0.

Example:

```
[1010]
SubNumber=2
ParameterName=Store Parameters
ObjectType=8


[1010sub0]
ParameterName=largest Sub-Index supported
ObjectType=0x7
DataType=0x0005
AccessType=ro
DefaultValue=0x4
PDOMapping=0

[1010sub4]
ParameterName=save manufacturer defined parameters
ObjectType=0x7
DataType=0x0007
AccessType=ro
DefaultValue=0x1
PDOMapping=0
```

4.5.2.3 Specific Flags

The entry `ObjFlags` allows to define a specific behaviour for Tools how to treat an object. Example:

A typical task for a configuration software is the Download of a configured DCF file. Doing this without special recognition of special objects leads to the following problem: Objects such as 1010H "Store parameters" will be written with either invalid values or at least in an invalid order. First the objects 1000H up to 100FH are written, then "Store Parameters" and then the other parameters. This will lead to inconsistencies and is not what the user expects. One solution could be the special treatment of such objects by the configuration software. But even then there may happen the case, that device profiles or manufacturer specific objects have a similar problem.

These special objects are marked in the EDS and DCF files. The object description sections may contain an entry `ObjFlags` with an unsigned32 content:

The lowest bit is a boolean value (0=false, 1=true) for "Refuse write on Download", the second bit is a boolean value for "Refuse read on Scan", the other bits are reserved for further use by CiA and have to be 0.

If the entry is missing, this equals having the value 0. It is recommended to write the entry in the EDS/DCF only if it is not 0. This avoids unnecessary increase of the file size.

4.5.2.4 Compact Storage

For devices with many objects and especially many arrays the EDS file might be very big. The load and store process may reach unacceptable times. For this reason the following definitions shall help to store the really necessary information much more compact.

4.5.2.4.1 PDO Definitions

In principle the object descriptions for PDOs are all nearly the same. The most important information is the number of TX and RX PDOs which is given by 4.4. It is

allowed to leave all PDO object descriptions. To mark this, the entry `CompactPDO` (Unsigned8) must be added to the section `DeviceInfo`. It contains the implemented sub-indexes as a bitmask. The lowest bit defines, if Sub-Index 1 is implemented, the second bit defines, if Sub-Index 2 is implemented and so forth.

```
[DeviceInfo]
...
CompactPDO=0x3
```

The appropriate data types are implicitly known by the CANopen specifications as well as the default values for the first PDOs COB-Ids. The other values such as TransmissionType and Mapping very often are the goal of Project Planning rsp. Configuration and do not need to be known on load time.

If a PDO is described explicitly, all sub-objects of the communication parameters as well as of the mapping parameters of this PDO must be described.

4.5.2.4.2 Array Values

Most often all sections of the Sub-Indexes of an array are equal except the name. It is allowed to describe only a template in the main object. For this the additional unsigned8 entry `CompactSubObj` can be added. If this exists and has a value not equal 0, then

- the names are assumed to be `xxxn` with xxx as the name of the main object and n as the decimal Sub-Index. Sub-Index 0 has the Name `NrOfObjects`
- the object types are assumed to be VAR
- the data type for all Sub-Indexes except 0 and 255 is given by the entry `DataType` of the main object. Sub-Index 0 always has the data type Unsigned8.
- the limits are assumed to be NONE
- the access type for all Sub-Indexes except 0 and 255 is given by the entry `AccessType` of the main object. The access type for Sub-Index 0 is assumed to be ReadOnly.
- the default values for all Sub-Indexes except 0 and 255 is given by the entry `DefaultValue` of the main object. The default value for Sub-Index 0 is the number given by `CompactSubObj`
- the entry `PDOMapping` for all Sub-Indexes except 0 and 255 is given by the entry `PDOMapping` of the main object. Sub-Index 0 is assumed not to be mappable.

It must be assumed, that the Sub-Index list does not contain any gaps.

If `CompactSubObj` is used, the entry `SubNumber` is not supported, it has to be 0 or empty or shall not appear.

It is possible to assign explicit names, if the default names are not useful enough. For this a list of according names can appear. The section name is given by `[xxxxName]` with xxxx as the Index. The entry `NrOfEntries` gives the number of names in the list. The names are listed with using their Sub-Index as decimal entry name (starting with 1).

Example:

```
[2050Name]
NrOfEntries=3
1=NameOfSubIndex1
2=NameOfSubIndex2
15=NameOfSubIndex15
```

The names not listed here are built upon the rule given above.

4.5.2.4.3 Network Variables

In case of Programmable Devices according to CiA DS-302 rsp. CiA DS-405 the description of the dynamic network variable arrays are not written in the EDS. All necessary information is already given by the section `DynamicChannels`. To ensure a consistent interpretation of the EDS it is not allowed to describe the dynamic network variable sections!

Description for Network variables, that are not treated dynamically, but are completely described in the EDS, may use the `CompactSubObj` mechanism.

4.5.3 Object Links

In order to ease the implementation of a configuration tool it is possible to group related objects together via the keyword `ObjectLinks`.

An object link has the following structure:

```
[<index>ObjectLinks]
ObjectLinks=<number of available links>
1=<index of 1st linked object>
2=<index of 2nd linked object>
3=<index of 3rd linked object>
4=<index of 4th linked object>
5=<index of 5th linked object>
:
```

The list of object links is numbered decimal beginning with number 1.

Example:

```
; assuming we describe closed loop
; this is the object „factor"
[5800ObjectLinks]
ObjectLinks=0x3
; gain
1=0x5801
; zero
2=0x5802
; pole
3=0x5803
```

4.5.4 Comments

Comments can be added to the EDS by using the `Comments` section. This section has only entries determining the line number and the line contents.

`Lines` - number of commentlines (Unsigned16)

`Line<line number>` - one line of comment (max 249 characters). The number is decimal coded.

Example:

```
[Comments]
Lines=3
Line1=|-------------|
Line2=| Don't panic |
Line3=|-------------|
```

5 Device Configuration File DCF

The device configuration file comprises all objects for a configured device. The device configuration file has the same structure as the EDS for this device. There are some additional entries in order to describe the configured device.

5.1 File Information Section

`LastEDS` - file name of the EDS file used as template for this DCF

5.2 Object Sections

5.2.1 Parameter Value in standard description

`ParameterValue` - object value (as defined by `ObjectType` and `DataType`)

Example:

```
; value for object 1006 (communication cycle period)
[1006]
SubNumber=0
ParameterName=Communication Cycle Period
ObjectType=0x7
DataType=0x0007
LowLimit=1000
HighLimit=100000
DefaultValue=20000
AccessType=ro
ParameterValue=15000
PDOMapping=0
```

If the ObjectType is Domain (0x2) the value of the object can be stored in a file:

UploadFile: if a read access is performed for this object, the data are stored in this file (character 244; according to OS restrictions)

DownloadFile: if a write access is performed for this object, the data to be written can be taken from this file (character 242; according to OS restrictions)

Example:

```
; manufacturer specific object 5600 (downloadable program)
[5600]
ParameterName=Real Good Program (RGP)
ObjectType=0x2
DataType=0x000F
AccessType=wo
DownloadFile=C:\FAST\PROGRAMS\FIRST.HEX

; manufacturer specific object 5700 (core dump)
 [5700]
ParameterName=Core Dump (CD)
ObjectType=0x2
DataType=0x000F
AccessType=ro
UploadFile=C:\FAST\DEBUG\DUMPALL.HEX
```

5.2.2 Denotation

When using a DCF in a concrete application it is useful to assign application specific names to the objects. This can be done by simply renaming the entry value of `ParameterName`. As an alternative it is possible to write the changed names into an extra entry called `Denotation`.

Example entry in EDS:

```
[6000sub1]
ParameterName=Dig.Input Lines 0-7
...
```

Example entry in DCF by changing `ParameterName`:

```
[6000sub1]
ParameterName=ApplicationSpecificName1
...
```

Example entry in DCF with `Denotation`:

```
[6000sub1]
ParameterName=Dig.Input Lines 0-7
Denotation=ApplicationSpecificName1
...
```

Using the mechanism of simple renaming has the advantage of smaller file size and easier implementation since `ParameterName` is mandatory and therefore always available.

Using the entry `Denotation` opens the possibility of re-generation of the according EDS file with the original object names.

5.2.3 Compact Storage

Refer to Compact Storage of EDS in chapter 4.5.2.4. With the same background it is required to have the possibility of a compact storage format for the DCF files.

5.2.3.1 PDO Definitions

If the entry `CompactPDO` of the section `DeviceInfo` exists and is not 0, it is optionally allowed to write only the Object and Sub-Object description for the concretely configured PDOs. All other PDOs are disabled.

5.2.3.2 Array Values

If the entry `CompactSubObj` exists and has a value not equal to 0, all the `ParameterValues` of the sub-objects are stored in an extra list. The section name is given by `[xxxxValue]` with xxxx as the Index. The entry `NrOfEntries` gives the number of entries in the list. The values are listed with using their Sub-Index as decimal entry name (range 1-254). All missing entries in the list have a value of `DefaultValue`.

Example:

```
[2050]
SubNumber=0
ParameterName=A big array
ObjectType=8
DataType=0x0007
AccessType=rw
DefaultValue=0x0
PDOMapping=0
CompactSubObj=200

[2050Value]
NrOfEntries=3
1=200
2=0xab
15=100
```

Using the same syntax it is allowed to store changed `ParameterNames` in an extra list. The section name is given by `[xxxxDenotation]` with xxxx as the Index. The entry `NrOfEntries` gives the number of entries in the list. The values are listed with using their Sub-Index as decimal entry name (range 1-254). All missing entries in the list have a name according to the default rule or the `[xxxxName]` section as described in EDS chapter 4.5.2.4.2.

5.2.3.3 Network Variable Values

Normally it is not required to configure concrete values for dynamic network variables. If there is a need to do so, the values must be stored in the standard format as given in 5.2.1. The reason is, that network variable arrays might have gaps, but it is not possible to handle gaps with the `CompactSubObj` mechanism.

Network variables, that are not treated dynamically (`DynamicChannelsSupported=0`) may be stored with the `CompactSubObj` mechanism.

5.3 Device Commissioning

There is an additional section in the DCF named `DeviceComissioning`:

`NodeID` device's address (Unsigned8)

`NodeName` node name (max 246 characters)

`Baudrate` device's baudrate (Unsigned16)

`NetNumber` number of the network (Unsigned32)

`NetworkName` name of the network (max 243 characters)

`CANopenManager` describes, if the device is the CANopen Manager. (boolean, 1 = CANopen Manager, 0 or missing = not the Manager)

`LSS_SerialNumber` serial number according to identity object sub 4 (Unsigned32)

Example:

```
[DeviceComissioning]
NodeID=2
NodeName=DEVICE2
Baudrate=1000
NetNumber=42
NetworkName=very important subnet in a big network
LSS_SerialNumber=9912345
```

6 Module Concept

A very common way for building flexible devices is using a bus coupler device which can be extended by modules. The base device automatically detects the existence of extension modules.

The methods commonly used are clearly structured and straight forward. Extending such a device by modules leads to a varying amount of process data and configuration data. Concerning CANopen this results in varying object dictionaries. For the extension of process data within the object dictionary, two methods are very common.

In example for I/O modules (Profile DS-401) this is the usage of sequential Sub-Indexes. For example, the first 8 bit digital input device creates the object 6000H, 01H, the second creates object 6000H, 02H and so forth. The first digital output device works on 6200H, 01H. Additional there might be global objects that have to be created when at least one module of a specific type is connected. For example DSP-401 defines the "Global_Interrupt_Enable" object 6423H, that may be used if at least one analogue input is connected. Connecting further analogue inputs will not duplicate that object.

Another method is given in DS-301. There the Multi Device Module uses the approach of shifting a copy of the object dictionary structure by an offset of 800H. The first object dictionary starts with object 6000H, the second with 6800H and so on. Object 1000H defines the device as Multi Device Module.

Since the combination of both mechanisms actually is not as usual in the market, the following specification treats only the Sub-Index method. This keeps the specification as compact as possible and does not exclude future extensions.

6.1 Electronic Data Sheet

6.1.1 Assignment of extension modules

The EDS of the bus coupler device contains a list of supported extension modules. For each type of extension module a Module Description MD describes the features of the module. The number of the supported MD is stored in the EDS section `[SupportedModules]`.

`NrOfEntries` Number of supported extension modules (Unsigned16)

6.1.2 PDOs

The pre-defined PDO mapping can be used according to the appropriate profile. Each new Sub-Index can be filled into the corresponding pre-defined PDO. This way up to 8 byte of digital input data, 8 bytes of digital output data, 4 analogue inputs and 4 analogue outputs can be mapped. Further data may be mapped to additional PDOs. Since they are not pre-defined and since their COB-IDs have to be marked as invalid, their usage has to be switched free on boot-up. In systems using the pre-defined connection set, the application performs this task and will know the appropriate usage of the additional PDOs. In more complex systems, a configuration has to be created that includes configuration of the mapping. In that case the Configuration Manager has the task of enabling all PDOs.

Using pre-defined PDOs requires the knowledge of the corresponding mapping entries. Since this mapping depends on the extension modules it can not be entered in the EDS. A generic method, that fulfils all possible cases will be very complex. To leave this specification as simple as possible, only one rule is given:

If the EDS contains a non-empty list of extension modules, it is allowed to describe mapping entries with objects, that are not necessarily existing. When the concrete configuration of the module is known, the mapping list is valid up to the first not existing mapped object. The Sub-Index 0 then is shortened appropriately.

This rule will not allow to describe every case, but a wide range of practical implementations.

Example:

The EDS of a IO bus coupler contains the following descriptions: First TPDO mapping for objects 6000H,1 up to 6000H,8. The third TPDO for 6000H,9 up to 6000H,16. Two 16 bit extension modules shall be added. Then the first TPDO will be valid. It's mapping contains the entries 6000H,1 up to 6000H,4. The Sub-Index 0 is shortened to 4. The third TPDO is invalid.

6.2 Module Description

The module description is placed in sections of which names begin with `Mx`, where `x` is the decimal counter beginning with 1 up to the value of `NrOfEntries` from section `[SupportedModules]` without leading 0. In the following this is always abbreviated with `Mx`. Each module description has some entries in the section `MxModuleInfo`:

`ProductName`	Name of the product (max 243 characters)
`ProductVersion`	Version information (Unsigned8)
`ProductRevision`	Version information (Unsigned8)
`OrderCode`	Manufacturer specific order code (max 245 characters)

A textual module description appears in the optional section `[MxComments]`.

`Lines` - number of comment lines (Unsigned16)

`Line<line number>` - one line of comment (max 248 characters). The number is decimal coded.

Example:

```
[M1Comments]
Lines=2
Line1=Module with 16 input lines
Line2=and 8 output lines.
```

If required, the device has to instantiate objects on specific fixed Indexes if at least one module of that type is connected. These objects are described in a list in the section `[MxFixedObjects]`:

`NrOfEntries`	Number of fixed objects (Unsigned16)
`1=0x6423`	The number N is decimal, starting with 1.
`2=`	
`N=...`	

The fixed objects are placed in sections

`MxFixedxxxx`.

xxxx is the hexadecimal Index without leading 0x and without any further leading 0. The contents of those sections are the same as specified for object descriptions in chapter 4.5.2.2. Sub-Objects are defined in the same way. The naming of the Sub-Object sections is

`MxFixedxxxxsubx`

`xxxx` is the hexadecimal Index without leading 0x. `x` is the hexadecimal Sub-Index without leading 0x and without leading 0.

If several modules contain the same fixed objects, their attributes shall be equal.

The section [MxSubExtends] contains a list of all objects that instantiate new Sub-Indexes.

NrOfEntries — Number of extended objects (Unsigned16) followed by a decimal counted list starting with 1:

```
1=0x6000
2=...
```

The objects are placed in sections [MxSubExtxxxx]. **xxxx** is the hexadecimal Index without leading 0x and without any further leading 0. In this section the same entries as in a standard object description of an EDS according to chapter 4.5.2.2 exist. Additional entries are

Count — Number of extended Sub-Indexes with this description that are created per module. The format is Unsigned8 [;Unsigned8].

If one or more Sub-Indexes are created per attached module to build a new sub-index, then Count is that Number. In example 32 bit module creates 4 Sub-Indexes each having 8 Bit: Count=4

If several modules are gathered to form a new Sub-Index, then the number is 0, followed by semicolon and the number of bits that are created per module to build a new Sub-Index. In example 2 bit modules with 8 bit objects: The first Sub-Index is built upon modules 1-4, the next upon modules 5-8 etc.: Count=0;2. The objects are created, when a new byte begins: Module 1 creates the Sub-Index 1; modules 2-4 fill it up; module 5 creates Sub-Index 2 and so forth.

ObjExtend — Optional Unsigned8 entry. If an array is filled up to the end, the next object may be used. For example, in DS-401 the object 6020H addresses 128 single input lines. Line 129 is addressed with the next main Index 6021H.
The value of this entry defines the maximum Sub-Index after which the next object is used. If the value is 0 or the entry is missing, the next object can not be used.

If several modules contain the same extension objects, their attributes shall be equal.

6.3 Device Configuration File

With the information of the bus couplers EDS and the extension module descriptions configuration tools are able to create the correct object dictionary and write it into the DCF. For this task no DCF specification changes are necessary.

For an easier handling it is desirable to store the information of which modules the device consists. This can be done by copying the section [SupportedModules] into the DCF and then creating a reference list in the section [ConnectedModules]:

NrOfEntries — Number of connected modules (Unsigned16)

```
1=3
2=3
x=...
```

x is the decimal list counter, starting with 1. The Unsigned16 value is a reference to the [SupportedModules] list. The corresponding Module Descriptions can be found according to the rules of the EDS (Section names beginning with Mx, see 6.2).

Example:

```
[SupportedModules]
NrOfEntries=4

[ConnectedModules]
NrOfEntries=3
1=2
2=2
3=4
```

This device has three modules connected. The first and second are described in sections named with `M2...` , the third is described in sections named with `M4...`

Note: The ordering of the devices is important since this defines the assignment of objects to physical lines.

6.4 Example

The following simple example describes a base device, that can be extended by digital input and output devices:

EXAMPLE.EDS

```
[FileInfo]
FileName=example.eds
.........

[DeviceInfo]
VendorName=XYZ
ProductName=DemoDevice
OrderCode=0
..........

[SupportedModules]
NrOfEntries=2

[MandatoryObjects]
SupportedObjects=2
1=0x1000
2=0x1001

[1000]
ParameterName=Device Type
......

[OptionalObjects]
SupportedObjects=8
1=0x1008
2=0x1009
3=0x100a
4=0x1004
5=0x1400
6=0x1600
7=0x1800
8=0x1a00

.......
```

```
[M1ModuleInfo]
ProductName=Digital Input Module
ProductVersion=1
ProductRevision=0
OrderCode=DI4711

[M1SubExtends]
NrOfEntries=1
1=0x6000

[M1SubExt6000]
ParameterName=ReadState8InputLines
DataType=5
DefaultValue=0
PDOMapping=1
AccessType=ro
Count=1

[M2ModuleInfo]
ProductName=Digital Output Module
ProductVersion=1
ProductRevision=0
OrderCode=DO0815

[M2SubExtends]
NrOfEntries=1
1=0x6200

[M2SubExt6200]
ParameterName=WriteState8OutputLines
DataType=5
DefaultValue=0
PDOMapping=1
AccessType=rww
Count=1
```