Object and sub-object

In CANopen and POWERLINK, an object defines an interface between the application, the network interface and other network participants.

Objects may (but need not to) be sub-divided into sub-objects. Sub-objects cannot be sub-divided any further and may only be of type VARIABLE.

Object

Each object consists of a set of attributes:

Attribute	Description
Index	A 16 bit identifier of the object (given in hex)
Name	The name of the object
Object type	Type of the object (can be VARIABLE, ARRAY, or RECORD)
Data type	Data type of the object (can be a static or complex data type)
Value range	Value range that is allowed on this object
Access	Access type of the object (const, ro, wo, rw)
PDO mapping	Determines whether the object is mappable
Default value	The default (i.e. unconfigured) value of the object
Actual value	The current value of the object

Sub-object

Each sub-object consists of a set of attributes:

Attribute	Description
Sub-index	An 8 bit identifier of the sub-object (given in hex)
Name	The name of the sub-object
Object type	Type of the sub-object (must be VARIABLE)
Data type	Data type of the sub-object (must be a static data type)
Value range	Value range that is allowed on this sub-object
Access	Access type of the sub-object (const, ro, wo, rw)
PDO mapping	Determines whether the sub-object is mappable
Default value	The default (i.e. unconfigured) value of the sub-object
Actual value	The current value of the sub-object

Attributes

The following section describes the attributes in more details.

Index

The object index is the unique identifier of the object for any kind of operation. It has to be within one of the ranges outlined here.
The index is given as a 16 bit unsigned integer.

Sub-index

A sub-object is addressed via its object index plus an according 8 bit sub-index.
The sub-indexes 00h and FFh have special formats. Sub-index 00h is always of data type UNSIGNED8 and contains the number of array/record entries. Sub-index FFh is always of data type UNSIGNED32 and describes the structure of the object.

Name

An object shall consist of a name that makes it easier for the user to identify its purpose. For objects implemented according to a standard (e.g. EPSG 301 Communication profile specification), the correct naming is defined by the specification.
The name is given as a string.

It is of good practice to group objects by prefixes and state their data type by suffixes, both of which are separated via underscores "_" from the other parts of the name.

Object type

This attribute determines the type of object.
It is given as an integer value with the following meaning:

Object type	Description
7	VARIABLE (single value, no sub-objects)
8	ARRAY (array of values of the same data type, entries are given as sub-objects)
9	RECORD (structure of values of different data types, entries are given as sub-objects)

Data type

This attribute determines the data type of the object. It is usually given as a reference to the data type definition area of the **Object Dictionary** and therefore a 16 bit unsigned integer.

The most commonly used data types are listed below:

Data type	Description
0001h	BOOLEAN
0002h	INTEGER8
0003h	INTEGER16
0004h	INTEGER32
0015h	INTEGER64
0005h	UNSIGNED8
0006h	UNSIGNED16
0007h	UNSIGNED32
001Bh	UNSIGNED64
0008h	REAL32
0009h	VSTRING
000Ah	OSTRING
000Fh	DOMAIN

The data type is mandatory for objects/sub-objects of type VARIABLE. It shall be given on objects of type ARRAY (here, it shall be stated with the object itself and is mandatory to be equal on all sub-objects, except sub-objects 00h and FFh). And it might be given on objects of type RECORD, if the RECORD structure is also defined (e.g. in a specification)

Value range

The value range of an object is generally given by the boundaries of the data type. Any further restriction might be specified.

Example:
An object of type UNSIGNED8 might be restricted to the values of 0..10.

Access type

This attribute determines the access from the network side to the object.
It can be of the following values:

Access	Description
const	The object's value is pre-defined and cannot change during runtime. The object can only be read from network and node side.

ro	The object is read-only from the network side. The node itself can change the value during runtime
wo	The object is write-only from the network side. The node itself can only read the value during runtime
rw	The object is read-write from the network side. Also the node itself can read and write a value from/to the object.

PDO mapping

This attribute defines whether an object can be mapped to a PDO or not.
It can be of the following values:

PDO mapping	Description
no	The object cannot be mapped to a PDO.
default	The object is mapped by default to a PDO. (The mapping must be specified in an according mapping parameter!)
RPDO	The object may be mapped to an RPDO.
TPDO	The object may be mapped to a TPDO.
optional	The object may be mapped to either RPDO or TPDO.

The actually possible mapping configuration depends also on the object's access type. That means that an object with access type read-only can only be mapped to a TPDO, even if PDO mapping specifies optional.

Note: Any object SHALL always be accessible via SDO!

Default value

The default value determines the value of the object when it is unconfigured. This is usually the case when a device is rebooted. If an object is storable, the default value is only restored after issuing the RestoreDefaultParameters command.

An object might not have a default value when such does not make sense, e.g. an object containing the actual position of a drive cannot have a default value.

Actual value

From the network side, the actual value of the object is retrieved via SDO read command or via mapping the object to a TPDO. The actual value of the object can be set via SDO write command or via mapping the object to an RPDO.

Sitemap | Top