AUCOTEC
Create Synergy – Connect Processes

Engineering Base

Formula Attributes

December 2019

AUCOTEC AG
Hannoversche Str. 105
D- 30916 Isernhagen
Phone:+49 (0)511 61 03-0
Fax: +49 (0)511 61 40 74

www.aucotec.com

AUCOTEC, INC.
2701 Troy Center Drive,
Suite 440
Troy, MI 48084
Phone: +1 630 485 5600
Fax: +1 248 655 7800

Content

1 What are formula attributes?

Formula attributes are individually definable formulas you can use to access system information and object attributes. Their purpose is to represent attributes of "pertinent" objects in compact form. For this purpose the access to the attributes of other objects is controlled via the relation of the source object.

To create formula attributes, you need good knowledge of the EB object structure as well as programming know-how.

Formula attributes can be used in dialog masks, for shape editing, in graphics and in worksheets.

A formula attribute enables:

- Navigation within the object structure.
- Access to attributes and read out of the values.
- Extraction of substrings from a text value.
- Creating conditional queries.
- Mathematical operations.
- Access to version and environment information.

The task:

"Represent the devices connected to a wire with their reference designator, material number and associated functions (in brackets)"

can e.g. be realized with formula attributes.

1.1 Definition of the formulas

The total string of a formula attribute is composed of interconnected or nested access descriptions.

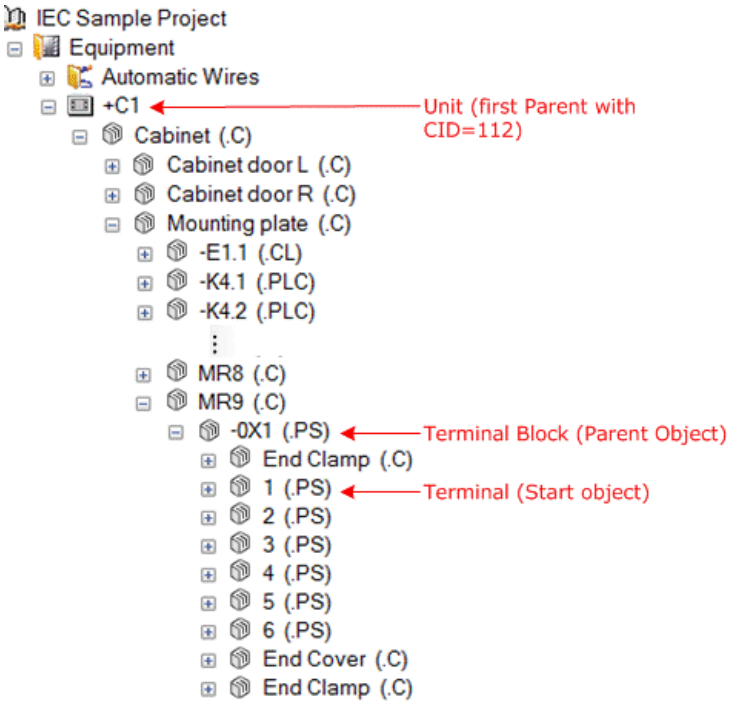
A simplified version of an access description is structured as follows:

KeyIdentifierVariables;

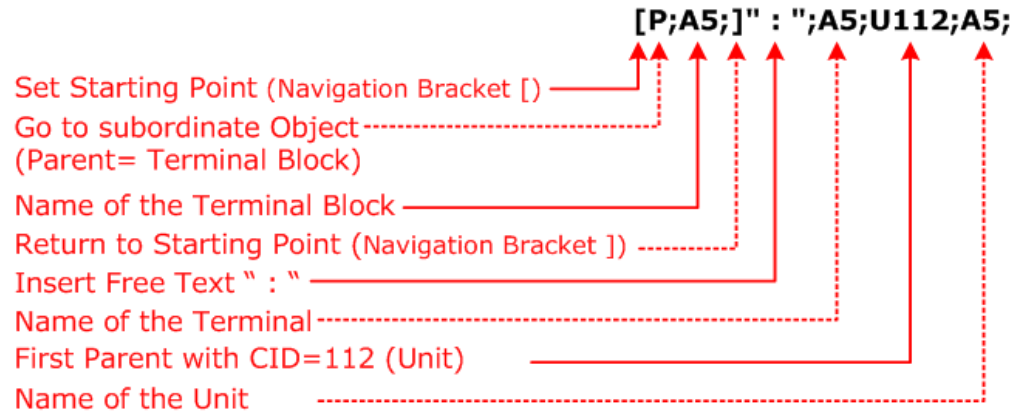
The key identification controls the kind of access to attributes, attribute information or system information.

Example:

Create a formula attribute at a terminal that gives out the name of the terminal block, the name of the terminal and the unit.



Formula used:



Result of the formula: -0X1 : 1+C1

The following must be heeded when creating a formula:

- A formula may consist of several instructions.
- The individual parts of the formula must always be separated by a semicolon (;).
- The length of the formula is limited to 1,000 characters.
- At the end of the formula, a semicolon (;) is mandatory.
- A key identification consists in most cases of one or two letters.
- Formula attributes cannot be used recursively.

- The evaluation of a formula attribute is aborted if it is impossible to navigate to a requested object.
- For parameters and variables, small/capital letters must be observed.
- Folder objects are not taken into account in the search for parent objects. This holds in particular for the navigation steps counted.

Formula attributes must by all means be created in a test environment and tested for their applicability! We strongly advise you against creating and testing the formula attributes in the productive working environment!

1.2 Basics in the creation of formula attributes

For each object, all of the information (e.g. device designation, associated function, etc.) is stored in attributes. The objects are assigned to certain global types (e.g. project type, device type, cable type, etc.) and subordinate types (connector group, generator, motor, etc.) using single attributes.

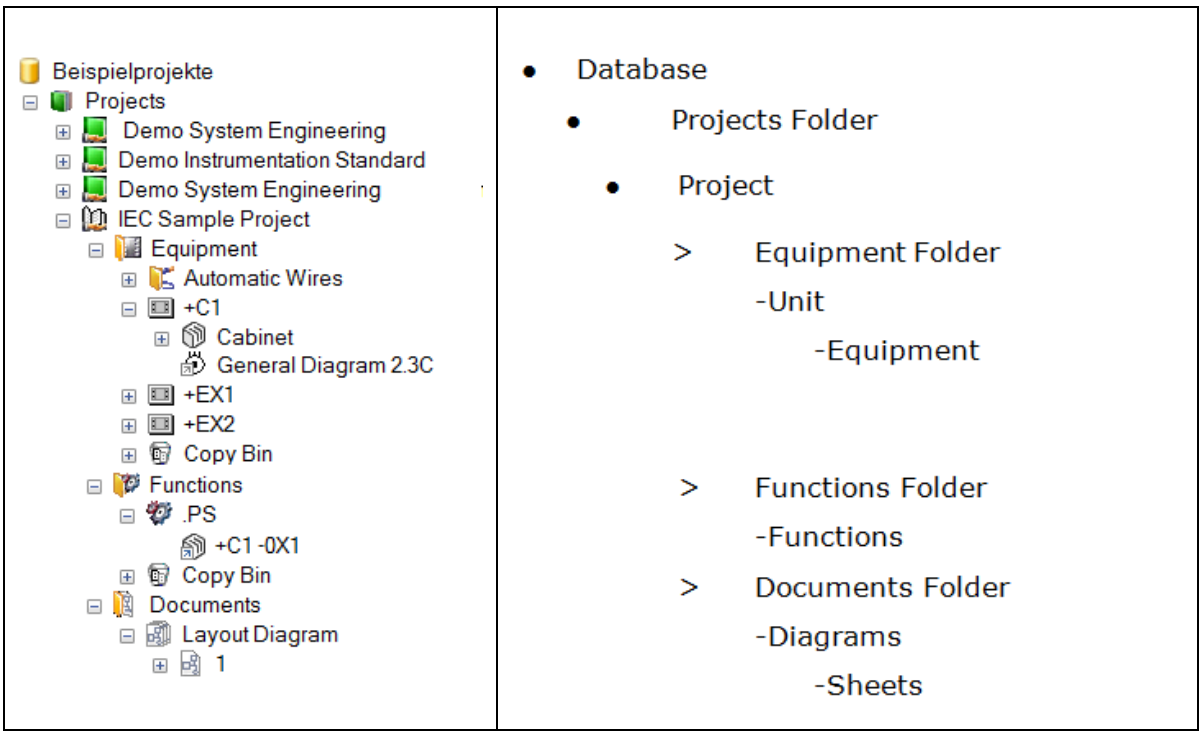
In the Engineering Base database, the objects are stored in a hierarchical structure.

This structure makes it possible to access the attributes of all objects within a database that have a relation with the source object. System information (e.g. EB version date, etc.) can likewise be shown.

Using the hierarchical structure, you can deliberately access the attributes of superordinate (parent) or subordinate objects (child).

Using the global type definition, the cover ID (CID) and the type definition (TID) you can selectively access the attributes of certain object types.

Setup of the object hierarchy:



Relationships between the objects:

- **Child**: Object that is subordinate to the source object (for example, devices are children of the unit).
- **Parent**: Object superordinate to the source object (for example, the unit is a parent object of devices).

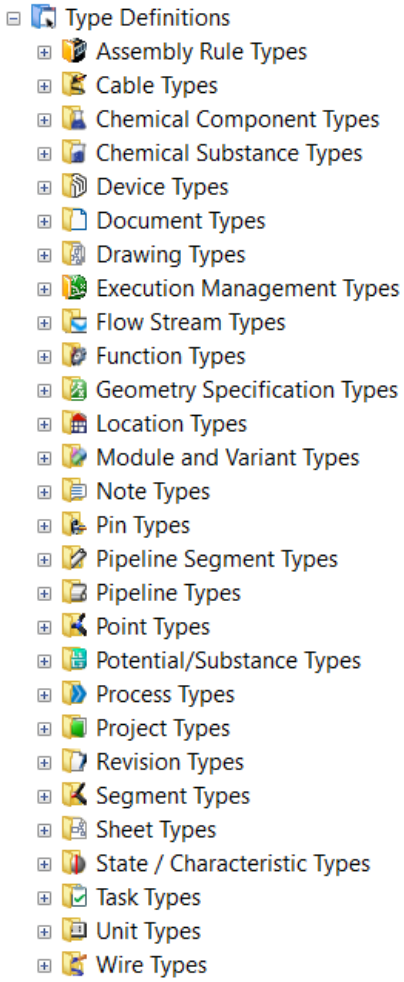
Associations between objects are relations via roles (for example, the relation between pin and wire). Relations have a starting point and an end point, that is a direction.

When accessing associated or superordinate objects, "forward" and "backward" navigation is possible. Multiple results can be recorded.

When an associated or superordinate object is accessed, then this becomes the "current" object, and all subsequent attribute functions are carried out with this object.

Global type definition (CID):

Specific attributes are assigned to each object. The assignment of the attributes is controlled by the global type definition (CID).



One cover ID (CID) is assigned for each global type definition.

Examples of CIDs:

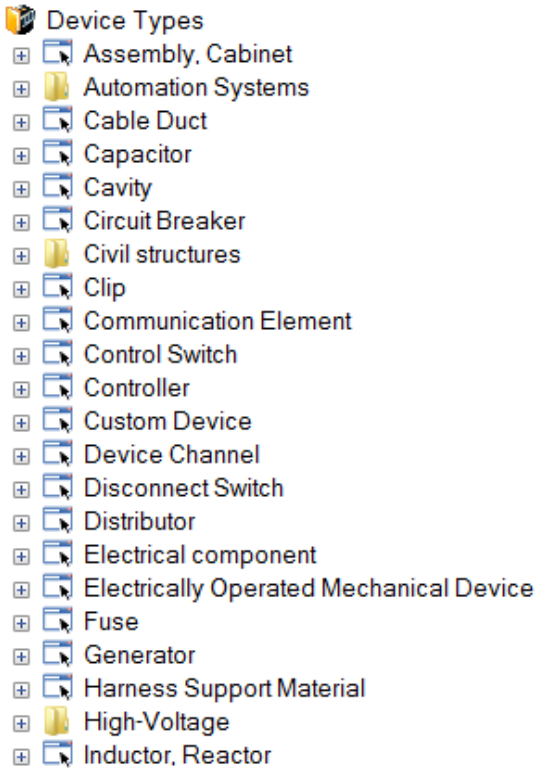
Equipment	7
Project	10
Unit	112
Devices	113

A list of all applicable CIDs is provided by the EB WebHelp in the chapter References.

For internal purposes Engineering Base uses additional CIDs apart from the above-mentioned type definitions.

Type definition (TID):

The global object types (CID) are further subdivided into object types (TID).



Object types of the global object type device types (CID 113)

One type ID (TID) is assigned for each object type.

Examples of TIDs:

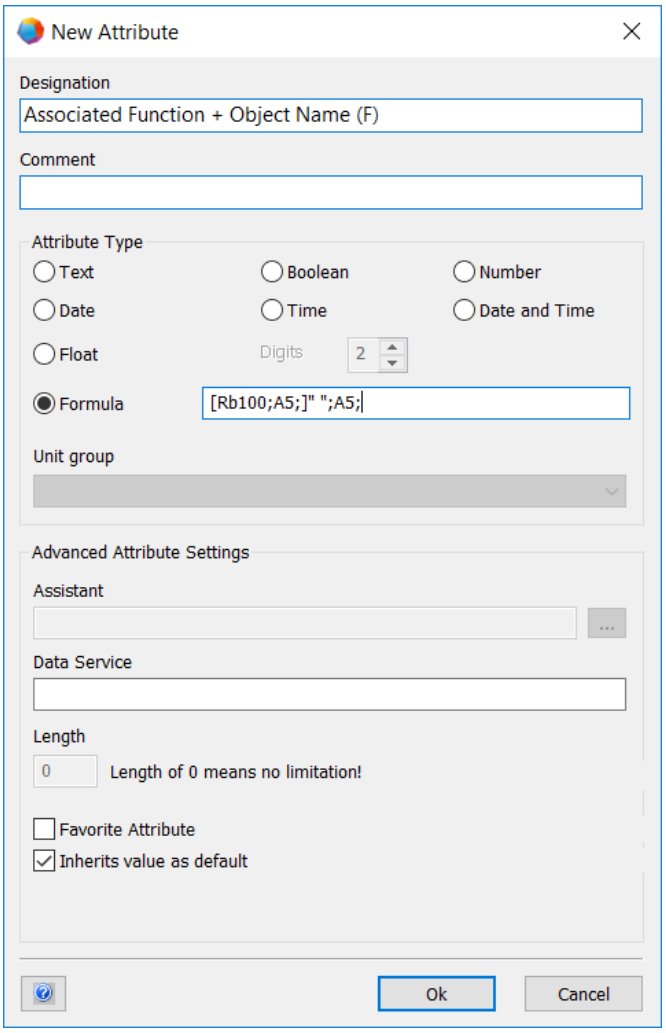
Motor	123
Terminal	136
PE	141

A list of all applicable TIDs is provided by the EB WebHelp in the chapter References.

1.3 Where are formula attributes defined?

1. In Engineering Base open a **database for testing formula attributes** or create a new database for this purpose.
2. In the Engineering Base Explorer, click on **Database**.
3. Select the folder **Attributes**.
4. In the context menu, point at **New**, then click on **Attribute**.

   The dialog **New Attribute** is opened.



5. In the field **Designation**, enter the name of the new attribute. With CTRL+F3 you can select an entry from the DB dictionary or if required create a new one.
6. Mark **Formula** under **Attribute Type**.
7. Enter the desired formula in the adjacent field.
8. Click **OK** to create the new attribute.
9. Restart Engineering Base.
10. Test the formula attribute for all required use cases before taking it over in your productive Database.

The newly created formula attribute can now be used.

2 Definition of the formula attributes

The following detailed descriptions of formula attributes are based on the Engineering Base version 6.4.1. Older versions may lack some language elements.

Legend of the syntax description:

Bold type	Literals (key identifications, variables and mandatory characters of an instruction (**;**, **{**, **}**, …).
Italics	User entries are recorded in italics.
< *term* >	A term in angle brackets is used as placeholder for a number of different values whose meaning is later on explained in detail.
[ ]	Encloses the optional parts of an instruction.
|	Separator (exclusive OR) between alternative entries.

2.1 Integration of system information

2.1.1 Version information

Key identification **V**	**V**ersion
Syntax: **V** [**v**] [**b**]**;**	
V;	Engineering Base **V**ersion number. **B**uild number.
Vv;	Engineering Base **V**ersion number.
Vb;	**B**uild number of the Engineering Base version.

2.1.2 Environment information

Key identification **Eu**	**E**nvironment **U**ser information
Syntax: **Eu** [**f** | **s** | **n** | **c** | <AccountSelData>]**;**	
Euf;	**F**ull Name - user name (default).
Eus;	**S**hort Name – short user name
Eun;	Windows login **N**ame
Euc;	**C**omputer name.
Extended user account data can be displayed with <**AccountSelData**>.	
Eu0;	NameFullyQualifiedDN Name, unambiguously fully qualified Examples CN=Paula Mustermann, OU=User, OU=AUCOTEC, DC=aucotec,DC=com.
Eu1;	NameSamCompatible Account name, example AUCOTEC/PMu.
Eu2;	NameDisplay Name, example Paula Mustermann.
Eu5;	NameUniqueId Unambiguous identifier of the name, example {edf3accf-47d2-4d5f-9d49-efea0f4ea8a18}.
Eu6;	NameCanonical Canonical name, example aucotec.com/OU-Users/AUCOTEC_DE/User/OU-GS/Paula Mustermann.
Eu7;	NameUserPrincipal Example PMu@aucotec.com.
Eu8;	NameCanonicalEx Extended canonical name, example aucotec.com/OU-Users/AUCOTEC_DE/User/OU-GS/ Paula Mustermann.

2.1.2.1 Date and time

Key identification **Ed**	**E**nvironment **d**ate and time	
Syntax: **Ed** [**d** | **D** | **t**] ["<time format specification>"]**;**		
d	Short version of the date based on the current settings under control panel / region and language.	
D	Long version of the date based on the current settings.	
t	Time based on the current settings.	
Time format specification	**%x**	Date in the format of the current country settings, example DD.MM.YYYY.
	%a	Day of the week in short form, e.g. We for Wednesday.
	%A	Day of the week.
	%d	Day as decimal number (DD, 01-31)
	%j	Day as decimal number (001-366)
	%w	Day of the week as decimal number (0-6, Sunday =0).
	%b	Month in short form, example Aug for August.
	%B	Month.
	%m	Month as decimal number (01-12).
	%y	Year in short form, example 14 for 2014.
	%Y	Year.
	%U	Week number (00-53, Sunday first day of the week).
	%W	Week number (00-53, Monday first day of the week).
	%c	Date and time (DD.MM.YYY hh.mm.ss)
	%H	Hours in the 24-hour format (00-23).
	%h	Hours in the 12-hour format (01-12).
	%M	Minutes as decimal number (0-59).
	%S	Seconds as decimal number (0-59)
	%X	Time in the format of the current country settings, example hh.mm.ss.
	%z, %Z	Time zone, for example Central European Time.
	%%	Percent character.

Example:	
Formula:	Value of the formula attribute (character string):
Ed"%x**";**	Date in the format of the current country settings, example DD.MM.YY.
Edt"%X**";**	Time in the format of the current country settings, example hh.mm.ss.

2.1.2.2 Environment Text

Key identification **Et**	**E**nvironment **T**ext. Controls the sequence of the formula attribute output if it consists of more than one result. Thus e.g. values from the object can be hierarchically determined in upward direction yet be displayed in the other direction (top down) without having to repeatedly go through the hierarchy.
Syntax: **Et l | r;**	
Etl;	The output is from right to left.
Etr;	The output is from left to right (default).

2.1.2.3 Environment Language

Key identification **El**	Suppression of the translation (**L**anguage). Switches off the translation of contained translate elements for the complete formula. Contained references are output in the form <l:ttt-dd>, e.g. <0:100-24> , where l = Selection of the language via translate setting of the project ttt = Text number -dd = Dictionary ID; (can be absent with global reference).
Syntax: **El;**	

2.2 Integration of texts

<table>
<tr><td>Key identification " "</td><td colspan="2">Entering of free text (character strings) or translate references</td></tr>
<tr><td colspan="3">Syntax:
"<free text> | [<translate references>] | [\<control characters>] ";</td></tr>
<tr><td rowspan="3">"free text";</td><td colspan="2">Any character string may be entered between the inverted commas and is inserted into the result of the formula attribute.</td></tr>
<tr><td colspan="2">Translate references can be inserted via the 'display format' <0:4711>.</td></tr>
<tr><td>\control characters</td><td>Control characters are predefined special characters or direct character codes that influence the output. Currently the input of the character codes is only supported in hexadecimal form (max. 4 places, leading zeros permitted, corresponds to the Windows character codes)</td></tr>
</table>

Examples of formulas:	
Formula:	Value of the formula attribute (character string):
"*Test Text* **<***0:100***>";**	"Test text device list (structured)".
"*Test Text******0A**";**	"Test text" with subsequent carriage return.

2.3 Integration of attributes

To be able to integrate attributes, the attribute ID (aid) of the desired attribute must be known.

You can display the attribute ID in the object properties, the attribute properties or have it shown in the list view of all attributes.

<table>
<tr><td>Key identification A</td><td colspan="3">Simple attribute access (scalar attributes)</td></tr>
<tr><td colspan="4">Syntax:
A [<modif>] ["<prefix>"] < aid > ["<postfix>"]
[([<numchar> [:<poschar>]] [[prefix F] "<fieldformat>"])];</td></tr>
<tr><td rowspan="8">modif</td><td colspan="3">Selector for special predefined attributes</td></tr>
<tr><td>F</td><td colspan="2">AF ["<prefix>"] aid ["<postfix>"] (<value>);
Test whether the specified attribute flags (manual, automatic, ...) are set.
Result:
1 or prefix if the flag is set.
0 or postfix if the flag is not set.</td></tr>
<tr><td></td><td>Value</td><td>ID number of the flag
2 = Attribute is read only
4 = Attribute value is empty
256 = manually entered value
1024 = Attribute value from the catalog
2048 = Translate text exists
8192 = Attribute is write protected</td></tr>
<tr><td>j</td><td colspan="2">Ajaid;
Access to project attributes; the project object becomes the current object</td></tr>
<tr><td>J</td><td colspan="2">AJaid;
Access to project attributes; the current object remains unchanged.</td></tr>
<tr><td>r</td><td colspan="2">Ar;
Classical reference designator of the object. (with part of..).</td></tr>
<tr><td>R</td><td colspan="2">AR;
Reference designator of the object with path output (with folder).</td></tr>
<tr><td>o or 1</td><td colspan="2">Ao; or A1;
Object ID of the object.</td></tr>
<tr><td>c or 2</td><td colspan="2">Ac; or A2;
Cover ID of the object.</td></tr>
</table>

	t	**At;** Type ID of the object.
	C	**AC***aid***;** Comment attribute.
Prefix / Postfix	Character string that is inserted before or after the attribute value if the latter is not empty.	
aid	Attribute reference (numerical or predefined C, r, R).	
numchar	Number of characters to be shown of the value determined for the formula attribute (>0 start from the beginning; <0 start from the end).	
poschar	Position of the character from which to start.	
Field format	Specifies the precise length of the character string including substitute characters to be inserted. For the prefix F = "-", "e" und "f" the meaning is different (see below).	
Prefix F	Prefix for field format.	
	-'0	Leading zeros are deleted.
	-	The characters in "Field format" are eliminated from the value of the formula attribute.
	e	"Field format" is used only if the value of the formula attributes is not empty.
	f	"Field format" is interpreted as c-conforming format string. Possible output formatting is described in the annex.

Examples of formulas with attributes:	
Formula:	Value of the formula attribute (character string):
A*"Präfix"aid"Postfix"***;**	Prefix/Postfix = character string that is inserted before or after the attribute value if the latter is not empty.
A*aid***(***number***);**	Display of a specified number of characters (>0 start from the beginning; <0 start from the end).
A*aid***(***n:x***);**	Display of characters starting from position x.
A*aid***("***0000000000***")**	Display of the attribute value, filling up to 10 characters with "0".
A*aid***(***-n-***'0"--------");**	Value of the attribute, use last n characters, delete leading zeros and fill up to 8 characters with "-".
A*aid***(***n-"Value"***);**	Value of the attribute, use first n characters, delete characters corresponding to "Value" ("Wert").
A*aid***(f***"%8.2f"***);**	Output of the attribute value formatted according to C syntax, here floating-point number with 8 pre-decimal and 2 decimal places.

The attribute with the ID= 5 contains the object name, it be 00123456 in this case.	
A*5***("***0000000000***");** Value of the attribute with the ID = 5, and fill up to 10 characters with "0".	Value of the attribute with the ID = 5, and fill up to 10 characters with "0". „0000123456“
A*10212***;"-"; A***10386***;"-";** **A***10175***;**	Show values of the listed attributes separated by "-". Diagram type-sheet format-drawn by
A*5***(***4***-'0');**	Value of the attribute with the ID = 5, use first 4 characters and delete leading zeros. "12".
A*5***(***4***-'0"--------");**	Value of the attribute with the ID = 5, use first 4 characters, delete leading zeros and fill up to 8 characters with "-". „------12“
A*5***(-***4***-'0"--------");**	Value of the attribute with the ID = 5, use last 4 characters, delete leading zeros and fill up to 8 characters with "-". "----3456"
A*5***(***6***-"***0012***");**	Value of the attribute with the ID = 5, use first 6 characters and delete character string "0012".. "34"
A*5***(***3***:***5***);**	Value of the attribute with the ID = 5, and 3 characters are shown from position 5 on. „456“
AF"*manual***"***5***(***256***);**	"manual" if the attribute with the ID = 5 was entered manually.
A"("*102875***")"(***f***"%.2f");**	Attribute 102875 is to be a floating-point number with the value -47.1256 "(-47.13)" The value is rounded to 2 decimal places.
A*102875***(***f***"%0 10.3f");**	"-000047.136" The value is shown as 10-digit floating-point number, leading zeros are inserted.
A*102675***(***f***"%.6d");**	The attribute 102675 is to be an integer with the value Wert 4711. "004711" The result is to be six-digit, therefore leading zeros are inserted.

2.4 Access to higher-order objects (aggregations)

Information from a higher-order object (parent) can be processed for an object.

2.4.1 Parent objects

Key identification **P**	Access to **P**arent Objects
Syntax: **P** [**j**]**;**	
P;	Access to direct parent object.
Pj;	Access to the project object.

Examples of formulas with access to parent objects:	
Formula:	Value of the formula attribute (character string):
P;A*5***;**	Shows the name of the parent object.
Pj;**A***5***(***2***)**;	Shows the first 2 characters of the project name.
Pj;Ar;	Indicates the reference designator of the project.

2.4.2 Bottom Up

Key identification **U**	Bottom **U**p - access to first parent object of a CoverID / type (bottom-up), that is to the first higher-order object with a defined ID (TID or CID).
Syntax: **U** [**t**] [<id>] [(<aid> **:** ("<valuetext>" | <aidvalue>))]**;**	
t	Selection via type (and not CID).
id	Definition of the relevant parent object (CID or TID).
aid	Additional check for attribute value.
valuetext	Test value (text) in inverted commas "test value". .
aidvalue	Attribute ID whose value at the start object determines the test value.

Examples of formulas with access to first parent object:	
Formula:	Value of the formula attribute (character string):
U*cid***;**	Definition of the parent object with the corresponding CID.
Ut*tid***;**	Definition of the relevant parent object with the corresponding type ID.
U*113***;A***5***;**	Shows the name of the first device above the current object (CID device =113).
Ut*123***;A***5***;**	Shows the name of the first motor above the current object (type ID motor =123).

2.4.3 Mostup Parent

Key identification **M**	Access to the **M**ostUp parent object of a CoverID / type (bottom-up), that is to the uppermost higher-order object with the specified CoverID. **M** returns the current object if it meets the condition and if there is no higher-up object meeting the condition.
Syntax: **M** [**t**] <**id**>**;**	
t	Selection via type (and not CID).
id	Definition of the relevant parent object (CID or TID).

Examples of formulas with access to uppermost higher-order parent object:	
Formula:	Value of the formula attribute (character string):
M*cid***;**	Access to the uppermost parent object with the specified CID. There are no limits as to which object types may be present in the parent chain (example: M223 accesses the uppermost parent object of all cables).
Mt*tid***;**	Access to the uppermost parent object with the specified type ID (example: Mt1101 accesses the uppermost parent object of the type multi-core cable).
M*113***;A***5***;**	Shows the name of the uppermost device above the current object (CID device =113).
Mt*123***;A***5***;**	Shows the name of the uppermost motor above the current object (type ID motor =123).

2.4.4 Top Down

Key identification **T**	**T**op down navigation - navigates upward or downward from the reference object by a specified number of parent objects.
Syntax: **T** [**s**] <counter> [**:** <cid>]**;**	
s	Within the aggregation chain, any object types may be present. Default: The hierarchic chain may be structured only by folder elements, otherwise the navigation is aborted.
counter	Number of steps for the navigation depth. (>0 upward; <0 downward)
cid	Relevant object type

Examples of formulas with access via top-down navigation:	
Formula:	Value of the formula attribute (character string):
T*n***;**	Navigates upwards to the parent object on the n-th level.
T*-n***;**	Navigates downwards to the parent object and from there down to the n-th level.
T*n:cid***;**	Navigation in the hierarchic chain of the parent objects (with the specified CID) is upwards to the parent object on the n-th level.
Ts*n:cid***;**	Navigation in the hierarchic chain of the parent objects (with the specified CID) is upwards to the parent object on the n-th hierarchic level. Within the hierarchic chain, any object types may be present.
T*1***:***111***;A***5***;**	Shows the name (A5) of the folder (CID 111 = folder) on the 1st hierarchic level (1).

2.5 Navigation aids

2.5.1 Object Navigation

Key identification **O**	**O**bject Navigation - navigation code Here you can mark an object (object reference) for further editing. Depending on the key identification used, the stored object reference is used as starting point after an invalid navigation. There is only one navigation memory!
Syntax: **O** [**c** | **C** | **m** | **s** | **S** | **j** | **J** | **I** | **L** | **g** | **GE**]**;**	
c	Delete marking of the restart point (globally).
C	Delete marking of the restart point within a multiple treatment.
m	Marks restart point after a multiple treatment.
s	Sets a navigation point, the multiple treatment is interrupted. Return to this point with Oj.
S	Sets a navigation point. A multiple treatment is not interrupted.
j	**J**ump to the navigation point specified with Os. The multiple treatment is interrupted.
J	**J**ump to the navigation point specified with Os. A multiple treatment is not interrupted.
I	Sets a jump marking (for **L**oops).
L	Delete the jump marking.
g	Go to jump marking (OI) (corresponds to "goto"). The processing is continued at the OI marking.
GE	The evaluation of the complete formula is terminated ("**g**oto **e**nd").

2.5.2 Object navigation with brackets

The square brackets must be set in pairs 'on the same level'. They cover a complete specification of conditions or are completely within a condition. An opening **before** the condition and closing **within** the condition are not possible.

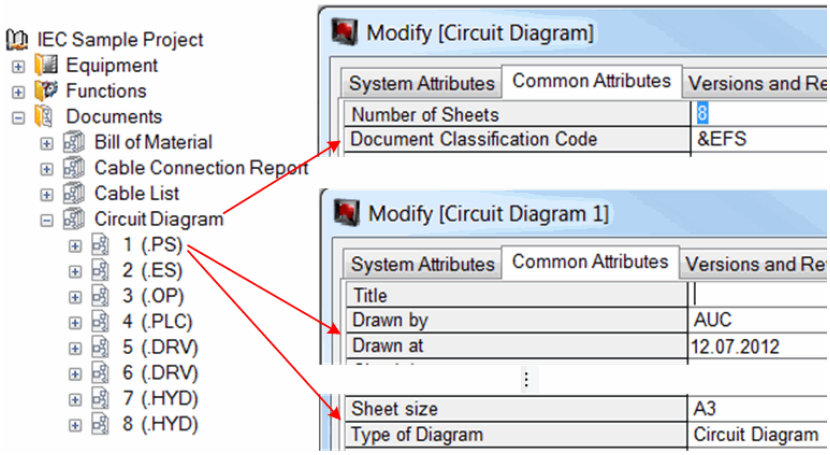
The brackets function analogously to Os … Oj (set navigation point and return). An active multiple treatment is as a matter of course **not** aborted at this point.

Key identification **[ ]**	Navigation brackets Treatment of the navigation object by 'enclosing in parentheses', i.e. when the evaluation reaches the closing bracket "**]**", then the navigation object is reset to the initial value, and the statement is further evaluated.
Syntax: **[** [**+**] <statement sequence> **];**	
+	Continue execution at the end of the brackets even if object navigation within the brackets fails.

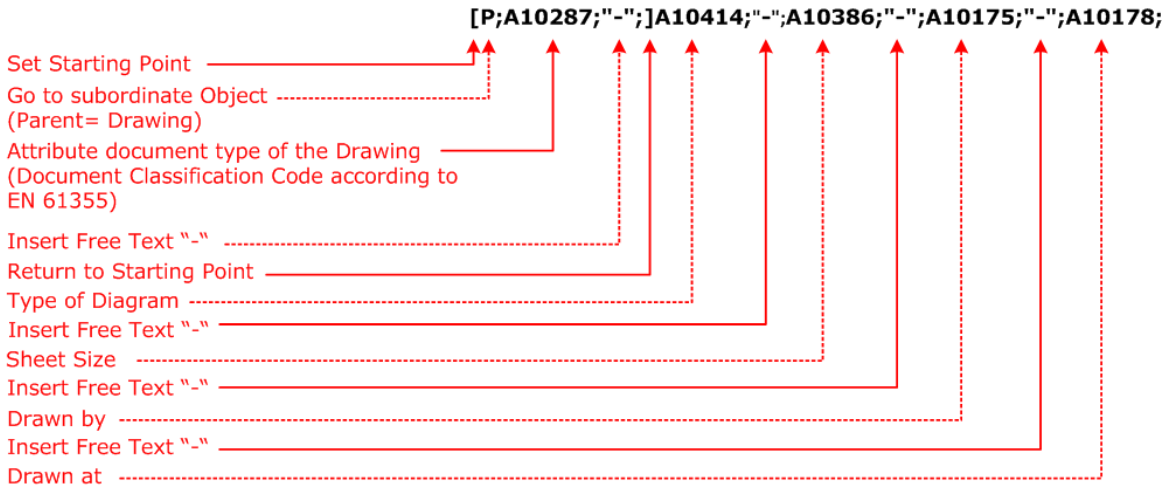
Examples of object navigation:	
Formula:	Value of the formula attribute (character string):
Rb*100***;[P;A***5***;]A***5***;**	Navigation to the function of an object. Navigation is carried out within the brackets to the parent of the function, and the corresponding designation (name =attribute ID=5) is output. Then the name (attribute ID=5) of the function is output.

Exercise:

A composite sheet number should be created in a sheet, which is made of Document Classification Code according to EN 61355 and the attributes Type of Diagram, Sheet size, Drawn by and Drawn at, separated by "-". The attribute Type of Diagram of the parent diagram contains the Document Classification Code.



Formula used:



Result of the formula: &EFS-Circuit Diagram-A3-AUC-12.07.2012

2.6 Access to associated objects (association)

Access to information for an object associated with the current object.

An association points from one object (source) to possibly many other targets. For example a unit can be assigned to several sheets but not vice versa. In this case the unit is the source and the sheets are the target of the association. Association types differ by their tasks.

A list of all applicable roles is provided by the EB WebHelp in chapter References.

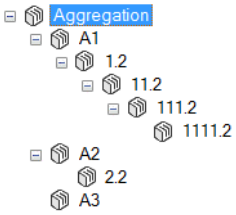
Example of association roles

Role	ID
Function to device	100
Function to sheet	106
Project to catalog	40
Object to child object	-3
Substitute roles for multiple treatment of core and cable destinations. **These substitute roles are available only in formula attributes.**	
Collective role for 108/109 core destinations	1104
Collective role for 110 cable destinations	1105
Substitute role for 110 cable destinations on the left	1106
Substitute role for 110 cable destinations on the right	1107

2.6.1 Relation Role

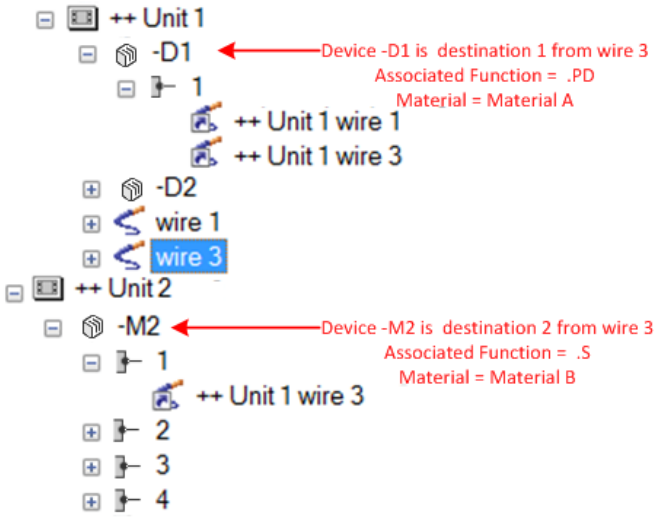
Key identification **R**	Access via **R**ole - access via target associations (**R**) or source association (**Rb**).
Syntax: **R** [**b**] [**cs**] [**t**] ([**#**] | [**s**]) (**<role>** | **O**) [**(<depth>)**] [**:** [**f** |**F**] **<assocflag>**] [**:** **<cid>**] [**: +d** | **+f** **<calculation>** **"<formatstring>"**] [**" <text> "**]**;**	
b	Access to associated object via source association (**b**ase)
t	Access to associated object via target associations with restriction to the type, <cid> represents the object type (tid) here.
cs	Access to 'siblings' (**c**ollect **s**ibling).
#	Indicates the number of associated objects. The navigation object is kept.
s	Sorting (according to standard sorting logic of EB).
role	Association roll.
O	List of the elements of an active object group (see object groups GO).
depth	In connection with cs: Navigation to sibling object (<0 to predecessor; >0 to successor).
assocflag	Relations having (**:f**) or not having (**:F**) the specified relation flags. Specified relation flags: 1 = the association contains a conflict 2 = manually assigned association 32 = the association is read-only 64 = the association is negative (special contextual meaning)
cid	Restriction to the objects with the specified cid.
+d	Calculation with integer arithmetic (long)
+f	Calculation with decimal number arithmetic (double).
formatstring	Output string with formatting information. Possible output formatting is described in the annex.
text	Text to be used for separating multiple results.

Syntax of <calculation>: (**<aid>** [**<charpos>**] | **#**) [[***** | **+** | **-** | **/**] [**p**] **<aid>** | **#**]	
aid	Attribute ID.
charpos	Start index when reading in numbers from text attributes (0 based).
#	In connection with the # operator the number of objects.
*** + - /**	Mathematical operators.

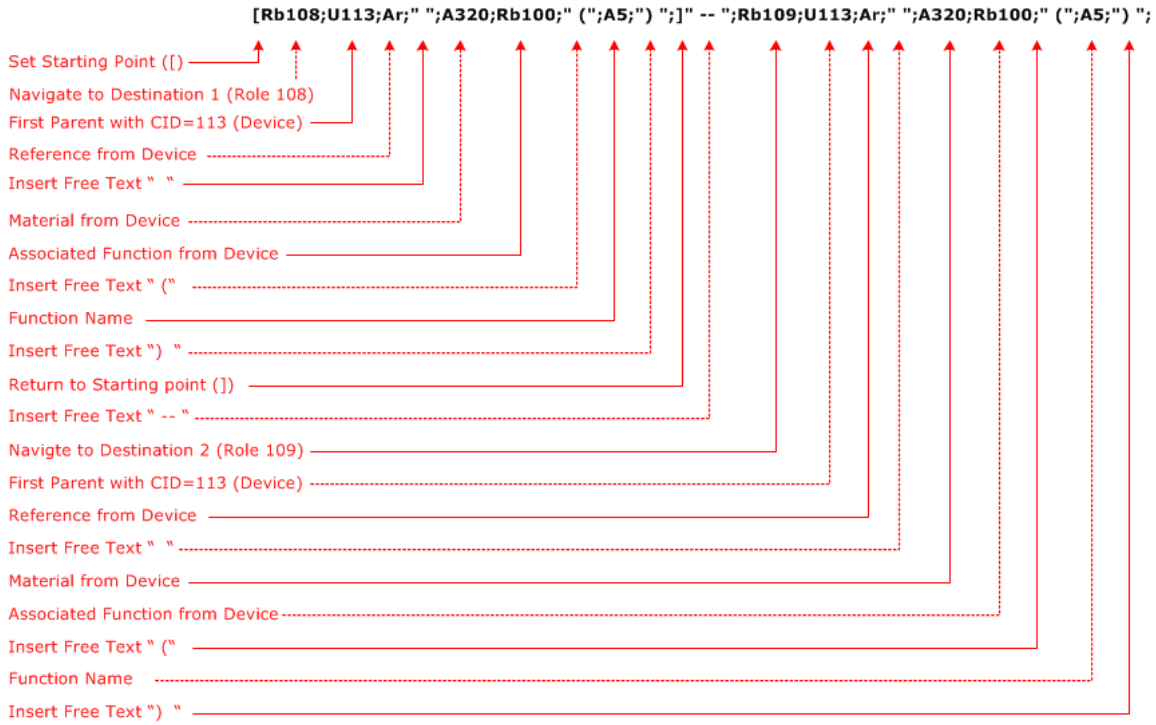
Examples of formulas:	
Formula:	Value of the formula attribute (character string):
R-*3***;**	Navigation to child objects (roll -3).
R*108***;**	Navigates from pin to wire (role 108).
Rt*108***:***904***;**	Navigates from pin to wire (role 108), restricted to PE (TID = 904).
R*100***" / ";A***5***;**	At a function: Specifies the names of all devices associated with the function (roll 100), separated by " / ".
R*100***:***223***" / ";Ar;**	At a function: Specifies the long names (full) of all cables associated with the function (roll 100 = all devices with restriction at CID=223), separated by " / ".
R*100***:***223***" / ";RM***1***:***2***;A***5***;**	At a function: Specifies the names of all cables associated with the function, separated by " / ". If more than 1 cable is associated with the function, then only 2 cables are to be shown.
Rb*100***;A***5***;**	At a device: Specifies the name of the associated function.
Rb*100***;A***5***;"(";A***25***;")";**	At a device: Function name and comment for the associated function.
R-*3***" / ";A***5***;**	Lists all aggregated objects (child objects), separated by " / ".  Result of the formula: A1 / A2 / A3

Exercise:

Create a representation of the devices connected to a wire with its reference name, material number and the assigned function (in parentheses).



Function used:



Result of the formula:
++Unit 1 -D1 Material A (.PD) -- ++Unit 2 -M2 Material B (.S)

2.6.2 Relation Multi

Key identification **RM**	Treatment of multiple targets (**R**elation **M**ulti).
Syntax: **RM** [**p** | **s**] <**counter**> [**:** <**modcounter**>] [**"** <**text**> **"**]**;**	
p	Specifies that the entered <text> is to become the **P**refix upon reaching the limit (counter).
s	Specifies that the entered <text> is to become the **S**uffix upon reaching the limit (counter).
counter	Defines the limit from which the special treatment is to start. When the number of objects reaches the value <counter>, then the number is changed to <modcounter>. <modcounter=0> has a special function: • The prefix/suffix becomes the result string. • The 'current' object is set to 'nothing'. • The number of previous multiple objects is reduced to 0. Thus the further evaluation of the subsequent statements is aborted and only continued at a later jump flag.
modcounter	Defines the modified number of objects to be dealt with. <modcounter> must be lower than <counter>.
text	Text that is to become the prefix or suffix.

Examples:	
…**;RMp**m"Text**";...**	Modification of the display of the associated objects found. If the number is greater than or equal to m, then the generated lines get the text prefixed (p=prefix).
...;RMm**:**n**;...**	Restriction of the display of the associated objects found. If the number is greater than or equal to m, then only n objects are shown. If the number is smaller than m, then all objects are shown.

2.6.3 Relation object groups

You can use an object group to restrict a formula evaluation to a given number of objects. The specification of the number of objects cannot be achieved by simple navigation. You can use an object group e.g. to prevent an object from being treated repeatedly.

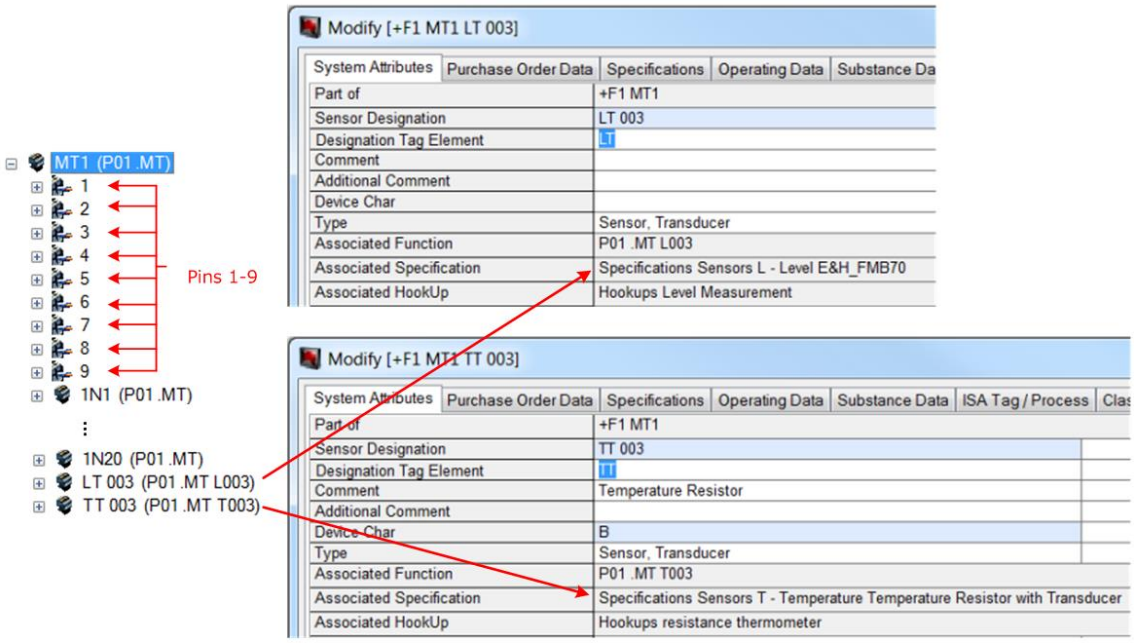
There is only one object group per formula attribute.

When adding objects to the group, you can optionally specify an attribute whose value is checked for matching before the object is added to the group. The same holds for deleting objects from the group. The first object that is added to the object group by means of its attribute ID defines the condition. An object group may contain elements that were added in different ways, i.e. with or without checking a condition.

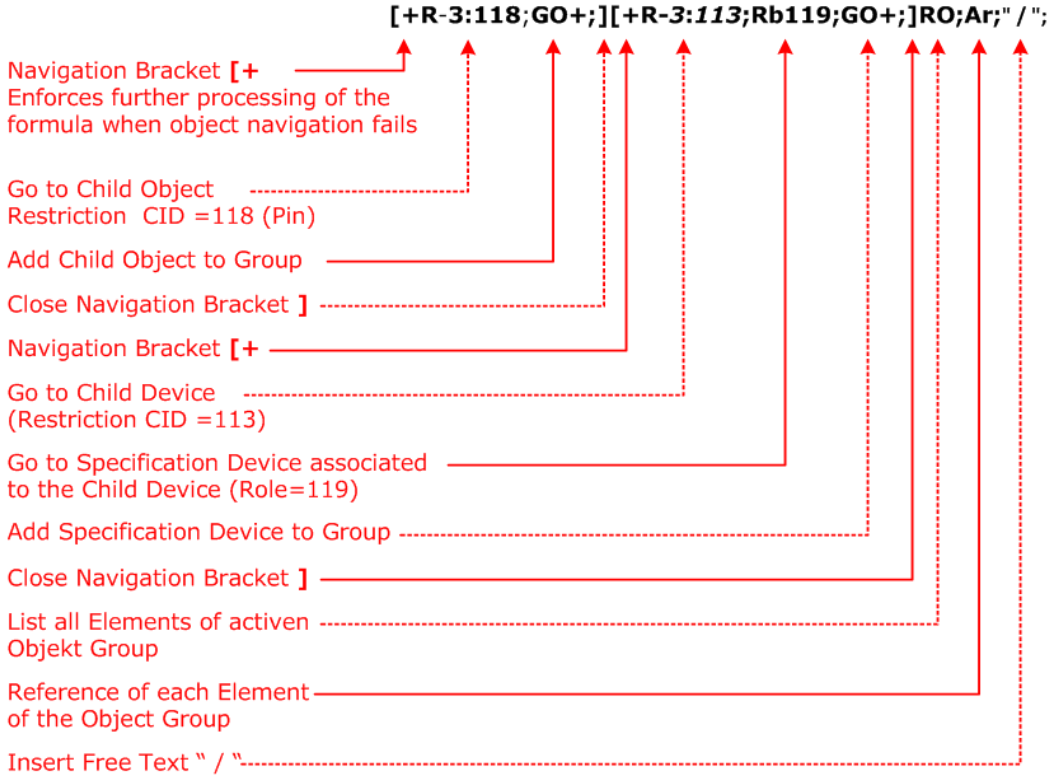
Key identification **GO**	Treatment of object groups (**G**rouped **O**bjects)
Syntax: ***GO*** [**+** [***A***<*id*>]] | [**-** [***A***<*id*>]] | ***t*** | ***c;***	
GO+A*id*	The first object of a group defines the condition and is itself added to the group.
GO-A*id*	The first object of a group defines the condition and is itself not added to the group.
+	Add object after checking the condition. The first object of the group specifies the condition.
-	Delete object.
t	Check whether the object is already contained in the group. The test returns a Boolean value (0 or 1).
c	Delete group.

Exercise:

There should be created a list that contains all pins and specification devices separated by "/" (CID 118 = Pin, CID 113= Device, Role 119 = Specification).



Formula used:



Result of the formula:

+F1 MT1 2 / +F1 MT1 8 / +F1 MT1 4 / +F1 MT1 1 / +F1 MT1 3 / +F1 MT1 6 / +F1 MT1 7 / +F1 MT1 9 / +F1 MT1 5 / Specifications Sensors T - Temperature Temperature Resistor with Transducer / Specifications Sensors L - Level E&H_FMB70 /

2.7 Integration of mathematical operations

When using attribute values in mathematical operations, please note that unusable attribute values are used with the value 1 as default. You can use parameters to specify that in certain cases the value 0 is returned.

You can carry out mathematical operations only with attributes from the current object. In order to use data from other objects, you must store them intermediately in registers (**r**) and recall them as needed (**R**). Using the registers, you can then for the ongoing calculation include values from other objects in the same formula.

Key identification **=**	Mathematical operations / operators		
Syntax: **=** [[**r** | **R**] <**RegSel**>] (**d** | **f** | **u**) <**expression**> ["<**formatstring**> "]**;**			
r RegSel	The calculated value is stored in a mathematical register (intermediate memory). Register: Values 1 – 5 are possible.		
R RegSel	The value of a register (1 – 5) is read. Register: Values 1 – 5 are possible.		
d	Specification of the integer calculation (**D**ouble)		
f	Specification of the floating point calculation (**F**loat)		
u	Definition of the floating point operation including related unit (**U**nit). You are recommended, to use only mathematical operators and the squre root. If only one operand has a unit, then this one is used. If both operands have units, then the one stated first will be used allocating units from left to right.		
expression	(**+** | **-** | ***** | **/** | **&** | **|** | **^** |***Q*** | ***S*** | **s** | **xy** | **exp**) <**argument**> [<argument>]		
	+, -, *, /	Mathematical operators	
	&, |, ^	Bitwise operators	
	Q	Square Root	
	S	Sine (radiand)	
	s	Sine (degree)	
	xy	Power	
	exp	Exponent of base *e*	
	argument: <**attribute**> | <**constant**> | <**expression**> | <**relation**>		
	attribute	**A** [**n**] [**e**] [**f**] <**aid**> [**:<charpos>**]	
		n	If the attribute at the object is missing, then 0 is returned instead of 1.
		e	If the attribute at the object is empty, then 0 is returned instead of 1.

		f	If no evaluable numerical component is present in the attribute for the object, then 0 is returned instead of 1.
		charpos	Position in the character string from which to start.
	constant	**K**<**value**> | **pi** | *e* Value is a fixed sequence of numbers. A decimal point must be used for floating point values. Thousands separators are not allowed. *e* = Euler's number	
	expression	See above.	
	relation	**#**[**P**] <**role**>**;** Returns the number of objects that are associated with each other (and are found via the role). **P**: The role is calculated starting from the parent. **role**: Relation identification **A list of all applicable roles is provided by the EB WebHelp in chapter References.** >0: Forward relation direction >0: Backward relation direction	
formatstring	Output string with formatting information. Possible output formatting is described in the annex.		

Examples of formulas with mathematical operations:	
Formula:	Value of the formula attribute (character string):
=#-*3***;**	Returns the number of child objects assigned to the current object.
=#*107***;**	Number of sheets assigned to the unit (roll 107 = UnitToDiagram).
=#*-107***;**	Number of units assigned to the sheet.
=f+A*1234***A***5678***;**	Adds the values of the attributes with the IDs 1234 and 5678. Representation as floating-point number
=u+A*1234***A***5678***;**	Adds the values of the attributs with ID 1234 and 5678. Example: A1234 = 3 cm A5678= 4 m The value of the formula attribute is 403 cm. A1234= 4,0 m A5678=3 cm The value of the formula attribute is 4,0 m.

=d*A1234**A**5678**;**	Multiplies the values of the attributes with the IDs 1234 and 5678. Representation as integer.
=f*A1234**K**1.2**;**	Multiplies the values of the attributes with the IDs 1234 and 5678 by the constant 1.2.
=fxyA1234**K**3**;**	Raises the attribute (base) to the power of the constant 3 (exponent)
=fxyA1**A**2**;**	Raises attribute A1 (base) to the power of attribute A2 (exponent)
=fexpA1**;**	Defines the exponential of attribute A1 (exponent) of base *e*
=d*A10061**#P**126**;**	Determination of the amount of a material for a hook-up. The amount of the material is multiplied by the use of the hook-up.
=d*A1234**#-**3**;**	Multiplication of the attribute value with the number of child objects.
=r3**d*A**1234**#-3;**	Multiplication of the attribute value with the number of child objects. The result is stored in register 3.
=d*R3**K**2**;**	Multiply the content of register 3 by 2.
=d+*A123**K**3**A**456**;**	Corresponds to the mathematical formula. (A123*3)+A456
=d*K2**+A**123**A**456**;**	Corresponds to the mathematical formula. 2*(A123+A456)
=d*Ane123**A**456**;**	Corresponds to the mathematical formula. A123*A456 Or 0*A456 if A123 is not present at the object or empty.
=r1**d*K**2**K**2**;=r**2**d+K**3**K**3**;=d+R**1**R**2**;**	The result of the multiplication 2*2 (=4) is stored in register 1. The result of the addition 3+3 (=6) is stored in register 2. The contents of registers 1 and 2 are added. Result: 10

2.8 Object check

Key identification **VO**	Verify object for a formula (**V**erify **O**bject) If the test result is negative, the formula is terminated.
Syntax: **VO s** | **S** | **d** | **D** | **a** | **A** | <**cid**>**;**	
s	Check whether the current object of the formula attribute represents a symbol.
S	Check whether the current object of the formula attribute does not represent a symbol.
d	Check whether the current object of the formula attribute represents a device.
D	Check whether the current object of the formula attribute does not represent a device.
a	Check whether the current object of the formula attribute represents an SAO (symbol without AO).
A	Check whether the current object of the formula attribute does not represent an SAO (symbol without AO).
cid	Check whether the current object of the formula attribute corresponds to the specified cover ID.

2.9 Conditions

You can specify conditions for formula attributes. Conditions are indicated by braces **{}**. **Conditions are written in the manner of a *case* instruction.**

There are always several alternatives per condition. Conditions can be nested.

The formula attribute must contain a blank after the reference value for a condition!

Key identification **{ }**	Condition
Syntax: **{** [**j**] <**condition formula**> [**{** [**=** | **<** | **>** | **<>** | **:** ("<**value string**>" | <**aid**>)]$^{1+}$ **blank** <**execution formula**> **}**]$^{1+}$ **}**	
j	Following processing of the condition, jump to the object preceding the execution.
Condition formula	Formula string whose result yields the test string.
=, **<**, **>**, **<>**	Check for equality, inequality, larger or smaller. Numbers are also checked as string comparison.
Value string	The character string that is compared with the test string. Here it is also possible to access register contents.
aid	Alternative value of an attribute at the start object that is compared with the test string.
:""	This statement (without condition) specifies the Else case of the case instruction. Here you can specify what is to be done if none of the other conditions is met.
Execution formula	Instruction that is carried out if the test was successful. Any sequence of possible commands is permitted as execution formula. For example: - Text included in inverted commas (" ") - Attribute value - Formula - …..

Examples of formulas with conditions:

Comparison of 2 attribute values:

The attribute with the ID=245 contains the width and that with the ID = 246 the height of an object.

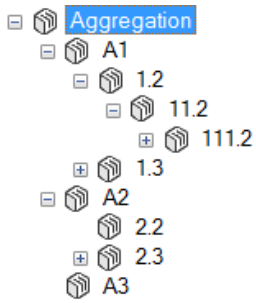
- **{A245;{=246 "equal";}{:"" "not equal";}};**
  Result for width = 20.00 mm and height = 20.00 mm "equal"
  Result for width = 20.00 mm and height = 30.00 mm "unequal"
- **{A245;{=246 "2 x ";A245;}{:"" =*A245A246; " mm";}};**
  Result for width = 20.00 mm and height = 20.00 mm: "2 x 20.00 mm"
  Result for width = 20.00 mm and height = 30.00 mm: "600.00 mm"
- **{=f+A245A246;{="40" "forty";}{:"" =f+A245A246; " mm";}};**
  If the sum of height and width is = 40, "forty" is to be output, otherwise the sum of width and height is to be shown as floating point number in "mm"

Check whether an attribute value meets a certain condition:

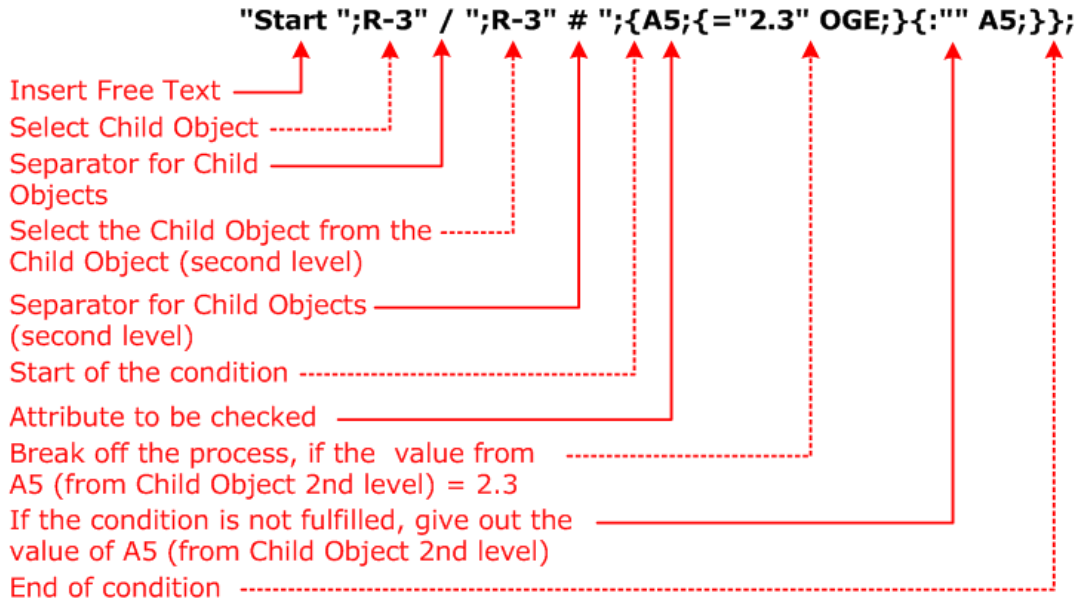
- **{AF245(256);{="1" "manual ";A245;}{:"" A245;}};**
  AF245(256) issues the value "1" if the attribute with the ID = 245 was entered manually.
  Result for manual entry of the attribute with the ID = 245: "manual 20.00 mm".
- **{A10025;{="1" A10001;}{:"" A10002;}};**
  Let the attribute with the ID=10025 be a Boolean one.
  If it has the value "1", then the value of the attribute with the ID=10001 is output, otherwise the value of the attribute with the ID=10002.
- **{A5;{="20"<"10">77 "within range";}{:"" "not within range";}};**
  If the value of the attribute with the ID=5 equals 20 or is smaller than 10 or larger than the value of the attribute with the ID=77 of the start object, then the result of the formula is "within range". If the condition is not met, then the message "not within range" is issued.
- **{A5;{="1"<"2" "<2";}{="10">"24" ">24";}{:"" "default";}};**
  If the value of the attribute with the ID = 5 is equal to 1 or smaller than 2, then "<2" is issued If the value is equal to 10 or larger than 24, ">24" is issued. If none of these conditions is met, then "default" is issued.

Exercise:

For each child object of the first level (A1, A2) all child objects of the second level (1.2, 1.3, 2.2, …) should be listed. The child objects of the first level should be separated with " / " those of the second level with " # ".Once the value of the child object of the second level reaches 2.3, the execution of the formula should be terminated.



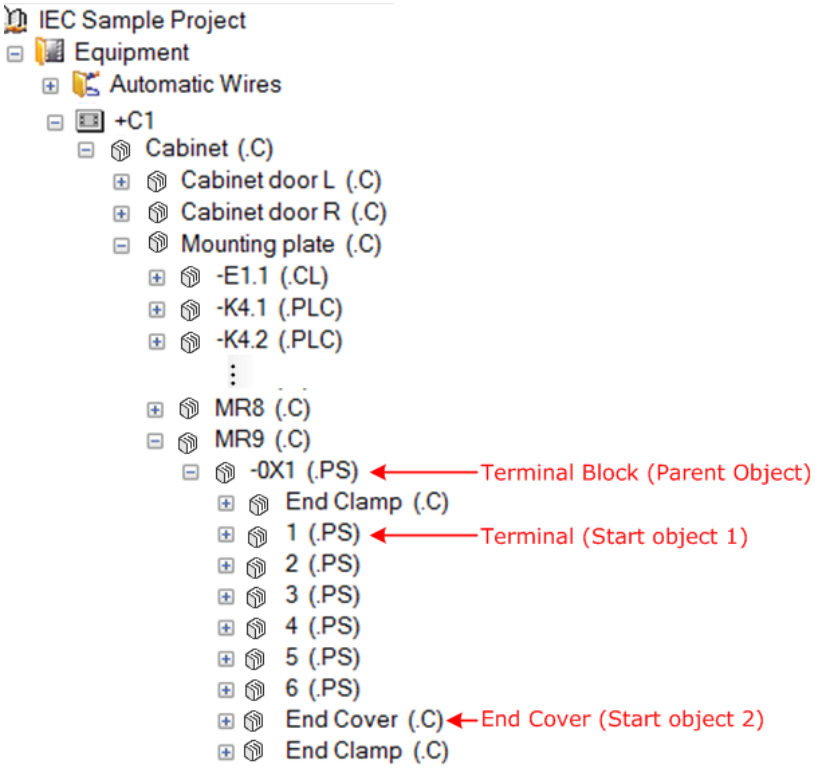
Formula used:



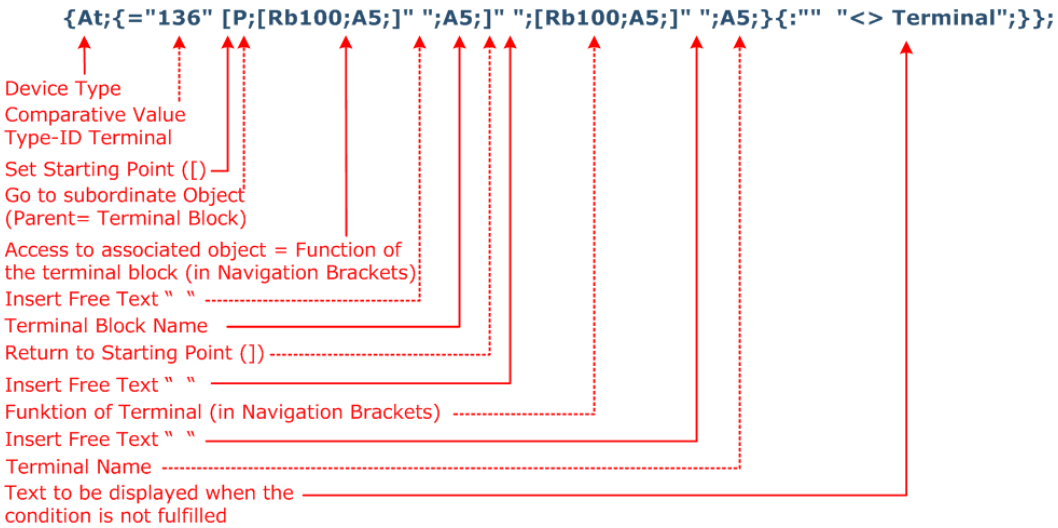
Result of the formula: START 1.2 # 1.3 / 2.2 #

Exercise:

If the equipment is a termial, the output should be the function of the terminal block, the terminal block, the function of the terminal and the terminal. Otherwise the output should be "<> Terminal".



Function used:



Result of the formula:

at Terminal (Start object 1): .PS -0X1 .PS 1

at End Cover (Start object 2): <>Terminal

2.10 Annex

2.10.1 Output formatting

The format specification consists of optional and mandatory fields.

Every single character of the format specification stands for a certain format option. The simplest form would be the percent character followed by a character (e.g. %s).

If the percent character is followed by a character that does not correspond to any format option, then the character is copied to the standard output.

Formatting the output		
Syntax: **%**[**flags**] [**width**] [**.precision**] **type**		
flags	**-**	Left alignment, the default value is right
	+	Enforces the sign (+ or -). Otherwise the sign is shown only for negative values.
	0	Fill up with leading zeros to the maximum field length. In combination with "-" there is no filling up of the field.
	" " (blank)	Prefixes positive output values with a blank. In combination with "+" no blank is prefixed.
	#	Together with type = f, a decimal point is set in any case. Is ignored with type = d.
width	Specifies the minimum length of the output in characters. If the output is shorter, then the output is complemented to the minimum length with blanks on the left or right, depending on the orientation. The length specification never causes a value to be shortened. All characters are output.	
.precision	Optional. Maximum number of characters or the minimum number of digits for integer values.	
type	**d**	Integer display (**D**ouble) The value specifies the minimum number of digits to be output. If the number is smaller, then leading zeros are prefixed.
	f	Floating point display (**F**loat) The value specifies the number of decimals. If a decimal point is output, then at least one digit is shown in front of it. The value is rounded according to the decimal places.