File: CVX_SPEC.DOC
Version: V2.1
Author: Uwe Rinn, SNI
Date: 02/14/00
History: 01/15/96, Uwe Rinn, V0.1 created
01/18/96, Uwe Rinn, V0.2
- added specification for value ordering of lines and maps

01/23/96, Uwe Rinn, V0.3
- added ASAP2 style type fields
- added extra line for optional axis descriptions of calibration maps
- removed description line for calibration value
- added types AXIS_PTS, ASCII and VAL_BLK
- added spec for axis point values, ascii strings and value blocks
- removed optional separators from examples
- removed optional fields in Excel examples

01/29/96, Uwe Rinn, V0.4
- fixed row/column mix-up in description of ordering
- fixed syntax for calibration header

03/15/96, Uwe Rinn, V0.5
- added syntax summary chapter
- added spec for X_AXIS_PTS, Y_AXIS_PTS, Z_AXIS_PTS, i.e. embedded axes

07/10/96, Uwe Rinn, V1.0
- relaxed syntax for record separators

16/08/99, Matthias Röck ETAS/PAM-P, V1.1
- support of verbal conversions (ASAP2 V1.21 keyword COMPU_VTAB)
- support of rescale values for axis points (ASAP2 V1.30 keyword AXIS_RESCALE)
- support of functions (ASAP2 V1.21 keyword FUNCTION)
- support of variant coding (ASAP2 V1.21 keyword VARIANT_CODING)

21/09/99, Falko Stuhler SIEMENS AT PT GS TT, V1.2
- support of Display Identifier (ASAP2 V1.31 keyword DISPLAY_IDENTIFIER)

19/10/99, redesignation V2.0 – no changes
14/02/00, Matthias Röck ETAS/PAM-P, V2.1
- Bugfix for string delimiter

Table of Contents:

Calibration Values Exchange Format

Overview

This document describes a format for the exchange of calibration values between application systems. A secondary goal of this format is to allow the data exchange with standard applications like word processing, spreadsheet or database applications.
Especially amongst those applications, the encoding proposed in this document (CSV) is widely supported as an export/import format.

CSV is an acronym for comma separated values format, meaning that a CSV transfer unit contains a sequence of lines and lines contain a sequence of values separated from each other by a special separator character. Values containing the separator character are surrounded with double quotes.

Values can be text or numbers (integers as well as floating point numbers). Text is a sequence of bytes (range from 32-255). The core part of the format defined in this document, will only make use of the characters in the range from 32-126 (printable ascii characters), but other optional parts may always contain characters from the entire range from 32-255. For their interpretation or correct rendering, bilateral agreements between exporting and importing systems/applications will be necessary.

Calibration values can be characteristic object values or axis point values. Calibration values are physical values represented by floating point numbers.

Meta Syntax

In the following, the syntactic structure of the calibration values exchange format is specified using EBNF (extended Backus Naur form).

```
::= production symbol
<>  delimit non terminal symbols
[]  optional - 0..1 occurrences
{}  0..n occurrences
|   separates alternatives
()  grouping of alternatives
'   delimiter for terminal values
;   end of rule
```

0 Transfer Unit Structure

A calibration values transfer unit is a sequence of records.

Records are separated from each other by a record separator. A record separator consists of one or more "empty" lines, i.e. these lines may contain value separators but must not contain any values.

```
<Record Separator> ::=
            <End of Line Separator> <Line Separator>
            { <End of Line Separator> <Line Separator> }
      |
            <End of Transfer Unit>;
<End of Line Separator> ::= {<Value Separator>} <New Line>;
```

There may optionally be a sequence of value separators at the end of a line. Exporting applications should not generate excess value separators, but some application, e.g. Excel, already do.

```
<Value Separator> ::= ','|';'|<TAB>; //as specified in the header
<TAB> ::= '\0x09';
<New Line> ::= <CR> <LF>;
<CR> ::= '\0x0D'; // carriage return control character
<LF> ::= '\0x0A'; // line feed control character
<Line Separator> ::= <End of Line Separator> | <End of Transfer Unit>;

<End of Transfer Unit>  // has no character representation, represents e.g.
                        // end of file
```

There are three types of records:

- the file header record type
- the comment record type
- the calibration values record type

The file header record must always be the first record in a transfer unit. It contains version information and information on the encoding of all subsequent records. The file header record may not appear anywhere else in a transfer unit.

A comment record may appear anywhere in a transfer unit after the file header and is not further interpreted.

```
<Calibration Values Transfer Unit> ::=
      <File Header>
      { <Record Separator> <Calibration Values Record> };
```

1 Header Elements

```
<File Header> ::=
      <Description Header>
      [<Function Header>
      <Variant Header>];
```

1.1 Description Header

```
<Description Header> ::=
      <File Content Identification> <Value Separator>
      [ <Decimal Point Separator> <Value Separator>
        <Comment Indicator> <Value Separator>
        <String Delimiter> <String Delimiter> <Value Separator> ]
      <End of Line Separator>;

<File Content Identification> ::= 'CALIBRATION VALUES V' <Version Number>;

<Version Number> ::= <Major Version Number> '.' <Minor Version Number>;
<Major Version Number> ::= <Natural Number>;
<Minor Version Number> ::= <Natural Number>;
```

The version number corresponding to the specification in this document is 2.0. The major version number should be increased for each new version of the specification resulting in transfer units, which are not backward compatible. The minor version number should be increased for each new version of the specification.

```
<Natural Number> ::= <Digit> {<Digit>};
<Comment Indicator> ::= {<Ascii character>}
<Decimal Point Separator> ::= ',' | '.'
```

Please note that the first character immediately following the file content identification determines the separator character to be used throughout the rest of a transfer unit. The character following the separator character determines the decimal point separator to be used throughout the rest of a transfer unit. The text following the decimal point separator determines the string to be used as the comment indicator throughout the rest of the transfer unit. The character following the comment indicator determines the character to enclose all ASCII strings. This character has to be stated two times without any <Value seperator> in between.

It should also be noted that the <Value Separator> and the <Decimal Point Separator> cannot have both the value ','. In this case an importing system would automatically assume the default decimal point separator.

Additionally the <Value Separator> and the <String Delimiter> cannot have both the same value.

The default for the decimal point separator is '.'.
The default for the comment indicator is '*'.
The default for the string delimiter is '"'.

Example - using ';' as a field separator and ',' as a decimal point character and '*' as a comment indicator and '"' (double quotes) as a string delimiter:

```
CALIBRATION VALUES V1.0;,;*; ""
```

1.2 Function Header

```
<Function Header> ::=
                  'FUNCTION_HDR' <Value Separator>
                  [<Anything>] <Value Separator> <New Line>
                  <Function Block Record>;

<Function Block Record> ::=
      {<Value Separator> <Function Name>} <End of Line Separator>;

<Function Name> ::= <Text>;         //any valid text string
<Text> ::= <String Delimiter> {<Ascii Character>} <String Delimiter>
```

The function header is optional. It is used to support a fast selection of calibration values on function level for a CVX file. Calibration values can be function coded only if there exists a function header.

Example - using ';' as a field separator and ',' as a decimal point character and '"' as a string delimiter:

```
---begin header---
FUNCTION_HDR;;
"Function1";"Function2";"Function3";"Function4"
---end header---
```

1.3 Variant Header

```
<Variant Header> ::=
                  'VARIANT_HDR' <Value Separator>
                  [<Anything>] <Value Separator> <New Line>
                  <Variant Block Record>;

<Variant Block Record> ::=
      {<Variant Criterion> <Value Separator>
      {<Criterion Value> <Value Separator>} <New Line> }
      <End of Line Separator>;

<Variant Criterion> ::= <Text>;
<Criterion Value> ::= <Text>;
<Text> ::= <String Delimiter> {<Ascii Character>} <String Delimiter>
```

Example - using ';' as a field separator and ',' as a decimal point character and '"' as a string delimiter:

```
The following variant criterion with the associated values exist:
      1. Car (Limousine, Cabrio, Kombi)
      2. Gear (Manual, Automatic)

---begin header---
VARIANT_HDR;;
"Car";"Limousine";"Cabrio";"Kombi"
"Gear";"Manual";"Automatic"
```

---end header---

A calibration an measurement system importing calibration values should check if there exist forbidden combinations of variant criterion values defined in the associated ASAP2 file and abort the import of such variant coded calibration values.

2 Calibration Values Records

A calibration values record associates a calibration identifier with physical values.

Syntactically, records are sequences of fields.
There are two types of fields:
- text fields
and
- floating point number fields

Floating point number fields contain a single floating point number conforming to the following Syntax:

```
<Floating Point Number> ::=
      [<Sign>] {<Digit>}
      [ <Decimal Point Character> <Digit> {<Digit>} ]
      [ ('e'|'E') [<Sign>] {<Digit>} ];
<Sign> ::= '+' | '-';
```

The syntax for floating point numbers was copied from the standard C library man pages for the scanf function. Passing strings conforming to this syntax to the scanf function should yield the corresponding internal float or double representation. Care should be taken about the decimal point character itself, which must be replaced by the appropriate system dependent decimal point character.

```
<Digit>     ::= '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
```

Text Fields contain arbitrary text, i.e. a sequence of characters.
Any field, that does not contain a floating point number is automatically a text field. The value of a text field containing a string enclosed in the string delimiter is the enclosed string.

```
<Text> ::=
      <String Delimiter> { <Ascii Character> } <String Delimiter>
```

Text fields identifying calibration data should conform to the following syntax:

```
<Calibration Identifier> ::=
            <Letter> { (<Letter>|<Digit>|'.'|<IndexSpec>) }

<Letter>    ::= 'a'-'z' | 'A'-'Z';
<IndexSpec> ::= '[' <Digit> {<Digit>} ']';
```

However, importing systems may simply perform a character by character comparison of calibration identifiers against identifiers stored in their local database and thus do not have to rely on the syntactical structure.

Calibration values records for 0, 1 or 2 dimensional calibration data all have the same syntactical structure. They start with a calibration header followed by the physical values and terminated by a record separator.

Only the calibration identifier and the physical values need to be extracted during import of a transfer unit into an application system. All

other parts may be skipped for import and filled with arbitrary text for export. Calibration values records are typed and depending on the value of the type field, the physical values can be found at fixed offset from the beginning of a record. Looking at transfer unit as a table, the calibration identifier is always found in column 2 of the first line of a record. Physical values are always located in column 3. The line offset from the beginning of the record however, depends on the value of the type field.

```
<Calibration Values Record> ::=
      <Calibration Header Line>
      <Calibration Description Line>
    [ <Function Coding Line>
      <Variant Coding Line>
      <Display Id Line> ; ];
```

2.1 Calibration Header Line

```
<Calibration Header Line> ::=
      [<Anything>] <Value Separator>
      <Calibration Identifier> <End of Line Separator>;

<Anything> ::= <Text> | <Floating Point Number>;
```

2.2 Calibration Description Line

```
<Calibration Description Line> ::=
      (
            'VALUE' <Value Separator> <Single Calibration Value Record> |
            'CURVE' <Value Separator> <Characteristic Line Values Record> |
            'MAP' <Value Separator> <Characteristic Map Values Record> |
            'CUBOID' <Value Separator>
                  <Characteristic Cuboid Values Record> |
            'VAL_BLK' <Value Separator> <Calibration Values Block Record> |
            'AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'X_AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'Y_AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'Z_AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'RESCALE_AXIS_PTS' <Value Separator>
                                    <Rescale Axis Points Values Record> |
            'ASCII' <Value Separator> <Ascii String Record>
      );

<Anything> ::= <Text> | <Floating Point Number>;
```

The calibration description line typically contains axis labels, physical units and axis name information. Application Systems importing a calibration values record should skip this line entirely.

Value Records see Chapter 2.6

2.3 Function Coding Line

```
<Function Coding Line> ::=
                        'FUNCTION' <Function Value Record>;

<Function Value Record> ::=
                        [<Anything>] <Value Separator>
                        <Function Name> <Value Separator>
                        <Function Description> <End of Line Separator>;

<Function Name> ::= <Text>
<Function Description> ::= <Text>
<Anything> ::= <Text> | <Floating Point Number>;
```

The function coding line is optional. If there exists a function coding line within a calibration values record then there must exist a function header in the file header at the beginning of the file.

2.4 Variant Coding Line

```
<Variant Coding Line> ::=
                  'VARIANT' <Value Separator>
                  [<Anything>] <Value Separator>
                  <Variant Code> <End of Line Separator>;

<Variant Code> ::= n ( <Ascii String Record 1>
                        <Decimal Point Separator>
                        <Ascii String Record 2> );

      with n ::= number of variant criterions  and

      <Ascii String Record 1> ::= <Text> // name of variant criterion
      <Ascii String Record 2> ::= <Text> // value of variant criterion
```

The variant coding line is optional. This line is used to support the exchange of variant coded calibration parameters. If a calibration parameter is variant coded by the ASAP2 file then more than one calibration parameter with the same name has to be written into one CXV file. The only difference of these calibration parameters is to be found in the additional variant information in the variant coding line. In this way data of more than one variant can be exchanged by one CVX file.
If there exists a variant coding line within a calibration values record then there must exist a variant header in the file header at the beginning of the file.
If a variant coding exists in a CVX file and a calibration value has no variant coding line then this calibration value is valid for all allowed variants.

Example - using ';' as a field separator and '.' as a decimal point separator and '"' as a string delimiter:

The following variant criterion with the associated values exist:

3. Car (Limousine, Cabrio, Kombi)
4. Gear (Manual, Automatic)

Calibration parameter with variant coding in one CVX file (2 variants displayed):

```
---begin example---
;Constants
VAL_BLK;;1;1.5;2.25;3.75
'VARIANT';;"Car"."Limousine";"Gear"."Manual"

;Constants
VAL_BLK;;1;2;3;4.89
'VARIANT';;"Car"."Cabrio";"Gear"."Manual"
---end example---
```

When importing values of calibration parameters with a variant coding then it should be checked if the variants associated to the imported parameters are valid variants or if they are forbidden variants (ASAP2 keyword VAR_FORBIDDEN_COMB).

2.5 Display Identifier

```
<Display Id Line> ::=
                        'DISPLAY_IDENTIFIER' <Display Id Record>;

<Display Id Record> ::=
                        [<Anything>] <Value Separator>
                        <Display Identifier> <End of Line Separator>;

<Display Identifier> ::= <Text>
<Anything> ::= <Text> | <Floating Point Number>;
```

The display identifer line is optional. The display identifier shall comply with the rules of the actual ASAP2 V1.x.y standard (today ASAP2 V1.3.1).

2.6 Value Records

2.6.1 Single Calibration Value Record (ASAP2 Value)

```
<Single Calibration Value Record> ::=
      [<Anything>] <Value Separator>
      <Physical Value>;

<Physical Value> ::= <Floating Point Number> | <Text>;
```

Example - using ';' as a field separator and ',' as a decimal point character:

```
---record begin---
;KaEGRC_Air_Temperature_Threshhold
VALUE;;1,57

---record end---
```

Example Excel Representation:

	KaEGRC_Air_Temperature_Threshhold		
VALUE		1,57	

2.6.2 Characteristic Line Values Record (ASAP2 Curve)

```
<Characteristic Line Values Record> ::=
      { [<Anything>] <Value Separator> } <New Line>
      <Indentation> <Physical Value>
      { <Value Separator> <Physical Value> };
<Indentation> ::=
      [<Anything>] <Value Separator>
      [<Anything>] <Value Separator>;
<Physical Value> ::= <Floating Point Number> | <Text>;
```

Ordering of values:
Let x denote the x axis and n its dimension;
Let z denote the array of characteristic line values;
Let line 1 denote the calibration header line;
Then the following table shows the ordering of values in a characteristic line values record.

```
line 1:     col2        col3        col4        ...   col n+2
line 2:                 x[1]        x[2]        ...   x[n]
line 3:                 z[1]        z[2]        ...   z[n]
```

The ASAP2 Standard calls this ordering of values "index increment"

Example - using ';' as a field separator and ',' as a decimal point character:

```
--- record begin ---
;KvEGRC_Overtemp_Time
CURVE
;;4,78;6,89;12

--- record end ---
```

Example Excel Representation:

	KvEGRC_Overtemp_Time				
CURVE					
		4,78	6,89	12	

2.6.3 Characteristic Map Values Record (ASAP2 Map)

```
<Characteristic Map Values Record> ::=
      { [<Anything>] <Value Separator> } <New Line>
      { [<Anything>] <Value Separator> } <New Line>
      { <Indentation> <Physical Values> <End of Line Separator> };
```

```
<Indentation> ::=
      [<Anything>] <Value Separator>
      [<Anything>] <Value Separator>;
<Physical Values> ::=
      <Physical Value> <Value Separator>
      { <Physical Value> <Value Separator> };
<Physical Value> ::= <Floating Point Number> | <Text>;
```

Ordering of values:
Let x denote the x (first) axis and n its dimension;
Let y denote the y (second) axis and m its dimension;
Let z denote the two-dimensional array of map values;
Let line 1 denote the calibration header line;
Then the following table shows the ordering of values in a characteristic map values record.

```
line 1:     col2        col3        col4        ...   col n+2
line 2:                 x[1]        x[2]        ...   x[n]
line 3:     y[1]        z[1,1]      z[1,2]      ...   z[1,n]
line 4:     y[2]        z[2,1]      z[2,2]      ...   z[2,n]
            ...
line m+2:   y[m]        z[m,1]      z[m,2]      ...   z[m,n]
```

The ASAP2 Standard defines this ordering of values as "by rows" (german: zeilenweise) and "index increment"

Example - using ';' as a field separator and ',' as a decimal point character:

```
---begin record---
;KaEGRC_Base_Position_Lo_Oct
MAP

;;4,5;3,9;4,89
;;5,345;2,89;6,89
---end record---
```

Example Excel Representation:

	KaEGRC_Base_Position_Lo_Oct				
MAP					
		4,5	3,9	4,89	
		5,345	2,89	6,89	

2.6.4 Characteristic Space Values Record (ASAP2 Cuboid)

```
<Characteristic Space Values Record> ::=
//Under Investigation
//Type field value 'CUBOID' reserved.
```

2.6.5 Calibration Values Block Record (ASAP2 Value Block)

```
<Calibration Values Block Record> ::=
      { [<Anything>] <Value Separator> } <New Line>
      { <Physical Value> <Value Separator> };

<Physical Value> ::= <Floating Point Number> | <Text>;
```

Example - using ';' as a field separator and ',' as a decimal point character:

```
---begin record---
;Constants
VAL_BLK;;7,65;0,24;9;0,456

---end record---
```

Example Excel Representation:

	Constants					
VAL_BLK		7,65	0,24	9	0,456	

2.6.6 Rescale Axis Points Values Record (ASAP2 Axis Points with Rescale values)

```
<Rescale Axis Points Values Record> ::=
      [ <Anything> ] <Value Separator>
      <Physical Value> <Value Separator> <Physical Value>
      { <Value Separator> <Physical Value>
      <Value Separator> <Physical Value> };

<Physical Value> ::= <Floating Point Number>;
```

This record format is to be used for AXIS_PTS, X_AXIS_PTS, Y_AXIS_PTS and Z_AXIS_PTS.

Example - using ';' as a field separator and ',' as a decimal point character:

```
---begin record---
;KpmGroupAxis_3_26
RESCALE_AXIS_PTS;;1;15.75;20;30.75;35;60.75

---end record---
```

The example is associated with the following pairs of rescale values in the hexfile:

```
      (1,10.5),(20,20.5),(35,40.5)
```

and the following conversion formula:

```
      phys = hex * 1,5
```

The axis point values are calculated by a fixed algorithm based on the pairs of the rescale values in the hexfile. The amount of the calculated axis points depends in every case on the associated calibration parameter . When calculating the physical values every 1st hexadecimal value of a pair will be 1:1 a physical value and the 2nd hexadecimal value of a pair will be calculated by using the conversion formula.

2.6.7 Axis Points Values Record (ASAP2 Axis Points)

```
<Axis Points Values Record> ::=
      <Value Separator> [ <Anything> ]
      <Physical Value> { <Value Separator> <Physical Value> };

<Physical Value> ::= <Floating Point Number> | <Text>;
```

Ordering of values: ascending

This record format is to be used for AXIS_PTS, X_AXIS_PTS, Y_AXIS_PTS and Z_AXIS_PTS.

Example - using ';' as a field separator and ',' as a decimal point character:

```
---begin record---
;KpmGroupAxis_3_26
AXIS_PTS;;600;800;1000

---end record---
```

Example Excel Representation:

	KpmGroupAxis_3_26				
AXIS_PTS		600	800	1000	

2.6.8 Ascii String Value Record (ASAP2 Ascii)

```
<Ascii String Record > ::= <Text>
```

Example - using ';' as a field separator and '"' as a string delimiter:

```
---begin record---
;MyName
ASCII;;"CVX V1.0"

---end record---
```

Example Excel Representation:

	MyName				
ASCII		CVX V1.0			

3 Comment Records

```
<Comment Record> ::=
      <Comment Indicator> <Value Separator>
      {[<Anything>] <Value Separator>} <New Line>;
```

4 Verbal conversion formulae (ASAP2 COMPU_VTAB)

Since the ASAP2 V1.21 it is possible to define every type of calibration parameter including a verbal conversion formulae. To support this feature within the CVX format it is necessary to define as physical value not only floating point numbers but also ascii strings (<Text>).

```
<Physical Value> ::= <Floating Point Number> | <Text>;
```

Example - using ';' as a field separator and '"' as a string delimiter:

```
---begin record---
;Constants
VAL_BLK;;"Textstring1";"Textstring2";"Textstring3";"Textstring4"

---end record---
```

Example Excel Representation:

	Constants					
VAL_BLK		Textstring1	Textstring2	Textstring3	Textstring4	

5 Limits

Max. Field Width: 1023 Characters

A 1 Kbytes input buffer is sufficiently large to read a transfer unit field by field including all optional fields.

When only significant fields need to be extracted from a transfer unit a 32 Byte buffer is sufficient.

Max. Identifier Length: 256 Characters

Application conforming to the format defined in this document should not export identifiers longer than 256 characters, but may accept longer identifiers.

(Limit taken from the ASAP2 Standard V1.30)

Smallest Max. Floating Point Number: 1.7976931348623157e+308
Largest Min. Floating Point Number: 2.2250738585072014e-308

(Limits taken from the ANSI C Standard for data type double)

Applications may of course implement larger floating point ranges.

Max. size of transfer units

Not defined here. Exporting applications should however be aware, that there are limitations with standard applications, e.g. Excel, on the maximum number of lines, columns and total size. Thus, exporting applications should provide means to partition transfer units into smaller pieces.

6 Case Sensitivity

When importing calibration values, comparison of calibration identifiers should be non case sensitive. In the event of an ambiguity, the importing system should attempt to resolve the ambiguity by applying a case sensitive comparison. If there exists a case sensitive ambiguity the system should check if there exists a variant coding. If a variant coding exists all calibration values must have a variant coding line (see 2.3 Variant Coding Line).

If the ambiguity cannot be resolved, no import of the calibration record in question should be performed.

On Export the identifier should be written exactly as entered by the user or received from another external system.

7 Dimension Mismatch

Interpretation of value arrays which are too big or too small compared to the definition associated with a given calibration identifier are up to the importing system.

Possible actions:
- abort import of record in question
- fill with values
- ignore excess values

8 Floating Point Precision

Importing systems must be capable of importing floating point numbers with 15 significant digits without loss of precision.

Systems should export floating point numbers with the maximum available precision.

9 Import of embedded axis points

Axis point values contained in characteristic line values and map values records shown in some examples in this document have been put there to improve readability and to demonstrate usage of optional fields of the CVX format. They are not part of the format as axis points, but only as optional text fields.

Application systems importing calibration values should discard axis point values embedded in characteristic line values records and map values records without warning.

Axis point values should only be exported/imported explicitly using axis points values records.

For anonymous axis points objects, i.e. axes which are not associated with group characteristic objects, which do not have a name of their own in the exporting system and thus cannot use the above mentioned method, two new record types 'Y_AXIS_PTS' and 'X_AXIS_PTS' are used. 'Z_AXIS_PTS' is reserved for future use with cuboids. The identifier associated with these records short be the name of the map.

Example Excel Representation:

CC47116	**KaEGRC_Base_Position_Lo_Oct**				
MAP	->				
	->				
	mm:°C:rpm	600	800	1000	
	3,2	**4,5**	**3,9**	**4,89**	
	5,8	**5,345**	**2,89**	**6,89**	
CC47116	**KaEGRC_Base_Position_Lo_Oct**				
X_AXIS_PTS	°C	**600**	**800**	**1000**	
CC47116	**KaEGRC_Base_Position_Lo_Oct**				
Y_AXIS_PTS	rpm	**4,5**	**5,8**		

10 References

ASAP2: Standardisierte Beschreibungsdaten, Version 1.0, 31.03.1994
Applications Systems Standardization (ASAP); Working Group Interface Specifications; Interface 2;
Version 1.21 of 09/03/99 and V1.30 of 02/11/99.

11 Syntax Summary

```
<Calibration Values Transfer Unit> ::=
      <File Header>
      { <Record Separator> <Calibration Values Record> };
<File Header> ::=
      <Description Header>
      [<Function Header>
      <Variant Header>];
<Description Header> ::=
      <File Content Identification> <Value Separator>
      [ <Decimal Point Separator> <Value Separator>
        <Comment Indicator> <Value Separator>
        <String Delimiter> <String Delimiter> <Value Separator> ]
      <End of Line Separator>;
<File Content Identification> ::= 'CALIBRATION VALUES V' <Version Number>;
<Version Number> ::= <Major Version Number> '.' <Minor Version Number>;
<Major Version Number> ::= <Natural Number>;
<Minor Version Number> ::= <Natural Number>;
<Version Number> ::= <Major Version Number> '.' <Minor Version Number>;
<Major Version Number> ::= <Natural Number>;
<Minor Version Number> ::= <Natural Number>;
<Natural Number> ::= <Digit> {<Digit>};
<Comment Indicator> ::= {<Ascii character>}
<Decimal Point Separator> ::= ',' | '.'
<End of Line Separator> ::= {<Value Separator>} <New Line>;
<Function Header> ::=
      'FUNCTION_HDR' <Value Separator>
      [<Anything>] <Value Separator> <New Line>
      <Function Block Record>;
<Function Block Record> ::=
      {<Value Separator> <Function Name>} <End of Line Separator>;
<Function Name> ::= <Text>;         //any valid text string
<Text> ::= <String Delimiter> {<Ascii Character>} <String Delimiter>
<Variant Header> ::=
      'VARIANT_HDR' <Value Separator>
      [<Anything>] <Value Separator> <New Line>
      <Variant Block Record>;
<Variant Block Record> ::=
      {<Variant Criterion> <Value Separator>
      {<Criterion Value> <Value Separator>} <New Line> }
      <End of Line Separator>;
<Variant Criterion> ::= <Text>;
<Criterion Value> ::= <Text>;
<Record Separator> ::=
            <End of Line Separator> <Line Separator>
            { <End of Line Separator> <Line Separator> }
      |
            <End of Transfer Unit>;
```

```
<Floating Point Number> ::=
      [<Sign>] {<Digit>}
      [ <Decimal Point Character> <Digit> {<Digit>} ]
      [ ('e'|'E') [<Sign>] {<Digit>} ];
<Sign> ::= '+' | '-';
<Digit>     ::= '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
<Calibration Identifier> ::= <Text>
<Valid Calibration Identifier> ::=<String Delimiter> <Letter> {
      (<Letter>|<Digit>|'.'|<IndexSpec>) } <String Delimiter>;
<Letter>    ::= 'a'-'z' | 'A'-'Z';
<IndexSpec> ::= '[' <Digit> {<Digit>} ']';
<Calibration Values Record> ::=
      <Calibration Header Line>
      <Calibration Description Line>
      [ <Variant Coding Line>; ];
<Calibration Header Line> ::=
      [<Anything>] <Value Separator>
      <Calibration Identifier> <End of Line Separator>;
<Calibration Description Line> ::=
      (
            'VALUE' <Value Separator> <Single Calibration Value Record> |
            'CURVE' <Value Separator> <Characteristic Line Values Record> |
            'MAP' <Value Separator> <Characteristic Map Values Record> |
            'CUBOID' <Value Separator>
                  <Characteristic Cuboid Values Record> |
            'VAL_BLK' <Value Separator> <Calibration Values Block Record> |
            'AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'X_AXIS_PTS' <Value Separator> <Axis Points Values Reco rd> |
            'Y_AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'Z_AXIS_PTS' <Value Separator> <Axis Points Values Record> |
            'RESCALE_AXIS_PTS' <Value Separator>
                                    <Rescale Axis Points Values Record> |
            'ASCII' <Value Separator> <Ascii String Reco rd>
      );
<Function Coding Line> ::=
      'FUNCTION' <Function Value Record>;
<Function Value Record> ::=
      [<Anything>] <Value Separator>
      <Function Name> <Value Separator>
      <Function Description> <End of Line Separator>;
<Function Name> ::= <Text>
<Function Description> ::= <Text>
<Anything> ::= <Text> | <Floating Point Number>;

<Variant Coding Line> ::=
      'VARIANT' <Value Separator>
      [<Anything>] <Value Separator>
      <Variant Code> <End of Line Separator>;
<Variant Code> ::=
      n ( <Ascii String Record 1>
      <Decimal Point Separator>
      <Ascii String Record 2> );
      with n ::= number of variant criterions and
<Ascii String Record 1> ::= <Text> // name of variant criterion
<Ascii String Record 2> ::= <Text> // value of variant criterion
<Anything> ::= <Text> | <Floating Point Number>;
<Single Calibration Value Record> ::=
      [<Anything>] <Value Separator>
```

```
      <Physical Value>;
<Physical Value> ::= <Floating Point Number> | <Text>;

<Display Id Line> ::=
      'DISPLAY_IDENTIFIER' <Display Id Record>;
<Display Id Record> ::=
      [<Anything>] <Value Separator>
      <Display Identifier> <End of Line Separator>;
<Display Identifier> ::= <Text>
<Anything> ::= <Text> | <Floating Point Number>;

<Characteristic Line Values Record> ::=
      { [<Anything>] <Value Separator> } <New Line>
      <Indentation> <Physical Value>
      { <Value Separator> <Physical Value> };
<Indentation> ::=
      [<Anything>] <Value Separator>
      [<Anything>] <Value Separator>;
<Characteristic Map Values Record> ::=
      { [<Anything>] <Value Separator> } <New Line>
      { [<Anything>] <Value Separator> } <New Line>
      { <Indentation> <Physical Values> <End of Line Separator> };
<Physical Values> ::=
      <Physical Value> <Value Separator>
      { <Physical Value> <Value Separator> };
<Calibration Values Block Record> ::=
      { [<Anything>] <Value Separator> } <New Line>
      { <Physical Value> <Value Separator> };
<Rescale Axis Points Values Record> ::=   [ <Anything> ]
      <Value Separator> <Physical Value> <Value Separator> <Physical Value>
      { <Value Separator> <Physical Value>
      <Value Separator> <Physical Value> };
<Axis Points Values Record> ::=
      <Value Separator> [ <Anything> ]
      <Physical Value> { <Value Separator> <Physical Value> };
<Ascii String Record > ::= <Text>;
<Comment Record> ::=
      <Comment Indicator> <Value Separator>
      {[<Anything>] <Value Separator>} <New Line>;
```