CiA Draft Standard 306

Electronic data sheet specification for CANopen

Version 1.3

01 January 2005

History

Date	Version	Changes
2005-01-01	*1.3*	*Publication as Draft Standard*

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.

Furthermore some derived file formats are specified. The Device Configuration File describes a concrete incarnation of a device configuration. The Module Data Sheet describes modules of devices with a modular structure.

2 References

/CiA301/	CiA 301, CANopen application layer and communication profile
/CiA302/	CiA 302, CANopen framework for CANopen managers and programmable CANopen devices
/CiA401/	CiA 401, CANopen device profile generic I/O modules
/CiA405/	CiA 405, CANopen interface and device profile for IEC 61131 programmable devices
/ISO646/	ISO/IEC 646, ISO 7-bit coded character set for information interchange

3 Abbreviations and definitions

3.1 Abbreviations

ASCII	American standard code for information interchange
CAN	Controller area network
COB	Communication object
COB-ID	COB identifier
CR	Carriage return
DCF	Device configuration file
DIN	Deutsches Institut für Normung
EBNF	Extended Backus-Naur form
EDS	Electronic data sheet
ID	Identifier
ISO	International standardisation organisation
LF	Line feed
LSS	Layer settings specification
MD	Module description
MDS	Module data sheet
NMT	Network management
OS	Operation system
OSI	Open systems interconnection
PDO	Process data object
RPDO	Receive PDO
SDO	Service data object
TPDO	Transmit PDO

3.2 Definitions

The definitions given in /CiA301/ and /CiA405/ shall apply to this specification, too.

Notation for the time:

hh:mm (AM/PM)

hh:	two decimal digits which represent the hours
mm:	two decimal digits which represent the minutes
(AM/PM)	two characters which represent if the time is located forenoon or afternoon

Notation for the date:

mm:dd:yyyy

mm:	two decimal digits which represent the month
dd:	two decimal digits which represent the day
yyyy	four decimal digits which represent the year

4 Electronic data sheet specification

4.1 General

In order to give the user of a CANopen device more support the device description shall 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 may be used to describe the:

- Communication functionality and objects as defined in /CiA301/ and Application Frameworks
- Device specific objects as defined in the device profiles

The EDS serves as template for a device „XY" of the vendor „UV". The DCF describes the incarnation of a configured device not only with the objects but also with the values of the objects. Furthermore it contains a value for the bitrate of a device and for the node-ID.

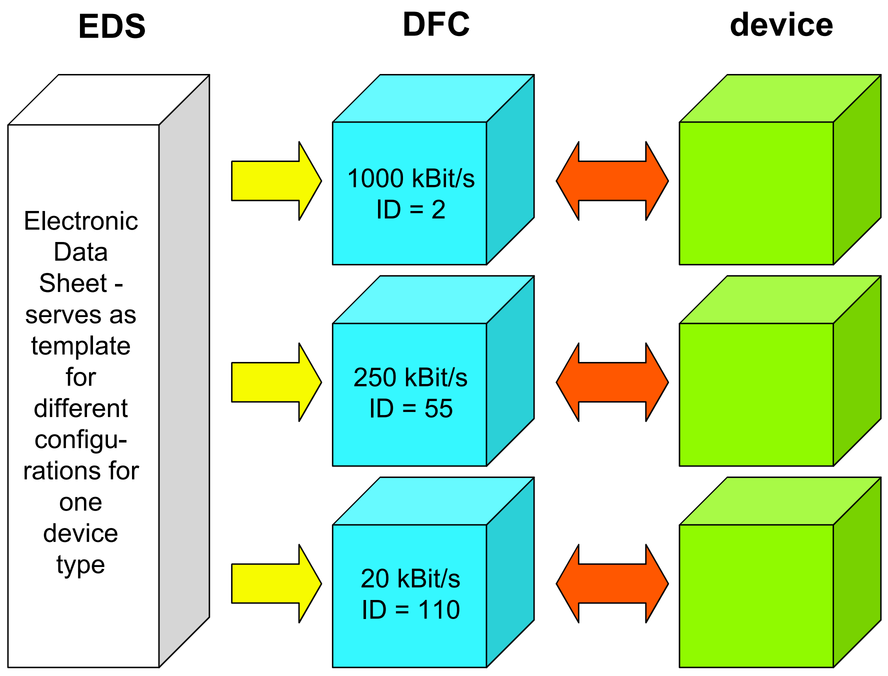
EDS
DFC
device
Electronic Data Sheet - serves as template for different configu-rations for one device type
1000 kBit/s ID = 2
250 kBit/s ID = 55
20 kBit/s ID = 110

Figure 1: EDS structure

An EDS is supplied by the vendor of a particular device. If a vendor provides no EDS 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 shall be aware, that the description is different from the concretely implemented features of that device, what may cause severe problems!

4.2 Basic structure

The files are ASCII-coded, the /ISO646/ character set shall be used.

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

The EDS contains several sections; each consists of a group of related entries. The sections and the entries shall be 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 shall 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 consists of any combination of letters and digits, and shall 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 shall apply to section names.

The value is a string, which is interpreted depending on the entry (*see* 4.3).

Comments and empty lines may be included in the EDS files. Each line of a comment shall start with a semicolon (;). It shall be in the leftmost column.

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

If not specified otherwise, all sections and entries in this paper are mandatory. In order to support future extensions additional sections and additional entries inside the sections may be included. The Conformance Test Tool is going to recognize these entries with warning messages.

4.3 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 shall be written as decimal numbers, hexadecimal numbers or octal numbers. Hexadecimal numbers are preceded by 0x. Octal numbers shall 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 has no value, this shall be 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 shall 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 5.3.1).

Example for octett string:

```
    DemoSeq=01a1053c45aabbccddeeff
```

Bitstrings shall be stored as a sequence of 0 and 1.

Example:

BitDemo=11001010000111

For entries of one of the integer types a formula may be used. This gives the chance to describe values that depend on other values. For example, the COB-IDs 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 is replaced by the actual node-ID of the device. The $NODEID shall appear at the beginning of the expression. Otherwise the line is interpreted as without a formula. It is only possible to use an offset to this node-ID in the formula. More complex expressions are not allowed.

4.4 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`	shall indicate the file name (according to OS restrictions),
`FileVersion`	shall indicate the actual file version (Unsigned8),
`FileRevision`	shall indicate the actual file revision (Unsigned8),
`EDSVersion`	shall indicate the version of the specification (3 characters) in the format “x.y”. If the entry is missing, this is equal to “3.0”. EDS files written according to this document shall use “4.0”.
`Description`	shall provide a file description (max 243 characters),
`CreationTime`	shall provide the file creation time (characters in format „hh:mm(AM|PM)“),
`CreationDate`	shall provide the date of file creation (characters in format „mm-dd-yyyy“),
`CreatedBy`	shall provide the name or a description of the file creator (max. 245 characters),
`ModificationTime`	shall provide the time of last modification (characters in format „hh:mm(AM|PM)“),
`ModificationDate`	shall provide the date of the last file modification (characters in format „mm-dd-yyyy“),
`ModifiedBy`	shall provide the name or a description of the creator (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
```

The optional section `[Tools]` may describe some aspects of the usage of the EDS by software packages. The specification of this section is done in /CiA405/.

4.5 General device information

The EDS shall contain device specific information about

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

This information shall be given in the section `[DeviceInfo]`.

The following keywords shall be used used:

`VendorName`	shall provide the vendor name (max. 244 characters)
`VendorNumber`	shall provide the unique vendor ID according to identity object sub-index 01_h (Unsigned32)
`ProductName`	shall provide the product name (max. 243 characters)
`ProductNumber`	shall provide the product code according to identity object sub-index 02_h (Unsigned32)
`RevisionNumber`	shall provide the product revision number according to identity object sub-index 03_h (Unsigned32)
`OrderCode`	shall provide the order code for this product (max. 245 characters)
`BaudRate_10`	shall indicate the supported baud rates (Boolean, 0 = not supported, 1=supported)
`BaudRate_20`	
`BaudRate_50`	

`BaudRate_125`
`BaudRate_250`
`BaudRate_500`
`BaudRate_800`
`BaudRate_1000`

`SimpleBootUpMaster`	shall indicate the simple boot-up master functionality (Boolean, 0 = not supported, 1 = supported),
`SimpleBootUpSlave`	shall indicate the simple boot-up slave functionality (Boolean, 0 = not supported, 1 = supported),
`Granularity`	shall provide 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 /CiA302/ this entry shall indicate the facility of dynamic variable generation. If the value is unequal to 0, the additional section `DynamicChannels` exists. Details are given in /CiA302/ and /CiA405/.
`GroupMessaging`	according to /CiA301/ Annex A this entry shall indicate the facility of multiplexed PDOs. (Boolean, 0 = not supported, 1 = supported) Details are given in /CiA301/.
`NrOfRXPDO`	shall indicate the number of supported receive PDOs. (Unsigned16)
`NrOfTXPDO`	shall indicate the number of supported transmit PDOs. (Unsigned16)
`LSS_Supported`	shall indicate if LSS functionality is supported (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
```

4.6 Object dictionary

4.6.1 General

In this logical part of the EDS the following information shall be provided:

- which objects of the object dictionary are supported
- limit values for parameters
- default values
- data types

In addition to that, there may be additional information provided.

The description of the objects shall be structurally divided into separate parts corresponding to:

- mandatory objects
- optional objects
- manufacturer specific objects

4.6.2 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 shall contain dummy entries for these first six bytes.

The indices from the data type area of the object dictionary shall be 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` shall be used for describing dummies. The entries shall be done according to the following 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 supports the mapping of the data types Integer8/16/32 and Unsigned8/16/32.

4.6.3 Object descriptions

4.6.3.1 Object lists

The object dictionary shall be structurally divided into three parts:

- `[MandatoryObjects]` shall only contain the mandatory objects. These are at least the objects 1000_h and 1001_h. For devices, that have implemented version 4.0 of CANopen, additionally the object 1018_h.
- `[OptionalObjects]` shall contain all other objects of the area 1000_h-$1FFF_h$ and 6000_H-$FFFF_h$.
- `[ManufacturerObjects]` shall contain all manufacturer specific objects (located in the area of 2000_h-$5FFF_h$).

Each of these sections shall contain a list of the supported objects which contains entries for the supported objects.

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

The entries are decimal numbered beginning with number 1. This way the last entry shall indicate 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
```

4.6.3.2 Object description

Each of the listed objects shall be described in an own section. The section names shall be all constructed following the same scheme. The section name shall be 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	shall indicate the number of sub-indexes available at this index (Unsigned8), not counting sub-index FF_h. This may be used for the description of sub-objects as defined below. This entry may be empty or missing, if no sub-objects exist.
ParameterName	shall provide the parameter name (up to 241 characters)
ObjectType	shall indicate the object code
DataType	shall indicate the index of the data type of the object in the object dictionary
LowLimit	shall indicate the lowest limit of the object value (only if applicable).

HighLimit	shall indicate the upper limit of the object value (only if applicable).
AccessType	for this object, represented by the following strings („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	shall indicate the default value for this object,
PDOMapping	shall indicate, if it is possible to map this object into a PDO (Boolean, 0 = not mappable, 1 = mappable).
ObjFlags	is an optional entry for assignment of special behaviour. *See* below.

The keyword ObjectType is optional. If the keyword ObjectType is missing, this is regarded as “ObjectType=0x7” (=VAR).

The following table gives an overview about the obligation of keywords. According to the object type the entries in object sections shall be mandatory (m), optional (o) or not supported (n). Missing ObjectType equals ObjectType VAR. Values in parentheses shall be regarded as replacement values, if the entry is missing. Empty parentheses ( ) are equal to “entry=”. For definition of ObjFlags and CompactSubObj see chapters below.

	DEFTYPE VAR	**DEFSTRUCT* ARRAY* RECORD***	**DEFSTRUCT** ARRAY** RECORD****	**DOMAIN**
ParameterName	m	m	m	m
ObjectType	o (VAR)	m	m	m
DataType	m	n	m	o (DOMAIN)
AccessType	m	n	m	o (rw)
DefaultValue	o ( )	n	o ( )	o ( )
PDOMapping	o (0)	n	o (0)	n
SubNumber	n	m	n***	n
LowLimit	o ( )	n	o ( )	n
HighLimit	o ( )	n	o ( )	n
ObjFlags	o (0)	o (0)	o (0)	o (0)
CompactSubObj	n	n ***	m	n

* without CompactSubObj or with zero CompactSubObj

** with non-zero CompactSubObj. *See* further remarks in 4.6.3.4.2.

*** not supported. Optional may be 0.

Example:

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

To describe a structured object (object has sub-indexes) each sub-index shall be described in an own section. The section name shall be 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 00_h shall always provide the highest sub-index implemented. In contrast, the EDS entry SubNumber shall contain the number of sub-indexes implemented, including the sub-index 00_h.

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
```

4.6.3.3 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 1010_h "Store parameters" will be written with either invalid values or at least in an invalid order. First the objects 1000_h up to $100F_h$ 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 shall be a boolean value (0=false, 1=true) for "Refuse write on download", the second bit shall be 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.

4.6.3.4 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.

4.6.3.4.1 PDO definitions

In principle the object descriptions for PDOs are all nearly the same. The most important information is the number of TPDOs and RPDOs which is given by /CiA301/. It is allowed to leave all PDO object descriptions. To mark this, the entry `CompactPDO` (Unsigned8) shall be added to the section `DeviceInfo`. It shall contain the implemented sub-indexes of the PDO communication parameter objects as a bitmask. The lowest bit shall define, if sub-index 01_h is implemented, the second bit, if sub-index 02_h is implemented and so on.

```
[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 *transmission type* 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 shall be described. It is explicitly allowed to combine implicit PDOs with explicit described PDOs:

NrOfRXPDO	Total number of RPDOs

Implicit PDOs = (Total number) – (listed PDOs)

Example:

```
[DeviceInfo]
NrOfRXPDO = 5
CompactPDO=0x3
;explicitly defined RX-PDOs
[1402]
...
[1403]
```

There exist 3 implicit compact PDOs with the indexes 1400$_h$, 1401$_h$ and 1404$_h$.

4.6.3.4.2 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` may be added. If this entry exists and contains a value not equal to 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 00$_h$ shall own the Name `NrOfObjects`
- the object types are assumed to be VAR
- the data type for all sub-indexes except 00$_h$ and FF$_h$ is indicated by the entry `DataType` of the main object. Sub-index 00$_h$ shall always own the data type unsigned8.
- the limits are assumed to be NONE
- the access type for all sub-indexes except 00$_h$ and FF$_h$ is indicated by the entry `AccessType` of the main object. The access type for sub-index 00$_h$ is assumed to be ReadOnly.
- the default values for all sub-indexes except 00$_h$ and FF$_h$ is indicated by the entry `DefaultValue` of the main object. The default value for sub-index 00$_h$ 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 shall be assumed, that the sub-index list does not contain any gaps.

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

It is possible to assign explicit names, if the default names are not useful enough. For this a list of according names may appear. The section name shall be given by `[xxxxName]` with xxxx as the Index. The entry `NrOfEntries` shall indicate the number of names in the list. The names shall be 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 shall be built upon the rule given above.

4.6.3.4.3 Network variables

In case of programmable devices according to /CiA302/ rsp. /CiA405/ the description of the dynamic network variable arrays shall not be 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.

4.6.4 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 shall be structured as follows:

```
[<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 shall be 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
```

4.6.5 Comments

Comments may be added to the EDS by using the `Comments` section. This section shall only provide 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 shall be decimal coded.

Example:

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

5 Device configuration file DCF

5.1 General

The device configuration file comprises all objects for a configured device. The device configuration file shall be structured the same way as the EDS for this device. There shall be some additional entries in order to describe the configured device.

5.2 File information section

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

5.3 Object sections

5.3.1 Parameter value in standard description

`ParameterValue` – shall indicate the object value (as defined by `ObjectType` and `DataType`)

Example:

```
; value for object 1006h (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 may 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 is taken from this file (character 242; according to OS restrictions)

Example:

```
; manufacturer specific object 5600h (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
```

5.3.2 Denotation

When using a DCF in a concrete application it is useful to assign application specific names to the objects. This may be done by simple 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.

5.3.3 Compact storage

Refer to compact storage of EDS in chapter 4.6.3.4. With the same background it is required to have the possibility of a compact storage format for the DCF files.

5.3.3.1 PDO definitions

If the entry `CompactPDO` of the section `DeviceInfo` exists and is not equal to 0, it shall be optionally allowed to write only the object and sub-object description for the concretely configured PDOs. All other PDOs shall be disabled.

5.3.3.2 Array values

If the entry `CompactSubObj` exists and contains a value not equal to 0, all the `ParameterValues` of the sub-objects shall be stored in an extra list. The section name shall be indicated by `[xxxxValue]` with *xxxx* representing the index. The entry `NrOfEntries` shall provide the number of entries in the list. The values shall be listed with using their sub-index as decimal entry name (range 1-254). All missing entries in the list shall own 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 shall be indicated by `[xxxxDenotation]` with *xxxx* as the index. The entry `NrOfEntries` shall provide the number of entries in the list. The values shall be listed with using their sub-index as decimal entry name (range 1-254). All missing entries in the list shall own a name according to the default rule or the `[xxxxName]` section as described in EDS chapter 4.6.3.4.2.

5.3.3.3 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 shall be stored in the standard format as given in 5.3.1. 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.

Device Commissioning

There shall be an additional section in the DCF named `DeviceComissioning`:

`NodeID`	shall indicate the device's address (Unsigned8)
`NodeName`	shall indicate the node name (max 246 characters)
`Baudrate`	shall indicate the device's baudrate (Unsigned16)
`NetNumber`	shall indicate the number of the network (Unsigned32)
`NetworkName`	shall indicate the name of the network (max 243 characters)
`CANopenManager`	shall indicate the device is the CANopen manager. (boolean, 1 = CANopen manager, 0 or missing = not the manager)
`LSS_SerialNumber`	shall indicate the 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

6.1 General

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 /CiA401/ this is the usage of sequential sub-indexes. For example, the first 8 bit digital input device creates the object 6000_h 01_h, the second creates object 6000_h 02_h and so forth. The first digital output device works on 6200_h 01_h. Additional there might be global objects that have to be created when at least one module of a specific type is connected. For example /CiA401/ defines the "Global_Interrupt_Enable" object 6423_h, 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 /CiA301/. There the multi device module uses the approach of shifting a copy of the object dictionary structure by an offset of 800_h. The first object dictionary starts with object 6000_h, the second with 6800_h and so on. Object 1000_h 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.

6.2 Electronic data sheet

6.2.1 Assignment of extension modules

The EDS of the bus coupler device contains a list of supported extension modules. For each type of extension module the features of the module shall be described in a module description (MD). The number of the supported MDs shall be stored in the EDS section `[SupportedModules]`.

`NrOfEntries`	Number of supported extension modules (Unsigned16)

6.2.2 PDOs

The pre-defined PDO mapping shall be used according to the appropriate profile. Each new sub-index shall 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 may be mapped. Further data may be mapped to additional PDOs. Since they are not pre-defined and since their COB-IDs shall be marked as invalid, their usage shall 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 shall 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 shall be valid up to the first not existing mapped object. The sub-index 00_h 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 6000_h 01_h up to 6000_h 08_h. The third TPDO for 6000_h 09_h up to 6000_h 10_h. Two 16 bit extension modules shall be added. Then the first TPDO will be valid. It's mapping contains the entries 6000_h 01_h up to 6000_h 04_h. The sub-index 00_h is shortened to 4_d. The third TPDO is invalid.

6.3 Module description

The module description shall be placed in sections of which names begin with `Mx`, where *x* shall be 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 shall provide some entries in the section `MxModuleInfo`:

`ProductName`	shall indicate the name of the product (max. 243 characters)
`ProductVersion`	shall indicate the version information (Unsigned8)
`ProductRevision`	shall indicate the version information (Unsigned8)
`OrderCode`	shall indicate the manufacturer specific order code (max. 245 characters)

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

`Lines`	shall indicate the number of comment lines (Unsigned16)
`Line<line number>`	shall indicate one line of comment (max. 248 characters). The number shall be decimal coded.

Example:

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

If required, the device shall 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`	shall indicate the number of fixed objects (Unsigned16)
`1=0x6423`	The number N shall represent a decimal value, starting with 1.
`2=`	
`N=...`	

The fixed objects shall be placed in sections

`MxFixedxxxx`.

xxxx shall indicate the hexadecimal index without leading 0x and without any further leading 0. The contents of those sections shall be the same as specified for object descriptions in chapter 4.6.3.2. Sub-objects shall be defined in the same way. The naming of the sub-object sections shall be

`MxFixedxxxxsubx`

xxxx shall indicate the hexadecimal index without leading 0x and `x` shall indicate 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]` shall contain a list of all objects that instantiate new sub-indexes.

`NrOfEntries`	shall indicate the number of extended objects (Unsigned16) followed by a decimal counted list starting with 1:

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

The objects shall be placed in sections `[MxSubExtxxxx]`. ***xxxx*** shall indicate 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 4.6.3.2 shall exist. Additional entries are:

`Count`	shall indicate the number of extended sub-indexes with this description that are created per module. The format shall be Unsigned8 [;Unsigned8]. If one or more sub-indexes shall be 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 shall be 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`	shall indicate an optional unsigned8 entry. If an array is filled up to the end, the next object may be used. For example, in /CiA401/ the object 6020_h addresses 128_d single input lines. Line 129_d is addressed with the next main Index 6021_h. The value of this entry defines the maximum sub-index after which the next object shall be used. If the value is 0 or the entry is missing, the next object shall not be used.

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

6.4 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 may be done by copying the section `[SupportedModules]` into the DCF and then creating a reference list in the section `[ConnectedModules]`:

`NrOfEntries`	shall indicate the number of connected modules (Unsigned16)
`1=3`	
`2=3`	
`x=...`	*x* shall indicate the decimal list counter, starting with 1. The unsigned16 value shall be a reference to the `[SupportedModules]` list. The corresponding module descriptions shall be done according to the rules of the EDS (Section names beginning with `Mx`, *see* 6.3).

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.

6.5 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
```