Design Automation

(Macro Technique)

THE WORLD OF CAD AND PDM SOLUTIONS

UNLIMITED PERFORMANCE

I·S·D

Contents

About This Manual 5

1 Working with Macros 7

1.1 Basics 7

1.2 Text Mode 9

1.3 Create Macro 10

1.3.1 Macro Structure 11

1.3.2 Macro Commands 11

1.3.3 Free Argument # 12

1.3.4 End Macro Creation 13

1.4 Macro Call 13

1.5 Process Macro 14

1.5.1 Overwrite Macro 14

1.5.2 Take Over Macro 15

1.5.3 Amend Macro 16

1.5.4 Macro Compilation 20

2 Macro Language 21

2.1 HCGS Command Groups 21

2.2 Argument Groups 23

2.3 Variables 26

2.3.1 User Variables 26

2.3.2 Point and Line Variables 26

2.3.3 System Variables 27

2.4 Arithmetic Expressions 33

2.4.1 Arithmetic Operators 33

2.4.2 Basic Functions 34

2.5 Logical Comparisons in Expressions 35

2.6 Logical Variables 37

2.7 String Expressions 39

2.7.1 Convert Numeric Variable to String 39

2.7.2 String Operations 40
2.7.3 String Functions 40
2.8 Variable Memory 42

3 Macro Language Commands 43

3.1 Commands 45
3.1.1 Antwort 47
3.1.2 APAUS / APEIN 49
3.1.3 CALL 50
3.1.4 COPY 52
3.1.5 DEL 54
3.1.6 DISTANZ 55
3.1.7 ECHO 57
3.1.8 END 59
3.1.9 FAA / FAE 60
3.1.10 FOR ... NEXT 61
3.1.11 GOTO 63
3.1.12 IF..ELSE..IFEND 65
3.1.13 IGNORE 67
3.1.14 INTEGER 68
3.1.15 MAKRO 70
3.1.16 MAUS/MEIN 72
3.1.17 MKDIR 73
3.1.18 OPEN INPUT .....CLOSE
OPEN OUTPUT ..... CLOSE 74
3.1.19 OPTION 77
3.1.20 PFD 80
3.1.21 POINT 81
3.1.22 REAL 83
3.1.23 REM 84
3.1.24 REPEAT 86
3.1.25 SAUS / SEIN 87
3.1.26 START 89
3.1.27 STRING 90
3.1.28 SZAUS / SZEIN 92

3.1.29 UDA / UDE 93
3.1.30 VAI 94
3.1.31 VAR 95
3.1.32 Variable Assignment 97
3.1.33 WAIT 99
3.1.34 WARTE 101
3.1.35 WAUS/WEIN 102
3.1.36 WERT 103
3.1.37 WHILE...WHEND 104
3.1.38 WINKEL 106
3.1.39 ZAA / ZAE 108
3.2 Example 110

4 Additional Notes 115
4.1 DXF Files 115
4.2 Macro variable: ZDSP 115
4.3 Select Text 115
4.4 Text Tools 115
4.5 True Type Font 116
4.6 Objektcursor 116
4.7 Develop Sheet Metal 116
4.8 Dimensioning 116
4.9 Sketching Cursor 116
4.10 Component ID 117
4.11 Call Operating System 117
4.12 Material Hatching 117
4.13 Export Data to STEP/MTA or IGES/CATIA 117
4.14 Temporary path 118
4.15 Macro Variables for UNDO 119

5 Error Messages 121

6 Macro Libraries 123

7 The FILEGRUP.DAT File 124

Index 133

About This Manual

Reserved words in HiCAD's graphical macro language are based on both English and German. At first sight, keywords in German-like language may be irritating, but in practice the syntactical rules are easy to grasp.

To assist orientation, we have attached a glossary listing German based acronyms and their meaning to this document. It can be unfolded and consulted until you are familiar with the syntax and semantics of the language.

Chapter 2 gives you an in-depth coverage of HCGS, HiCAD's macro programming language, while Chapter 3 contains a detailed description of HCGS commands.

Our goal is to help you find specific information quickly without having to search through the entire document, as well as to provide a comprehensive description of program features.

1 Working with Macros

1.1 Basics

HiCAD's advanced Macro and Variant capabilities enable you to speed up design tasks and automate HiCAD procedures.

The HiCAD Macro Development System HC-MES (HiCAD **M**akro**e**ntwicklungs-**s**ystem) enables you to record often-repeated HiCAD operations and save them in a macro. Based on HiCAD's macro language HCGS (**HiC**AD **G**rafische **S**prache = HiCAD Graphical Language), HC-MES uses its "program by example" capability to log user operations as they occur. This means that HiCAD routines, no matter how complex, can be directly recorded. Macros created in this way can be processed and tested at a later date with HiCAD's Macro Editor.

As HiCAD macros can be integrated in the screen menu, you can expand HiCAD's User Interface to suit your own environment.

In HiCAD, macros are created semi-automatically with HC-MES, our own macro development system.

To create a macro, you need to first specify it's name. Next enter the series of commands that should be executed. You do this by using the appropriate functions when creating a component in HiCAD. All selected functions and their associated specifications are then logged to the given macro. If a function expects a user input, you need to specify whether the input should be interpreted for a predefined value entry or a query during a macro run.

Individual functions are recorded in the given sequence until you exit macro creation.

HiCAD's Macro Interpreter not only allows you to interpret and execute commands created with **HC-MES** but enables you to enter an optional number of commands explicitly in the macro. These entries can range from a simple command that displays a specific user message on the screen to the type of complex loop instructions, logical queries etc., used by high-order languages.

Macros can be processed retroactively, either with the integrated Macro Editor or any text editor.

To record a sequence of commands and save them in a macro, you need to work in Text Menu Mode, i.e. not with icons. This mode is automatically activated when you create or edit macros. Although functions are dis-

played in special text menus, the program features are the same as those normally represented by icons.

When you call the **Create Macro** function, HiCAD activates Text Mode and displays the root menu, thus providing access all other functions.

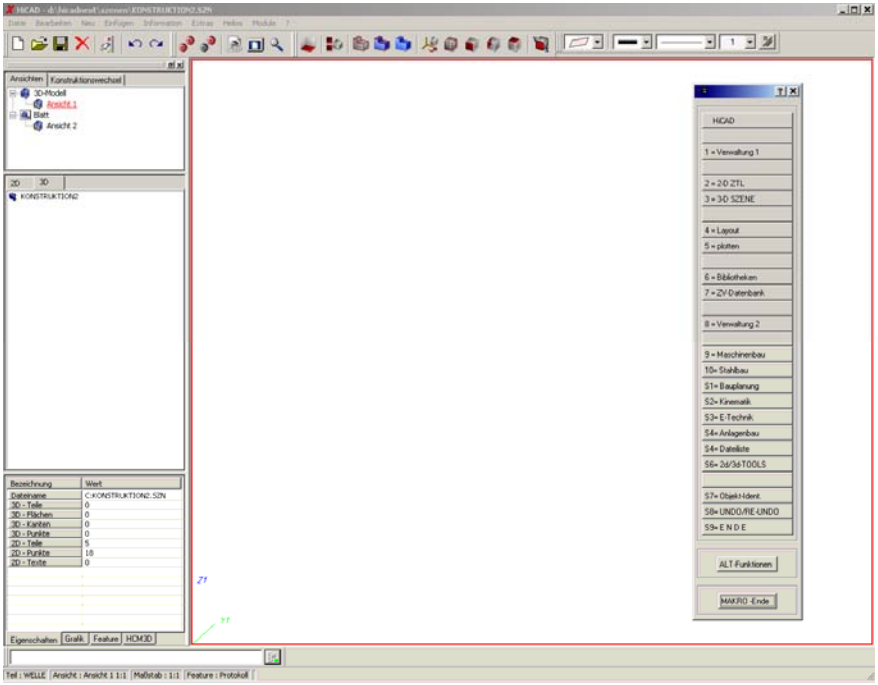


Fig. 1 User interface in text mode

1.2 Text Mode

When you create macros, it is important to remember that:

- When you create a new 2D or 3D drawing, you always need to invoke the functions: **3D Scene** and **Create New**.
- To create a new 2D part, you need to invoke the function sequence: **2D-ZTL**, **Process ZTL** and **New Main Part** or **New Part** (for subordinate components).
- To create a new 3D part, you need to invoke the function sequence: **3D Scene**, **Process** and **New Main Part** or **New Part** (for subordinate components).
- Special functions, i.e. zoom functions, view generation functions, grid functions as well as plotting and printing can be accessed by selecting the **ALT Functions** button.
- **Macro End** exits macro creation.
- ESC or the right mouse button returns you to the previous menu.

You will find a folded reference card showing all significant menus at the end of this manual.

1.3 Create Macro

To create a macro, invoke the **Create Macro** function with the keys

Ctrl + 7

Next, specify a name for the macro. HiCAD now automatically activates the macro creation system, switches to text menu mode and displays the root menu.

If a macro already exists under the given name, it can now be edited.

Please refer to. 1.5

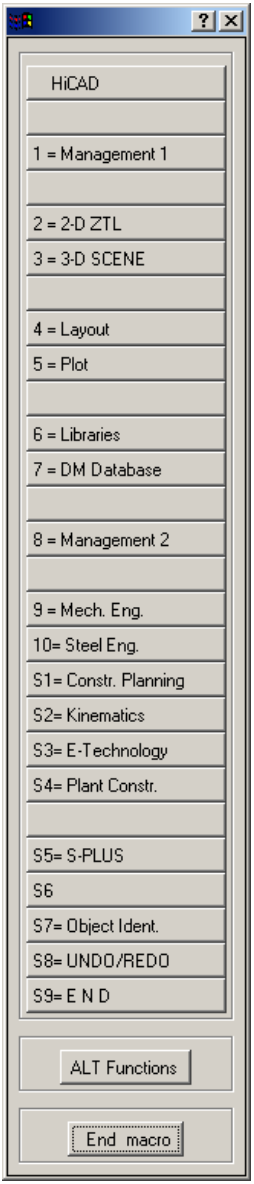


Fig. 2 Main menu

1.3.1 Macro Structure

- **Macro Name**
  The file name extension ".MAC" is automatically set by HiCAD.

- **Macro Frames**
  Each macro is identified by start and end instructions.

```
START menu level
.
.      Instructions
.
END
```

  Whereby, *menu level* is the number of the menu level, in which macro creation started. This number is set automatically by HiCAD when a macro is created.

Cf. 3.1.8 and 3

1.3.2 Macro Commands

When you select the **Create Macro** function and specify a name HiCAD records all invoked functions and associated specifications. As this also applies to incorrect entries. Please plan the command sequence carefully before starting macro creation.

Apart from the functions that are logged automatically as commands by HiCAD during macro creation, any number of instructions can be inserted, e.g.

- Loop instructions,
- Logical queries with jump instructions,
- Variable declarations, etc.

You can either use the built-in macro editor, or a text editor to type explicit commands. This can be done at a later date.

Cf. 1.5.3 and 3

1.3.3 Free Argument

You can use so called "free" arguments indicated by the character # in place of a text or name value, thereby providing a high degree of flexibility.

When the macro reaches a free argument, the run is paused and the entry of a value, text or name is prompted. Although to enable macro creation procedures to be correctly carried out you sometimes have to specify a value, text or name after specifying a free argument during macro creation, only the free argument is logged to the macro.

Point specifications are special cases. If you use RET during macro creation, the character # is automatically recorded in the macro.

The following example demonstrates the way in which free arguments can be used.

Example:

The creation of a 2D component is recorded and saved to a macro. The user should be able to specify an optional name. The following function sequence should be selected.

- Call the **Create Macro** function
- Select **2-D ZTL** and **Process ZTL**
- Select **New Main Part** and **Create Part**
- Enter the component name: **# COMP1**
- Select **MACRO END**

This creates the following macro:

```
REM     HiCAD
START   59
REM     HiCAD   2 = 2-D ZTL
OPTION  2  59
REM     DRAWINGS   3 = PROCESS ZTL
OPTION  3   1
REM     COMPONENTS 8 = New main part
OPTION  8   2
REM     NEW PART 1 = Create part
OPTION  1   3
STRING  #
END
```

When this macro is called, you are expected to specify a name under which the part should be saved. HiCAD then creates a new part with the given name.

1.3.4 End Macro Creation

Select **MACRO End** from the main text menu to close macro creation.

HiCAD then ends macro recording mode.

1.4 Macro Call

To call a macro select **Call Macro** by pressing the key combination **Strg+8**

To start the macro, either enter or select a macro name.

You can cancel a macro run by pressing the **ESC** key.

1.5 Process Macro

To edit a macro you need to invoke the **Create Macro** function and select the required macro. HiCAD then displays a pop-up menu with processing options. Select the required option.

If you select **Cancel**, no edits are made and the macro file is immediately deleted.

When you edit an existing macro, HiCAD automatically creates a backup file under the original name with the file extension *.BAK. Earlier copies are thereby overwritten. These data saves are not suitable for macros you mean to compile but are intended to save edits to the macro.

Fig. 3 Process macro

1.5.1 Overwrite Macro

This function overwrites the selected macro, i.e. the file content is deleted and subsequent functions recorded and saved.

HiCAD does not backup the source file!

1.5.2 Take Over Macro

This function enables you to copy part or all of a macro, and then expand it with new commands. You need to specify the number of lines you want to copy.

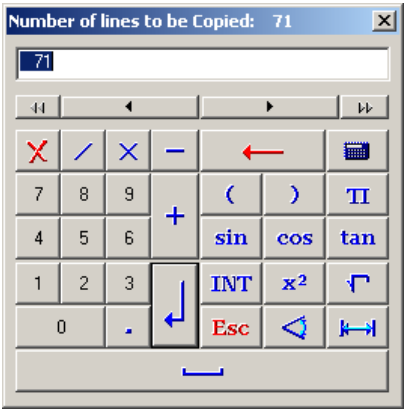


Fig. 4 Take Over Macro

When you call this function, the total number of macro command lines is offered, thus allowing you to copy the entire macro. The function is, however, normally used to enhance an existing macro. If a number **n** greater than 0 is entered, the first **n** lines of the macro are taken over. If **n** is less than 0, the last n lines are deleted from the macro, i.e. all macro lines except the last **n** lines are taken over.

A macro run is executed during the take over. Loops, conditional instructions and CALL commands are thereby respected. When all of these commands have been executed, HiCAD switches to macro creation mode. The options described in Section 1.3 are available.

HiCAD saves a backup of the source file with the file name extension .BAK.

1.5.3 Amend Macro

When you select this function, macro pages are displayed by the Macro Editor.

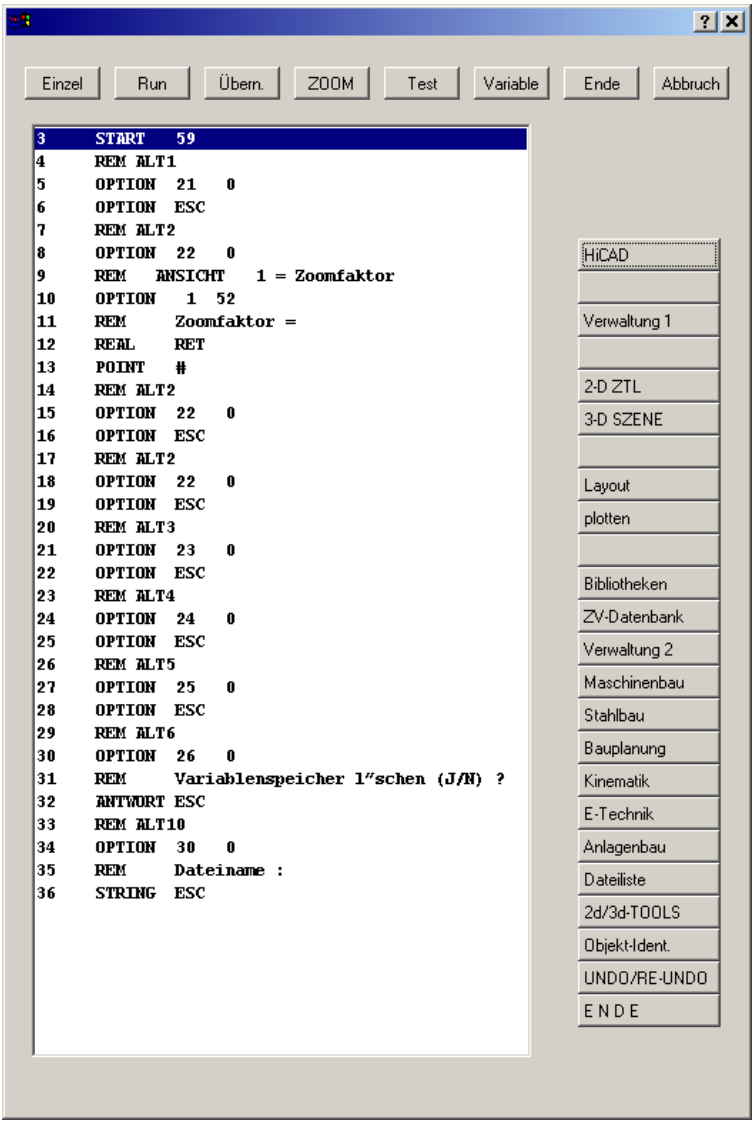


Fig. 5 Macro Editor

The current macro line is coloured in blue. The **START** command is always in the first line. Simply click the line you want to edit and enter the correction.

You can use your cursor to select macro processing options displayed in the top group of the Macro Editor.

1.5.3.1 Single

This function switches to single-step mode, i.e. **RUN** only executes the current macro line. Subsequently, the next executable line is offered.

1.5.3.2 Zoom

This option enables you to pause macro execution to, e.g. select another area of your design or define a grid. When you select the function, HiCAD displays the appropriate pop-up menu.

Fig. 6 Zoom options

After you have selected one of the zoom options, you are returned to macro processing.

1.5.3.3 Run

The macro is executed until it reaches the current (blue) line. In single-step mode, only the current line is executed.

1.5.3.4 Test

This function enables you to check macro commands. Incorrect commands are marked and can then be corrected.

1.5.3.5 Take Over

This function takes over the entire macro and returns you to macro creation mode.

1.5.3.6 Label

This function enables you to set labels for jump instructions within the macro. Invoke the function and select the line in which the label should be set. A pop-up menu enabling you to select a label is displayed.

1.5.3.7 Save

Select this function to save the macro.

1.5.3.8 Delete

Use this function to delete a line from the macro. Select the appropriate line with the cursor.

1.5.3.9 End

This function is similar to the **Take Over** function, but the macro is only taken over up to the current (blue) line. Subsequently HiCAD returns you to macro creation mode.

1.5.3.10 Cancel

This function interrupts macro processing. Modifications are not saved.

1.5.3.11 Variables

Select this function if you want to check or amend variables during macro processing. Currently defined numeric and text variables are then displayed. Choose the variable you want to change and enter a new value or text. Click **Take Over** to pass the new values to the macro.

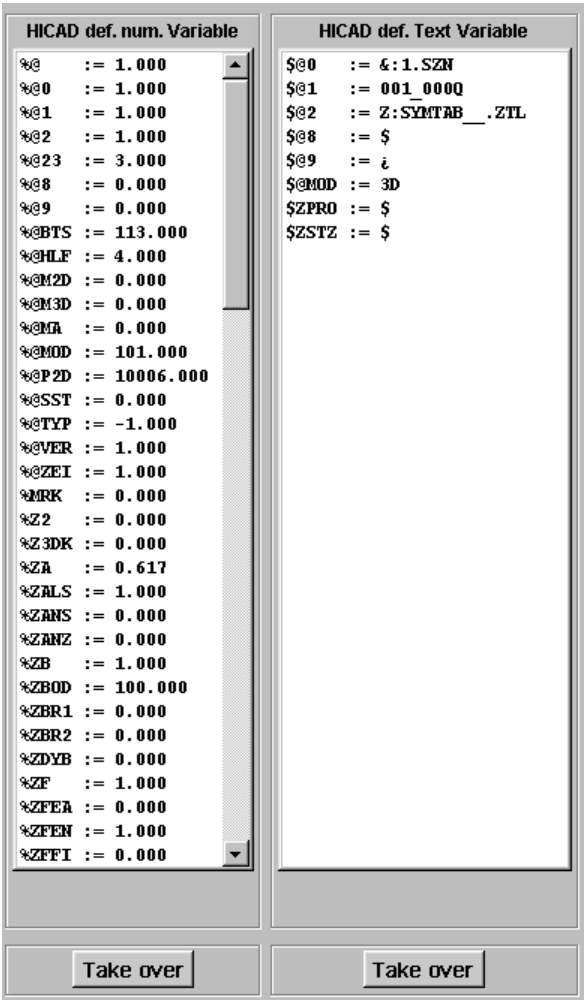


Fig. 7 Check or modify variables

1.5.4 Macro Compilation

When you call a macro, the ASCII file containing the macro commands must be first compiled, i.e. read and converted to a compact format for execution.

In this compact format, individual HCGS commands are coded by a single byte, "Remark" lines are not included, e.g. Complete. This compressed version only needs a fraction of the memory space required by the macro in "readable" ASCII format.

A compiled macro can also be stored in the database. This not only saves a considerable amount of memory space but shortens the time needed to retrieve it.

You will find that macro compilation is invaluable, especially for large or frequently used macros.

N.B. Compiled macros can no longer be amended or expanded.

As macro source text in ASCII format is no longer available, it is important to remember to backup the original version.

This function is an optional extra.

2 Macro Language

Each HCGS command contains two components, i.e.

Keyword ***Argument***

Both keywords and arguments can be divided into groups.

2.1 HCGS Command Groups

Initialisation	**START, END** These commands, designating the start and end of a macro, may only appear once.
HiCAD Functions	**OPTION** Activates HiCAD functions
Scalar Entries	**STRING, REAL, INTEGER, ANTWORT*** Commands required by the HiCAD formula interpreter.
Geometric Entries	**POINT, DISTANZ*, WINKEL*** Commands required for graphical specifications.
Sub-Macro Calls	**CALL, MAKRO** Calls a subordinate macro from the current macro.
Output Control	**APEIN, APAUS, ECHO, HFEIN, HFAUS, MEIN, MAUS, SEIN, SAUS, SZEIN, SZAUS, WEIN,** **WAUS, WAIT, WARTE*, UDA*, UDE*,ZAE*, ZAA** Command controlling output, i.e. to text boxes.

- *Antwort = response/answer, Winkel = angle, Makro = macro, Distanz = distance*

  Please refer to Section 3.1 for details.

Macro Control	**FOR...NEXT, WHILE...WHEND, REPEAT... UNTIL, GOTO, IF...THEN...ELSE...IFEND, IGNORE** Commands determining jump instructions and loops.
Value Assignments	**% , $, VAI, VAR, PFD, DEL** Commands used to assign value to variables.
File Procedures	**OPEN, INPUT, OUTPUT, CLOSE, COPY, MKDIR** Commands used to read, write, copy and create ASCII files.
Remarks	**REM** This command enables you to insert an explanatory text that is ignored when commands are executed.

2.2 Argument Groups

Arguments in HCGS commands can be divided into the following groups:

Numbers Numbers are understood as integer positive values.

File Names File names are character strings representing valid file names. HiCAD file groups (cf. 6), e.g. C:, are permitted. File name extensions are set according to the required file type (e.g. .MAC).

Free Argument Free arguments are represented by #. When this character appears in a macro, the run is paused and a user input requested, i.e. usually a prompt related to the command in which the free argument was used. Free arguments are valid for all entry commands (scalar and geometric).

Constants Constants are strings containing ASCII characters. These strings must not contain variables. In this case blanks are recognised as belonging to the constant, not as delimiters. HiCAD differentiates between:

- **Integer numerical constants**
  Integral values, i.e. whole numbers without decimal positions.
- **Real constants**
  Numeric values with at least one leading position and a decimal point '**.**' (commas are invalid!!).
- **String constants**
  Alphanumeric character strings.

Variables

User variables are divided into string and real variables.

String or alphanumeric variables are always indicated by a leading $ and contain optional character strings with a maximum of 60 positions. The name of a variable (without the $ character) may not exceed 4 characters and must begin with an alpha character.

In the case of **Numeric Variables,** HiCAD does not differentiate between integer and real variables. If an integer entry is required but you enter a real variable as an argument, it is automatically rounded to the next whole number. The % character indicates the numeric conversion in a string. In an argument, a % character may not be set in front of a variable. The name of a variable may not exceed 31 characters and must begin with an alpha character.

It is irrelevant whether you use upper or lower case in variable names.

Cf. Section 2.3

Arithmetic Expressions

Arithmetic expressions can normally be used whenever a numerical value is expected. These expressions can also contain variables and return numerical values.

Cf. Section 2.4

Logical Expressions

A logical expression compares two values or arithmetic expressions and may contain variables, and returns a value of "true" or "false".

Cf. Section 2.5

Fixed Arguments

RET	RETURN is used to take over the offered value.
YES	is only valid with the ANTWORT* command. Responses 'Y' and '1' are equivalent.
NO	reciprocal to YES. Responses NO and '0' are equivalent.

**ANTWORT – Reserved word meaning response or answer*

Control Arguments

ESC	corresponds to END in HiCAD
ZEI	only valid with the DISTANZ command and corresponds to the entry 'z' in HiCAD specifications
LLL	Delete last line
LLA	Undo last delete
	(LLL and LLA are only valid when used in POINT commands)

Point Option

special argument for the POINT command.

Valid point options are: A, K, R, P, W, D, N, SP, L. *Invalid* options I, S, S2,M, M2, Z, F, O etc., i.e. point option referring to lines.

Cf. Section 2.3.2

2.3 Variables

2.3.1 User Variables

HiCAD offers both **numeric** and **alphanumeric variables.** Variable names may not contain more than 4 digits.

The name of numeric variables must start with an alpha character, all characters are valid **except Z (apart from ZA...) as well as L0 to L9 and P0 to P9.**

The names of alphanumeric variables must start with $ followed by an alpha character (with the exception of Z), e.g. $A1 or $AB. HiCAD does not differentiate between high and low case.

If a value is not assigned to a user-specific variable in the macro, HiCAD first searches the Variable Memory. If a value has not yet been allocated to the variable, HiCAD halts the macro run and expects you to enter an appropriate value.

If a value has already been assigned to the given variable, it is offered as the default value. You can either take it over with RETURN or delete the variable with END.

2.3.2 Point and Line Variables

The **Point Variables P0,...,P9** as well as **Line Variables L0,...,L9** have a special significance for user-specific variables. Using these variables it is possible to access graphical information and line elements. Point and line variables require graphic assignments, i.e. the specification of points.

2.3.3 System Variables

As well as User Variables HiCAD offers a range of System Variables containing significant data for macro creation.

In HCMT, system and user variables are of equal importance. You can access these variables by name and use them in, e.g. arithmetical expressions. Although value assignments are accepted by system variables, care should be taken as they are continually changed during processing.

System variables are usually set in the special functions that can be called via the **Information** function.

System variables are listed at the end of the section. System variables with names beginning with ZA... are (with the exception of ZA) user defined. These variables are not erased when the variable memory is deleted.

If you assign the current date or time to variables, e.g.

$u:=TIM$ or $u:=DAT$,

the following system variables are subsequently defined:

ZSTU	current hour	**ZJAR**	current year
ZMIN	current minute	**ZMON**	current month
ZSEC	current second	**ZTAG**	current day

With the variable, **ZJAR,** the year is expressed by four digits.

Sample:

```
$XX:=TIM$          %T1:=ZSTU
%T2:=ZMIN          %T3:=ZSEC
```

The next section contains listings of all HiCAD 2D and 3D system variables.

2-D Variable

Z		Centrifugal moment of inertia
Z0		Surface of an object or closed contour
Z1		Periphery or length of continuos, contour or polylines
Z2		Angle (set by all angle entries) The following have special significance with variable dimensioning: Z0 Type: 1=line, 2=angle, 3=circles, 4=arc Z1 Type: 1=indep., 2=chain, 3=parallel, 4=running, 5=partial section., 6=height above datum Z2 Mode: 0=normal, 1=paraxial
Z3		Length of graphic elements or distance (set by for all distance entries)
Z4		Moment of inertia (torque) re. y - axis When Dimension Info is called : Paraxial length of linear dimensions (1=x-parallel, 2=y-parallel, 0=not paraxial)
Z5		Moment of inertia (torque) ref. x – axis
Z6		Moment of inertia ref. y – axis
Z7		Moment of inertia ref. x – axis
Z8	X	Coordinates of the previously accessed point or the start point
Z9	Y	of an identified graphic element
ZA		Current zoom factor
ZB		Number of saved fitting points (set when loading objects);
ZC	X	Coordinates of the end point of an identified graphic element
ZD	Y	(set by all GE identifications)
ZCX	X	Coordinates of the identification point
ZCY	Y	used to identify lines
ZE		Moment of linear force ref. x – axis
ZH		Moment of linear force ref. y – axis
ZF		Object scale
ZG		Number of graphics screens
ZGA1– ZGA4		Coordinates of the previous "View All"
ZI		
ZJ	X	Coordinates of the centre of gravity
ZK	Y	of continuous, contour or polylines
ZL	X	Centre point coordinates of a circular arc
ZM	Y	When Dimension Info is called : x or y-coordinate of the dimension
ZN		Result of an evaluation request (calculator function)
ZP	X	Coordinates of screen centre point
ZQ	Y	
ZS		Point designation of identified point / Type of fitting point ZS=-1 ! ZS=-2 ? ZS=-3 ??
ZSC1, ZSC2		Hatch-code 1, Hatch-code 2

ZSD1, ZSD2		Offset hatch-code1/2
ZSW1, ZSW2		Angle hatch-code 1/2
ZT		Database ON/OFF [1/0]
ZU		2D/3D toggle (only assigned for macro start, 2=2D, 3=3D)
ZV		Value of previous numeric entry
ZW ZX	X Y	Coordinates of the bottom left corner of the entire drawing (full values) – set for "View All" or "Selected Drawing"
ZY ZZ	X Y	Coordinates of the top right corner of the entire drawing full values) – set for "View All" or "Selected Drawing"

Apart from numerical system variables, HiCAD also offers alphanumeric system variables that can be accessed by **$@** to **$@9**.

@		Conversion factor for dimension units to millimetre
@0		Conversion factor for natural coordinates to millimetre (@/@2)
@1		Conversion factor for millimetre to natural coordinates (@2/@)
@2		Scale value
@3 @4 @5	A B C	Coefficients A, B and C of the Hessian normal form of a line: A*x + B*y = C with A*A + B*B = 1. These variables are set if Function 4 is used within ALT 3 to identify a line;
@5		Radius of an identified circle
@6 @7	X Y	Coordinates of the centre of gravity of an object or closed polyline;
@8 @9	X Y	Coordinates of the origin of a drawing (short values);
$@		Format string for the conversion of digits to text (F8.2), (I3), (F12.6), etc. Corresponds to the Fortran Format command.
$@0		Name of the active drawing
$@1		Name of the active object
$@2		Name of the previously loaded symbol table
$@3		Text belonging to an identified text reference point (set for ALT 3 and Text Information)
$@4		Name of digitizer assignment to pad/screen
$@9		Previous text entry
$@AWV		Selected method to develop sheet metal, e.g. F = Faktorenverfahren (DIN). The result ist identical to the ID in the file method.DAT. *method* is name of the corresponding DIN, for example DIN 6935.DAT (*MAKROABW* directory*)*
@BMH		Information about projection lines, set to 1,2,3 or 11,12,14 for dimension if the dimension line is faded out or the z-dimension switch set.

3-D Variables

Z8 Z9 ZB	X-coordinate of edge start point or isolated point Y-coordinate of edge start point or isolated point Z-coordinate of edge start point or isolated point
ZBTP	Polar surface moment of inertia
ZBWY ZBWX	Moment of resistance ref. x-SP-axis (SP = centre of gravity) Moment of resistance ref. y-SP-axis
ZC ZD ZE	X-coordinate edge end point Y-coordinate edge end point Z-coordinate edge end point
ZCTY	Type of curve 0-plane, 1-circle, 2-ellipse, 3-freeform
ZEDZ	Number of adjacent edges $\geq$ 0
ZFCZ	Number of adjacent identified facets $\geq$ 0
ZFF.	Freeform geometric variables
ZFFI	Indexing for body with freeform surfaces is 0=not active, 1=active
ZFFJ	-1 or 0 An indexed body with freeform surfaces is not present, or indexing cannot be used. >0 Index body with freeform surfaces
ZFFN	Default generation of a dummy body for freeform surfaces is 0 not executed/not required 1 required
ZFSX ZFSY ZFSZ	X-coordinate contour line or surface centre of gravity Y-coordinate contour line or surface centre of gravity Z-coordinate contour line or surface centre of gravity
ZFTR ZFTY	0 or 1, if triangulation is required during part modelling Surface types: 0-plane, 2-sphere, 3-cylinder, 4-cone, 5-torus, 6-freeform surface, <0-concave surface
ZHKV	Hyper-edge 0=exists, 1=does not exist
ZIAS	Number of the current view
ZIAZ	Number of defined views
ZKTY	Component category of a body
ZKV.	Variable for pre-setting composite edges
ZKZL	Composite edge lengths
ZKZE	Linear connection of composite edges 0 = equal number of points in both c-edges 1 = diverse number in c-edges
ZKZN	Number of points in composite edges
ZKZV	-1 Interactive query when joining c-edges 0 Shortest connection 3 Start -> Start 1 End -> Start 4 Start -> End 2 End -> End 5 Close open c-edge
ZMAX ZMAY ZMAZ	X-coordinate maximum point bounding box Y-coordinate maximum point bounding box Z-coordinate maximum point bounding box
ZMIX ZMIY	X-coordinate minimum point bounding box Y-coordinate minimum point bounding box

ZMIZ	Z-coordinate minimum point bounding box
ZMPX ZMPY ZMPZ	X-coordinate centre point 3D Y-coordinate centre point 3D Z-coordinate centre point 3D
ZOFL	Contour line, surface, solid surface
ZOTY	Type of object to be identified 1 wireframe 2 supporting polygon (FFS) 0 Other
ZPAX ZPAY ZPAZ	X-coordinate start point vector menu Y-coordinate start point vector menu Z-coordinate start point vector menu
ZPEX ZPEY ZPEZ	X-coordinate end point vector menu Y-coordinate end point vector menu Z-coordinate end point vector menu
ZPRO	Number of active project (Database)
ZRA1 ZRA2	Circle, sphere, cylinder radius, major torus radius Minor torus radius
ZSUB	Flag for part generation 0-main part generation 1-sub-part generation
ZYAX ZYAY ZYAZ	X-coordinate axis vector (cylinder, cone, torus) Y-coordinate axis vector (cylinder, cone, torus) Z-coordinate axis vector (cylinder, cone, torus)
ZVKX ZVKY ZVKZ	X-coordinate vector with value in vector menu Y-coordinate vector with value in vector menu Z-coordinate vector with value in vector menu
ZVNX ZVNY ZVNZ	X-coordinate contour line or normal vector to surface Y-coordinate contour line or normal vector to surface Z-coordinate contour line or normal vector to surface
ZVOL	Solid bodies
ZVSX ZVSY ZVSZ	X-coordinate of solid centre of gravity Y-coordinate of solid centre of gravity Z-coordinate of solid centre of gravity
ZWKG	Cone angle
ZXP0	Polyhedral or Analytical Model -1 undefined model 0 polyhedral 10 analytical model (precise) 11 analytical model (approximated)
ZXP2	Active model 0=analytical, 1=polyhedral
ZPX3	Polyhedral approximation
ZXPC	Degree of freedom for curves on surface and FFS curves
ZXPG	Generation of mesh for polyhedral approximations and freeform surfaces 0 not for planar freeform surfaces 1 also for planar freeform surfaces
ZXPO	Approximation of curves on surfaces -1 only for planar freeform surfaces 1 also between mesh lines
ZXPU	Precision of n-mesh (FFS)

ZXPV	Precision of m-mesh (FFS)
Z3DK	Status variable for automatic macro start (2D/3D toggle) 0 no macro start 1 generic cylinder 2 solid of revolution 3 bore/cut 4 subtract 5 slice 6 sectional view

2.4 Arithmetic Expressions

Optional arithmetic expressions can be used for numerical entries. These expressions can then be linked to logical conditions.

Under **arithmetic expression,** we understand a valid algebraic construction consisting of numeric constants, numeric variables, arithmetic operators and parenthesis having maximum length of 60 characters. Only round brackets, no winged or square parenthesis, are accepted. Brackets may be nested but must be paired, i.e. all open brackets must be closed.

2.4.1 Arithmetic Operators

The following operators are valid:

+	Addition
-	Subtraction
*****	Multiplication
/	Division
^	Exponential function

In place of the ^ operator you can use the function name, XHY (x to the power of y). The evaluation of arithmetic expressions conforms to the usual mathematical rules.

2.4.2 Basic Functions

All relevant mathematical and string functions may be used.

ABS	Absolute value
ACOS	Inverse cosine function
AINT	Integer portion of an expression
ARC	Conversion of radians to degrees
ASC	ASCII character code
ASIN	Inverse sine function
ATAN	Inverse tangent function
COS	Cosine function
COSH	Hyperbolic cosine function
EXP	Exponential function
GRD	Conversion of radians to degrees
LEN	String length
LOG	Natural logarithm
LOG10	Common logarithm (base 10)
NINT	Next integer (rounded to next whole number)
SIG	Signum (sign function: -1, 0 or 1)
SIN	Sine function
SINH	Hyperbolic sine function
SQR	Square
SQRT	Square root function
TAN	Tangent function
TANH	Hyperbolic tangent function
VAL	Converts string to numerical value

Example: %C:=SQRT (A*A + B*B) ⟶ $A^2 + B^2$

Although values in trigonometry functions are normally stated in radians, The way HiCAD's internal formula interpreter processes entries or results in degrees, depends on the given angle unit.

The functions ARC and GRD can be used to convert degrees to arc dimensions.

2.5 Logical Comparisons in Expressions

Expressions used for **logical comparisons** can be understood as a simple arithmetic comparison, e.g.

a1 **op** a2

whereby a1 and a2 express constant, variable or arithmetic expressions linked by a relational operator, i.e. **op**. Any of the following relational operators may be used:

=	equals	< >	unequal
<	less	>	greater than
<=	less than or equal	>=	greater than or equal

A logical comparison can also be constructed from two simple arithmetic comparisons plus the Boolean operators **AND** and **OR**. The operator **NOT** returns the logical negation of a Boolean expression.

Logical comparisons are important when, e.g. conditional value assignments are specified for variables.

To demonstrate this we show the following examples using the variables shown below:

%a1:=5
%a2:=1
$TEXT:=ABCDEF

Example 1:

```
IF a1 <= 5 THEN
.
.
IFEND
```

a1 has the value 5, therefore the logical expression "a<=5" is true, i.e. all commands preceding IFEND are executed.

Example 2:

```
WHILE a2 < 10
.
.
%a2:=a2+1
WHEND
```

He loop variable is incremented by 1 at the end of each executed loop. This means that the logical expression "a2<10" is true for the first 9 loops. The loop is then ended.

Example 3:

```
IF NOT a1 <= 5 THEN
.
.
IFEND
```

a1 has the value 5, therefore the logical expression "NOT a<=5" is false, i.e. commands preceding IFEND are not executed.

Example 4:

```
IF $TEXT="ABCDEF" THEN
.
.
IFEND
```

As TEXT has the value ABCDEF, the logical expression is true, i.e. all commands preceding IFEND are executed.

Example 5:

```
IF $TEXT(1:3)="ABC" THEN
.
.
IFEND
```

As the first three letters of TEXT conform to the value of the character string ABC, the logical expression is true, i.e. all commands preceding IFEND are executed.

2.6 Logical Variables

In addition to logical expressions, HiCAD enables you to use logical variables. These are especially useful in loops and IF statements. The following logical variables (TRUE and FALSE) are available:

3D	TRUE, when the macro starts in 3-D mode.
BEMA	TRUE, when dimensioning should be shown
DVORHD	TRUE, when the data record exists in accessed file
ESC	TRUE, when the END or ESC key is used.
FEATURE	TRUE, when a feature protocol existis for the active 3-D part. It is considered that the protocol is subject to a superior part (Example: Flange)
FEHL	TRUE, when an error occurs, e.g. in a POINT instruction
INT	TRUE, when the INT key is used
ISOP	TRUE, when FIXED POINTS mode is switched ON.
JA	TRUE, when the response to a Y/N query is YES
NEIN	TRUE, when the response to a Y/N query is NO.
PBEZ	TRUE, when point designation is active.
PESC	TRUE, when a point specification is confirmed with END.
PINT	TRUE, when INT is selected during a point specification.
SCHR	TRUE, when the HATCHING O/X toggle is set to ON.
SYMB	TRUE, when SYMBOLS O/X is switched ON
TEXT	TRUE, when the TEXT O/X toggle is switched ON.
VALD	TRUE, when global line representation mode is switched on.

VORHD	TRUE, when the accessed file exists. TRUE, when the selected component exists. TRUE, when the there is sufficient space between projection lines for the dimension. TRUE, when a numeric variable has already been set (WERT command).

** Please refer to the list of reserved words attached to this document.

Example 1

```
ANTWORT #
IF JA THEN
.
.
IFEND
```

If the response to the Yes/No query is YES, then the logical variable JA is true, i.e. the IF condition is fulfilled and the commands preceding IFEND are carried out.

Example 2

```
IF NOT VORHD GOTO 99

.
.
99:END
```

If a new component is selected, e.g. by name, and the given component is not present, then the macro is ended.

Example 3

```
IF FEATURE THEN

.
.
IFEND
```

If a feature protocol exists for the active part, then the logical variable FEATURE is TRUE, i.e. the IF condition is fulfilled and the commands preceding IFEND are carried out.

- Please refer to the list of reserved words attached to this document.

2.7 String Expressions

Alphanumeric variables and system variables can also be used in expressions. The interpreter decides whether alphanumeric entries are interpreted as string variables or string constants. Numeric variables can be converted to alphanumeric variables. Function calls made with string variables can return numeric values.

2.7.1 Convert Numeric Variable to String

If a numeric variable is assigned to a string variable in the macro, the content of the numeric variable is converted to a string, e.g.

$A:=%B

In this procedure, the system variable $@ is of particular significance as the content of the variable is then understood as a FORTRAN format and used for the conversion. The format string must be set in closed, round parenthesis. It may contain format instructions as well as a numeric format, e.g. auxiliary text in Hollerith format. The variable $@ can be useful when used for *notations* in drawings.

If the variable $@ is, for example, assigned to the string

$@:=(4HFl.: , F8.2 , 5H m^^2)

and the content of system variable ZO inserted as text in drawing, then it is output as:

Fl.: 148.25 m^2

The assignment

$@:=(I4)

for example, creates a four digit whole number.

The system variable $@ should be subsequently deleted.

If a string begins with the character %, but is not followed by a variable name, it is taken over without alteration as a string constant.

2.7.2 String Operations

In HiCAD strings can be linked with the operator **+**, whereby a space *must not* exist between the operand and the operator.

Example:

$A:=Text1

$B:=Text2

$TEXT:=ABCDEF

$C:=$A+$B	→	Text1Text2
$C:="This is "+$A	→	This is Text1
$NAME:=$TEXT(4:5)	→	DE

2.7.3 String Functions

HiCAD offers the following functions:

ASC(string) ASCII *code* of the initial string character

LEN(string) String length. The string can be a string constant or a string variable. A numerical value is returned.

CHR$(num) ASCII character set if the integer value of the numeric expression *num* lies between 0 and 255.

VAL(string) Assuming that the content of the string can be numerically interpreted, this function converts the string to a numerical value. It is possible that only part of the string can be used.

LTU$(string) String conversion in upper case. The string can be a string constant or string variable.

UTL$(string) String conversion in lower case.

IDX($A,$B) Find sub-string. Searches for the content of string variable $B in string variable $A. If $B is detected in $A, the function returns the start index of $B, i.e. the start position of the sub-string. The value 0 is returned if $B is not found in $A. Searches cannot be continued if a space is detected. If string constants are used instead or variables, they may not contain commas.

TIM$ the current time as: HH:MM:SS. Returns a string.

DAT$ the current date as: YY:MM:DD. Returns a string.

Example:

Assuming the variables: *$A:=TEXT, %A1:=84, $B:=EX*

Then:

%I1:=LEN($A)	→	4
%I1:=ASC($A)	→	84
$C:=CHR$(A1)	→	T
$C:=TIM$	→	13:52:18
$C:=DAT$	→	95:11:24
$C:=LTU$($A)	→	text
%I1:=IDX($A,$B)	→	2

2.8 Variable Memory

The Variable Memory functions enable you to define new user variables or delete them from variable memory.

Select **Information** and **Data structure, test and variable Memory** from the menu bar and pick

 **Delete Variable Memory**

If you respond with **yes**, all user-defined variables are deleted from variable memory.

If you want to define new variables, select **Information** and **Data structure, test and variable Memory** from the menu bar and pick

- **Set numerical variables,**
- **Set string variables,**
- **Set point varaibles** or
- **Set line variables.**

3 Macro Language Commands

Macros are written in the HCGS (HiCAD Graphical Language) macro language. Language elements are specially designed to interact with HiCAD. In the following section, HCGS commands are described in alphabetical order. Commands have the following structure:

1. Command (keyword)

Although the complete command name is always generated for the macro, only the first 3 positions are significant.

2. Function

A short description of the command.

3. Syntax

Dictates the form in which commands and arguments are written .

4. Argument

A list of all valid arguments. If arguments are separated by "/", only one of them is used. Arguments in square parenthesis are freely definable. Arguments written in high-case are fixed and must be identically entered; arguments written in lower case are flags (e.g. variables). Separators must be used between arguments as well as between arguments and the command. The REM command is an exception to this rule.

The REM command always lists special cases that should be considered in arguments. A detailed description of diverse arguments can be found in Section 2 of this manual.

5. Note

A detailed reference to the current command.

6. Macro Generation (HiCAD's macro creation system)

The commands that should be automatically carried out during macro creation are specified here. If commands cannot be automatically generated, they can be inserted with a text editor.

7. Example

Here you will find examples of applications for the current command.

Notes:

- Each command in the macro must start on a new line and may not extend beyond it. Spaces are ignored.
- The Macro examples in this manual are generated without using the HELiOS PDM functions.

3.1 Commands

Keywords	***Significance***
APEIN, APAUS	Autopilot on/off
ANTWORT	Response to Yes/No queries
CALL	Sub-macro call
CLOSE	Close file
COPY	Copy command in HCMT
DEL	Delete variables
DISTANZ	Specification of a distance relative to the current point
ECHO	Displays a freely definable user message in the text box
END	Macro end
FOR ... NEXT	Loop with a specific number of repetitions
GOTO	Jump instruction
IF ... IFEND	IF query
IGNORE	Ignores commands
INPUT	Read from file
INTEGER	Entry of a whole number
MAKRO	Calls a subordinate macro
MAUS , MEIN	C-edge colour-marking on/off
MKDIR	Creates a directory
OPEN	Opens a file
OPTION	Selects a menu option
OUTPUT	Write to file
PFD	Path detection
POINT	Point specification
REAL	Entry of a real number
REM	Remark
REPEAT .. UNTIL	Conditional loop

Keywords	***Significance***
SAUS, SEIN	User prompts on/off
START	Macro start
SZEIN, SZAUS	Status line update
STRING	Text entry
UDA, UDE	3D undo-save on/off
VAI	Integer variable entry
VAR	Variable entry
WAIT, WARTE	Pause
WAUS, WEIN	Pause on/off
WERT	Checks whether a numeric variable is available
WHILE ... WHEND	Conditional loop
WINKEL	Angle value
ZAE, ZAA	Cancel screen format
%variable:=	Value assignment to numeric variables
$variable:=	Value assignment to string variables

3.1.1 Antwort

Function

Alternative responses to Ja/Nein (yes/no) query

Syntax

ANTWORT JA / NEIN / # / ESC / 0 / 1

Arguments

JA/1	*see fixed arguments*
NEIN/0	*see fixed arguments*
#	*see fixed arguments*
ESC	*see fixed control arguments*

Notes

The ANTWORT command enables responses to YES/NO queries.

RET	⇒	ANTWORT 0
YES	⇒	ANTWORT 1
NO	⇒	ANTWORT 0

Please take great care when using free arguments # !

Generation with HiCAD macro creation system

The command is automatically generated whenever HiCAD expects a response to a Ja/Nein (yes/no) query.

Example

The following macro saves the current drawing, opens a new one, thereby expecting you to specify its name, and activates the **Process ZTL** menu.

```
REM     HiCAD-NEXT
START   59
REM   HiCAD   3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE   2 = Load
OPTION   2 101
REM     Save active scene ?
ANTWORT 1
STRING  RET
OPTION  ESC
END
```

The following macro deletes the active object. The deletion of the object is confirmed with JA (*yes*). In this case the free argument, #, is used instead of 1, a query is then output.

```
REM    HiCAD-Next
START   59
REM   HiCAD   3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE   3 = Process
OPTION   3 101
REM   PARTS   7 = Delete
OPTION   7 102
REM     O.K. ?
ANTWORT #
END
```

3.1.2 APAUS / APEIN

- **Function**

  Autopilot on/off toggle

- **Syntax**

  APAUS

  APEIN

- **Argument**

  None

- **Generation with HC-MES**

  Automatic generation is impossible.

- **Example**

  The following macro creates a rectangle. The auto-pilot is switched off before the first corner point is specified and switched on again after the second point has been set.

```
REM   LINES
START   6
REM   LINES   4 = Rectangle
APAUS
OPTION   4   6
POINT   #
POINT   #
APEIN
END
```

3.1.3 CALL

- **Function**

  This function executes a sub-macro called from the current macro and then returns to the current macro.

- **Syntax**

  CALL *filename*

- **Argument**

  filename
  The *filename* must represent a valid macro containing HCGS commands.

- **Generation with HC-MES**

  Automatic generation is impossible.

- **Example**

  This macro first loads a drawing, without saving the current drawing, and then calls (CALL command) the **TEXT** macro. This macro asks whether or not text should be shown. You are then returned to the current macro.

```
REM     HiCAD-Next
START   59
REM   HiCAD   3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE   2 = Load
OPTION   2 101
STRING  *
OPTION  ESC
REM   BERECHNUNG   2 = Eigene Verf.
OPTION   2 159
call c:texte
REM   DRAWINGS   3 = Process ZTL
OPTION   3   1
REM   COMPONENTS 8 = New main part
OPTION   8   2
REM   NEW PART   1 = Create part
OPTION   1   3
STRING  RET
REM   COMPONENTS   1 = LINES
OPTION   1   2
REM   LINES   2 = Polyline
OPTION   2   6
POINT    #
END
```

The "TEXT" macro

```
Makro TEXT
REM     HiCAD-Next
START   59
REM ALT1
OPTION  21   0
REM   REDRAW   6 = Text     O/X
OPTION   6  51
REM     Display Text ?
ANTWORT #
OPTION  ESC
REM ALT1
OPTION  21   0
REM   REDRAW   1 = all parts
OPTION   1  51
END
```

3.1.4 COPY

Function

Copies files

Syntax

COPY *filename1 filename2*
COPY *$variable1 $variable2*

Arguments

filename1 filename2
fileiname1 represents the name of the file to be copied, and *filename2* the name of the copy. A path specification or file group, according to the FILEGRUP.DAT file, can precede the file name. The COPY command has a total length of 60 characters.

Cf. MKDIR and PFD

$variable1 $variable2
In this case, the file name (if appropriate with path) is read from the text variables *$variable1* and *$variable2.* These variables may contain 60 characters.

Generation with HC-MES

Automatic generation is impossible.

Example 1

The file TEST.DAT should be copied from the C:\HICADNEXT\SZENEN directory to the directory E:\DATEN.

```
REM     HiCAD-Next
START   1
$A:=c:\hicadnext\SZENEN\TEST.DAT
$B:=e:\Daten\TEST.DAT
COPY $A $B
END
```

Example 2

The file TEST.DAT should be copied from directory C to file group Z. File groups are defined in the FILEGRUP.DAT file in the HiCAD *sys* sub-directory.

```
REM     HiCAD-Next
START   1
$A:=c:TEST.DAT
$B:=Z:TESTNEU.DAT
Copy $A $B
END
```

3.1.5 DEL

Function

Deletes numeric and text variables

Syntax

DEL *%variable*
DEL $*variable*

Argument

%variable or *$variable*

Note

Please remember that it is also possible to delete system variables with this function.

Generation with HC-MES

Automatic generation is impossible.

Example

```
REM    HiCAD-Next
START   1
$A:=Macro technique
$B:=technique
%I1:=IDX($A,$B)
OPEN c:Test
OUTPUT %I1
CLOSE
DEL $B
DEL %I1
END
```

3.1.6 DISTANZ

Function

Distance specification, relative to an identified point

Syntax

DISTANZ *real constant* / *arithm. expression* / *RET* / # /*ZEI* / *END*

Arguments

RET The suggested default value (fixed argument) is taken over when you run the macro. This argument is automatically set if you take over the offered default value during macro creation.

\# The value is queried (free argument) during the macro run.

ZEI This argument is taken over if **z** is entered during macro creation, i.e. the distance is derived from the drawing. The next macro command is then:

OPTION *number* 23

This is equivalent to the activation of the **Distance** function in the **Information** menu. From now on the command sequence depends on *number*. In a macro, ZEI simulates the actions taken when you specify "z" at the appropriate time during a normal HiCAD session.

Note

The DISTANZ command is generated whenever the message *DISTANZ* appears in the text box. Normally, this is when a new reference point is defined for the point options, R, P, PX, and PY.

Generation by HC-MES

The command is automatically generated whenever a distance specification is required.

Example

In this example, the y-distance of the line to be drawn is taken over from the drawing.

```
REM           COMPONENTS
START    2
REM           COMPONENTS           1 = LINES
OPTION   1   2
REM            LINES               2 = Polyline
OPTION   2   6
POINT   #
POINT   R # z
REM     X distance:
DISTANZ 45
REM     Y distance:
DISTANZ ZEI
REM           DISTANCES            6 = Length GE
OPTION   6  23
POINT   #
REM     Set negative distance (Y/N) ?
ANTWORT 0
POINT   ESC
END
```

3.1.7 ECHO

Function

Shows user prompts in the text box

Syntax

ECHO [*user prompt]*

Argument

user prompt
This string is output in one line without a carriage return, i.e. the next output in the text box is set directly behind the user prompt. This means that it is impossible to specify a maximum length for *user prompt* as the length of the subsequently issued text is unknown. However, if the text is too long, the screen can be re-structured when the macro run has been completed.

It is also possible use the HiCAD user prompts contained in the MAKROTXT.DAT file. To do this, simply copy the file to the Macro directory. Individual file lines can be accessed by specifying string

$100*n*

whereby *n* is the number of the line in the MAKROTXT.DAT file, e.g.

ECHO $10021

In this case, the text contained in line 21 is displayed.

Note

In addition to normal user prompts, you can use the ECHO command to output auxiliary information. Messages output in this way can be switched off using the SAUS command, thus enabling you to use your own prompts.

Generation with HC-MES

Automatic generation is impossible.

Bracketed ECHO and VAR Commands

ECHO and VAR commands can be set in parenthesis. This enables several user prompts or the variable specifications to be displayed in a pop-up menu during the macro run:

- Remarks set in parenthesis are respected.
- The ECHO commands set in brackets are only respected if they precede the first VAR command. Any subsequent ECHO commands are ignored.
- All ECHO commands set immediately in front of brackets are respected.

Example:

The macro shown below triggers the display of the illustrated pop-up menu.

```
REM   HICAD-NT  : VN:07
REM          HiCAD-Next
START   59
(
echo Variable A = Height in cm
echo Variable B = Width in cm
var %a A
var %b B
)
OPTION  ESC
END
```

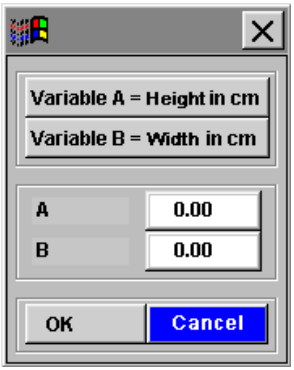


Fig. 8 VAR and ECHO commands enclosed in parenthesis

3.1.8 END

- **Function**

  Exits macro execution

- **Syntax**

  END

- **Argument**

  None

- **Note**

  Normally the END command is the last command in a macro. Any commands following END are ignored. After the macro run, if argument 1 is set, you are returned to the HiCAD menu level from which the macro was called. If no argument is set, you are returned to the menu level active when macro creation was completed.

- **Generation with HC-MES**

  The command is automatically generated when macro creation is ended.

- **Example**

  The macro frame

```
REM           DRAWINGS
START    1
END
```

3.1.9 FAA / FAE

- **Function**
  Suppresses all WAIT commands in a macro / switches WAIT on

- **Syntax**
  FAA
  FAE

- **Argument**
  None

- **Generation with HC-MES**
  Automatic generation is impossible.

- **Note**

  FAA suppresses all WAIT commands in a macro (including those that ignore WAUS). This means that no messages are output to the screen. **FAE** switches WAIT on again.

3.1.10 FOR ... NEXT

Function

Count-controlled loop, i.e. it is executed a fixed number of times.

Syntax

FOR *loop variable*:= *lower limit* TO *upper limit*
.
.
. (HCGS command)
.
.
NEXT *loop variable*

Arguments

Loop variable
A real variable, that has not necessarily been explicitly predefined. It is assigned the value *lower limit* at the beginning of the loop. When the NEXT instruction is found, the variable value of the loop is increased by 1.

Lower limit, upper limit
These values determine a specific number repetitions. Values can be integer constants, real variables or arithmetic expressions. If the result of an expression does not return a whole number, it is rounded to the next smallest whole number.

Note

You can assign values to the arguments *loop variable* and *upper limit* within the loop, whereby these arguments will naturally influence the number of times the loop is repeated. It is also possible to assign variables or arithmetic expressions to *lower limit* and *upper limit*. After each NEXT statement, the value of the *loop variable* is checked to see if it is smaller or equal than the value of the *upper limit* argument. If it is, the loop is executed again until the appropriate condition is reached. Otherwise the macro executes the instruction following the NEXT statement.

FOR...NEXT loops may be nested, whereby the inside loop is executed first.

Generation with HC-MES

Automatic generation is impossible.

Example

The following example creates 10 lines in colours available in HiCAD and then returns to default parameters.

```
REM           LINES
START    6
REM           LINES                9 = Preset Par.
OPTION   9   6
REM           LINE PARAM.          7 = Save param.
OPTION   7  18
OPTION  ESC
%a:=9
FOR %i:= 0 TO a
REM           LINES                9 = Preset Par.
OPTION   9   6
REM           LINE PARAM.          4 = Colour
OPTION   4  18
REM      Colour
INTEGER i
OPTION  ESC
REM           LINES                2 = Polyline
OPTION   2   6
POINT   A 100 100+10*i
POINT   R 100 0
POINT   ESC
NEXT i
REM           LINES                9 = Preset Par.
OPTION   9   6
REM           LINE  PARAM.         8 = Undo param.
OPTION   8  18
END
```

3.1.11 GOTO

Function

Unconditional jump instruction, i.e. the macro always executes a jump to the specified destination.

Syntax

GOTO *label*

.

label: HCGS command

Argument

label

This value indicates the destination of a jump, i.e. the command to be executed after the jump. A *label* is an identifier (name) that identifies a particular statement and can be the destination of several GOTO commands. The *label* must be a number between 1 and 9999! This number is preceded by a colon and followed by a valid macro command. A *label* is ignored without these entries, i.e. the GOTO command is not executed.

Note

As the GOTO command invokes an unconditional jump, it is always carried out. As this means that it can create an infinite loop, it should always be paired with an IF statement. Should an infinite loop be accidentally created, you need to break off the macro run. If a user entry is expected, specify END again.

Generation with HC-MES

Automatic generation is impossible.

Example

This is an example of a classic infinite loop – and should definitely not be imitated !

```
REM           LINES
START    6
90:REM           LINES                1 = Fixed points
OPTION   1   6
POINT
REM Symbol number ( 0 or 1-9999 ) (INT=graphical):
INTEGER RET
POINT ESC
GOTO 90
END
```

Cf. Example of an IF instruction

3.1.12 IF..ELSE..IFEND

Function

Conditional execution of macro commands

Syntax

IF *logical expression* THEN
.
. (HCGS command)
ELSE
. (HCGS command)
.
IFEND

Argument

Logical expression
The given expression must conform to the rules described in Chapter 2.

Note

There are two ways of using the IF instruction:

- Conditional execution with IF...GOTO
  If the given expression returns the value "true", the GOTO command is executed. If it returns the value "false", the macro executes the command following the IF statement.
- IF..THEN.. ELSE ..IFEND
  Several macro commands can be grouped in a block, thus allowing the macro to branch to an alternative instruction. If the logical expression (holding the data on which the current decision is based) of the IF statement is "true", then the entire block (THEN to ELSE) is executed. If the expression is "false", then the block following ELSE is executed. If an ELSE statement is not given, then the command following IFEND is invoked. Although IF-IFEND blocks may be nested, each block must end with IFEND.
- Value assignments can be made after single line IF commands (similar to GOTO commands). Example: if (x>100) %y:=10.

Generation with HC-MES

Automatic generation is impossible.

Example

In the following example the GOTO command is expanded by a conditional END.

```
REM           LINES
START    6
90:REM           LINES                1 = Fixed points
OPTION   1   6
POINT #
REM  Symbol number ( 0 or 1-9999 ) (INT=graphical):
INTEGER RET
POINT   #
IF pesc GOTO 99
GOTO 90
99:END
```

The next example, text parameter are dependent on the given scale. System parameter @2 is used to query given scale values.

```
REM           COMPONENTS
START    2
REM           COMPONENTS           S6= Text
OPTION  16   2
IF  @2 <= 10 THEN
%h :=3.5
%s :=2
IFEND
IF @2 >10  THEN
%h :=2.5
%s :=4
IFEND
REM           TEXT                 3 = Text parameter
OPTION   3  25
REM             PARAMETER          3 = Text height
OPTION   3  26
REM     Text height (mm):
REAL    H
REM           PARAMETER            2 = Char. type
OPTION   2  26
REM     Char. type
INTEGER S
OPTION  ESC
END
```

3.1.13 IGNORE

- **Function**
  Skips commands

- **Syntax**
  IGNORE

- **Argument**
  None

- **Note**
  If this command is found in a macro, all commands up to the next OPTION command are interactively queried.

- **Generation with HC-MES**
  Automatic generation is impossible.

3.1.14 INTEGER

Function

Requests the entry of a whole number

Syntax

INTEGER *integer constant / arithm.expression* / RET / # / ESC / END

Argument

Integer constant
The given value must conform to a function-specific value range.

Arithmetic expression
If the expression does not return a whole number, it is rounded to the next whole number.

ESC/END
The argument ESC (END) is only valid in exceptional cases, normally to close a repetitive entry. If ESC (END) is illegally entered, the macro is temporarily interrupted by an error message. In this case the macro run may be broken off.

Note

The command expects the entry of a whole number, e.g. when a selection should be made in the text window. The INTEGER command is also used if the value entry for a parameter needs to be integer, e.g. layer number.

Generation with HC-MES

If the entry of a whole number is required during macro creation, the command is automatically generated.

Example

In the extract below the layer number is set to 2, and a colour selection requested.

```
REM           LINES
START    6
REM           LINES                9 = Preset Par.
OPTION   9   6
REM           LINE PARAM.          3 = Layer number
OPTION   3  18
REM     Layer
INTEGER 2
REM           LINE PARAM.          4 = Colour
REM      Colour
OPTION   4  18
INTEGER #
OPTION  ESC
END
```

In this case the text parameter are changed and a font specification confirmed with RET.

```
REM           COMPONENTS
START    2
REM           COMPONENTS             S6= Text
OPTION  16   2
REM           TEXT                   3 = Text parameter
OPTION   3  25
REM           PARAMETER              2 = Char. type
OPTION   2  26
REM     Char. tye
INTEGER RET
REM           PARAMETER              3 = Text height
OPTION   3  26
REM     Text height (mm):
REAL    RET
END
```

3.1.15 MAKRO

Function

This command calls a sub-macro from the current macro. The calling macro is ended and the sub-macro executed.

Syntax

MAKRO *file name*

Argument

File name
The *file name* must be a valid HCGS macro.

Note

The MAKRO command invokes a *file name* macro and executes all commands contained in it. The *file name* macro must be a "stand-alone" macro. MAKRO commands may be nested, i.e. the called macro may contain a MAKRO command.

Generation with HC-MES

Automatic generation is impossible.

Example

A drawing is loaded without saving the current one. The TEXT macro, querying whether or not text should be represented, is then called with the MAKRO command. The current macro is thereby ended, i.e. all macro commands following the MAKRO instruction are ignored. In this case, the CALL command is clearly preferable to the MAKRO command.

```
REM          HiCAD-Next
START   59
HNEXT
REM   HiCAD           3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE       2 = Load
OPTION   2 101
STRING  *
OPTION  ESC
REM   BERECHNUNG   2 = Eigene Verf.
OPTION   2 159
MAKRO C:TEXTE
REM   HiCAD           2 = 2-D ZTL
OPTION   2  59
REM   DRAWINGS        3 = Process ZTL
OPTION   3   1
REM   COMPONENTS      8 = New main part
OPTION   8   2
REM   NEW PART        1 = Create part
OPTION   1   3
STRING  RET
REM   COMPONENTS      1 = LINES
OPTION   1   2
REM   LINES           2 = Polyline
OPTION   2   6
POINT    #
END
```

The TEXT macro

```
REM          HiCAD-Next
START   59
REM ALT1
OPTION  21   0
REM   REDRAW       6 = Text     O/X
OPTION   6  51
REM     Display text ?
ANTWORT #
OPTION  ESC
REM ALT1
OPTION  21   0
REM   REDRAW       1 = All parts
OPTION   1  51
END
```

3.1.16 MAUS/MEIN

Function

Switches composite edge colour marking on/off (3D)

Switches symbols marking line elements On or Off during line identifications (2D)

Syntax

MAUS
MEIN

Argument

None

Note

Normally, alternate 3D lines segments are displayed in magenta/green. You can change this setting with the MAUS/MEIN toggle. The default setting is MEIN.

Generation with HC-MES

Automatic generation is impossible.

Example

```
REM      Process
START  106
REM      PROCESS           2 = Comp. edges
OPTION   2 106
REM      3-D C.EDGE        S5= Proc. c.edge
OPTION  15 137
MAUS
REM      COMP. EDGE        4 = Mark c.edge
OPTION   4 133
POINT   #
REM     Retain marking (Y/N) ?
ANTWORT 1
POINT   ESC
END
```

3.1.17 MKDIR

Function

Directory creation

Syntax

MKDIR *directory name*

Argument

Directory name

Names should respect the normal conventions.

Generation with HC-MES

Automatic generation is impossible.

Example

The macro below creates the sub-directory Scenesbak and then copies the TEST.SZN file from the drawing directory to the new created directory.

```
REM          HiCAD-Next
START   1
MKDIR \HICADNEXT\Scenesbak
COPY \HICADNEXT\STENEN\TEST.SZN HICADNEXT\SCENESBAK\TEST.SZN
END
```

3.1.18 OPEN INPUT .....CLOSE
OPEN OUTPUT ..... CLOSE

Function

Reads/Writes ASCII files

Syntax

Read file	**Write file**
OPEN *file name*	OPEN *file name*
INPUT *variable*	OUTPUT *variable*
INPUT *variable*	OUTPUT *variable*
.	.
.	.
.	.
CLOSE	CLOSE

Argument

File name
This file name represents the name of the file from which or to which data should be read or written. If necessary a file can be created for write, but existing files can only be read. A path can be set in front of the file name. The file name extension must always be **.DAT**.

variable
This argument determines the data to be read from or written to a file. This can be a numerical value, text, data held by a variable or results returned by an expression.

Notes

Read

The OPEN command opens the given file. Numerical values, text or point coordinates can then be read. These values, text, or coordinates are assigned to respective variables by the INPUT command. The first INPUT instruction reads the first line of the file, while the second INPUT instruction reads the second line.... and so on. The appropriate INPUT command assigns the line content to the given variable. If it is not possible to allocate a value to a variable, the logical variable VORHD (present) is set to FALSE, otherwise it is set to TRUE. This variable can be used to detect the end of an ASCII file.

As the variable type in an INPUT instruction can change, the file must be appropriately structured. Point and line variables are special cases. If an INPUT instruction is used to assign coordinates to these variables, the values should be inserted, separated by at least one space, in the appropriate line of the ASCII file.

When all required values have been read, the file must be closed with the CLOSE instruction. An open file is automatically closed when another file is opened or macro execution ended.

Write

The OPEN command first opens the file to be created for "Write" data. The OUTPUT command enables the content of variables, results returned by expressions as well as fixed values or text to be written to a file. Analogue to "Read", this data is also written line-wise. Point and line variables are again special cases. Coordinates are written in one line, whereby individual coordinate values are separated by a space. The file must end with a CLOSE command.

Simultaneous write and read operations on an opened file are prohibited.

Generation with HC-MES

Automatic generation is impossible.

Example

The string variables A1 and A2 are read from the BSP.DAT file.

```
REM           DRAWINGS
START    1
OPEN C:BSP
input $A1
input $a2
close
END
```

BSP.DAT file

```
Text1
Text2
Text3
Text4
```

After the next macro run, the variables hold the following data:

A1=Text1 and A2=Text2

Data held by point variables P0,P1 and P2 should be written to the BSP1.DAT file.

P0=100 100
P1=200.75 300
P2=400 570

```
REM           DRAWINGS
START    1
OPEN C:BSP1
OUTPUT P0
OUTPUT P1
OUTPUT P2
CLOSE
END
```

BSP1.DAT file

```
100 100
200.75 300
400 570
```

3.1.19 OPTION

- **Function**

  Selection of a menu option from the currently active menu level.

- **Syntax**

  OPTION *menu number*/ ESC *menu level*

- **Argument**

  menu number

  The function menu number is activated from the active menu level.

If *menu number* should be a number between 21-26, then one of the special functions (ALT + 1 - 6) is accessed. The following allocation is made by *menu number* and a special function (only the first six special functions are valid in macros):

Menu number	**Special Function Keys**
21	Characters (ALT 1)
22	Zoom (ALT 2)
23	Information (ALT 3)
24	Print screen (ALT 4)
25	Grid (ALT 5)
26	Variable (ALT 6)

In this case the 2nd argument is *menu level* 0.

ESC (END)

The control argument ESC (END) invokes a jump to the next superior level in menu structure. The argument *menu level* is deactivated.

Menu level

Each menu has a menu level number. In the OPTION command this number specifies a menu level from which the *menu number* function was selected. This does not mean that normal hierarchical selection procedures need not be observed. It is not possible to jump within the menu structure, e.g. from Menu 1 - **Drawing** to Menu 6 - **Lines**, the intermediate Menu 2 – **Components** must first be called.

Menu number is automatically generated when the OPTION command is found.

Note

This is the most powerful command in the HCGS macro language. It invokes the *menu number* function from the currently active menu level. Usually it is the 2nd command directly following the START instruction. **The number of the *menu-level* must, for the first OPTION command, correspond with the argument of the START instruction.** If, after calling a menu, you find yourself again on a menu level, a further OPTION command must be given. If a specification via the text box is expected, a input command from the groups **scalar or geometric entry** must follow. With the argument ESC you jump one level up within menu structure. If you are already on the top level, the command has no effect. The same applies if the *menu level* does not correspond with the currently active menu.

Generation with HC-MES

The command is automatically generated when a function is selected.

Example

In this example a new drawing is created without changing the drawing attributes. Subsequently, a new component is created, and the **Polyline** function activated.

```
REM HiCAD
START   59
HNEXT
REM   HiCAD        3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE    1 = Create new
OPTION   1 101
STRING  #
REM     Change ?
ANTWORT 0
OPTION  ESC
REM   BERECHNUNG   2 = Eigene Verf.
OPTION   2 159
REM   DRAWINGS     3 = Process ZTL
OPTION   3   1
REM   COMPONENTS   8 = New main part
OPTION   8   2
REM   NEW PART     1 = Create new
OPTION   1   3
STRING  #
REM   COMPONENTS   1 = LINES
OPTION   1   2
REM   LINES   2 = Polyline
OPTION   2   6
POINT   #
END
```

3.1.20 PFD

Function

Detects a HiCAD path in accordance with the FILEGRUP.DAT file

Syntax

PFD *file group*

Argument

file group

The name of a file group should be entered. The function then returns the path, to which this file group is assigned in the FILEGRUP.DAT file. The detected path is assigned to the system variable $@9.

dateierweiterung

When a file extension is entered, the variable **$@9** is occupied by the appropriate Windows path for the given file type. The command, *PFD &:.SZN* returns, e.g. the relevant path for scene files.

Generation with HC-MES

Automatic generation is impossible.

Example

In this case, the path for file group c: is detected and assigned to the variable $A.

```
REM          HiCAD-Next
START   1
PFD c:
$A:=$@9
END
```

In this case the path of the scene directory is assigned to the variable $A.

```
REM   HICAD-Next VN:1203
START   1
PFD &:.SZA
$A:=$@9
END
```

3.1.21 POINT

Function

Specifies a point option when required

Syntax

POINT *point option* / # / ESC / END / LLL / LLA

Argument

Point option

Only HiCAD point options that determine unique points without additional point/line specifications can be freely used as arguments.

As the options:

A, K, R, P, N, L

are needed for the unique specification of coordinate values, they can be appended, separated by a space, to the point option as auxiliary arguments, e.g.

POINT A 100.0 155.55

Apart from constants, arithmetic expressions can also be used.

Please remember, in macros the K option is normally preferable to A.

All remaining point options are only valid for macros in conjunction with previously set (ALT function) P0 to P9 point variables and line element variables L0 to L9, **Variable Memory** or the VAR command.

If point options

I, S, S2, M , M2, F, T, O

are specified during macro generation with HiCAD's Macro Development System, a free argument, #, is entered for the POINT command.

LLL LLA	These arguments are only allowed in conjunction with POINT commands. They correspond to the entry Delete Last Line (LLL) and Undo Last Deletion (LLA) and are automatically entered as arguments if used during macro creation.

Notes

The point option $X (Snap Radius) is not logged during macro creation.

If a special prompt should be issued, e.g. for a point specification in the macro, it must be set in front of the appropriate point entry command.

Generation with HC-MES

The command is automatically generated when a point specification is required.

Example

In this example the point variables P0 and P1 are defined and used to determine a follow-on point for the Polyline function.

```
REM           LINES
START    6
REM           LINES                2 = Polyline
OPTION   2   6
POINT   #
POINT   A 0 100
POINT   P 45 50
POINT   ESC
REM ALT6
OPTION  26   0
REM     Delete variable storage (Y/N) ?
ANTWORT 0
REM     User def. num. variables :
STRING  ESC
REM     User def. text variables :
STRING  ESC
REM     Point :
INTEGER 0
POINT   #
REM     Point :
INTEGER 1
POINT   #
REM     Point :
INTEGER ESC
REM     Graphical element :
INTEGER ESC
REM           LINES                2 = Polyline
OPTION   2   6
POINT   p0
POINT   p1
POINT   ESC
END
```

3.1.22 REAL

Function

Entry prompt for a real number

Syntax

REAL *real constant* / *arithm.expression* / RET / #

Arguments

RET The displayed default value (fixed argument) is taken over during macro execution.

The value (free argument) is queried during the macro run.

Note

The REAL command corresponds to the INTEGER command, with two exceptions, i.e. a real number needs to be entered instead of a whole number, and the control argument ESC is invalid.

Generation with HC-MES

If the entry of a real number is requested.

Example

The macro below, changes the scale of a drawing.

```
REM          HiCAD-Next
START   59
REM ALT2
OPTION  22   0
REM   ZOOM   1 = Zoom factor
OPTION   1  52
REM     Zoom factor =
REAL    #
POINT   A 50  50
END
```

3.1.23 REM

Function

Inserts remark lines

Syntax

REM [*character sequence*]

Argument

character sequence
The length of the string *character sequence* is limited to the length of a screen line, i.e. normally 80 characters. This is the only HCGS instruction than need not be separated from the command by a space.

Note

REM commands do not influence macro execution. They are only intended for user orientation and may be inserted anywhere in the macro.

Generation with HC-MES

A remark line is, with the exception of the POINT command, set in front of each automatically generated command. Remarks for the OPTION command are divided in two. The first part holds the menu designation, the second the selected function. User prompts for all other commands are taken over from the text box.

Example

In this case, the command OPTION is expanded with the function **Set Colour**. The colour is set to green.

```
REM          HiCAD-Next
START   59
REM   HiCAD       3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE   1 = Create new
OPTION   1 101
STRING  #
REM     Modify ?
ANTWORT 0
OPTION  ESC
REM   BERECHNUNG   2 = Eigene Verf.
OPTION   2  59
REM   DRAWINGS    3 = Process ZTL
OPTION   3   1
REM   COMPONENTS  8 = New main part
OPTION   8   2
REM   NEW PART    1 = Create part
OPTION   1   3
STRING  #
REM   COMPONENTS  1 = LINES
OPTION   1   2
REM   LINES       9 = Preset Par.
OPTION   9   6
REM   LINE PARAM. 4 = Colour
OPTION   4  18
REM     Colour
INTEGER 1
OPTION  ESC
REM   LINES       2 = Polyline
Option 2 6
Point   #
END
```

3.1.24 REPEAT

Function

The execution of a conditional loop (with HCGS commands) is repeated until the condition is fulfilled. As the condition is tested at the end of the loop, each loop is executed at least once.

Syntax

REPEAT
.
. HCGS command
.
UNTIL *logical expression*

Argument

logical expression
In contrast to other HCGS commands, the Argument does not immediately follow the command but is contained in the auxiliary command marking the loop end. Otherwise the conditions of the WHILE command are respected.

Note

With the exception that loop conditions are queried at the end of the loop, the REPEAT command corresponds closely to the WHILE command. REPEAT is often used with functions enabling multiple point entries.

Generation with HC-MES

Automatic generation is impossible.

Example

This macro corresponds more or less to the HiCAD macro allocated to the **Polyline** function.

```
REM          LINES
START    6
REM          LINES                2 = Polyline
OPTION   2   6
REPEAT
POINT #
UNTIL PESC
END
```

3.1.25 SAUS / SEIN

Function

Enables or disables the output of user prompts to the text box as well as component calculation.

Syntax

SAUS
SEIN

Argument

None

Note

- When a macro is executed, user prompts are usually shown in the text box for all functions used in the macro. This, especially with long macros, considerably slows up execution. You can use SAUS to disable and SEIN to enable text output. It is, for instance, necessary to activate it when working with "free" entries (#) in order to display the appropriate entry requests. The output of ECHO and WAIT commands are not influenced. SEIN need not be set at the end of a macro, as text output is automatically re-activated.

  A sub-macro CALL automatically sets SAUS back to SEIN.

- If a processing plane is active (in HiCAD) when a recessed feature is created, dynamic mode is enabled as soon as the depth specification is expected, i.e. the depth can be specified dynamically by moving the mouse. Enter RET to take over or change the displayed value.

  When recessed features are created by a macro, HiCAD also switches to dynamic mode during macro execution, but in this case the depth value specified in the macro is always taken over. You can use the macro command SAUS to prevent this. It also works during the creation of generic cylinders.

Generation with HC-MES

Automatic generation is impossible.

Example

This macro corresponds to the example used for the REPEAT command, except that the user prompts are switched off.

```
REM           LINES
START    6
SAUS
REM           LINES                2 = Polyline
OPTION   2   6
REPEAT
POINT #
UNTIL PESC
SEIN
END
```

3.1.26 START

Function

Macro start

Syntax

START *menu level*

Argument

Menu level
This argument belongs to the numerical group, whereby, in this case the following menu numbers are only valid as the start point of an executable macro.

59	Main menu		
1	Drawing	**101**	Scene (3D)
2	Component (2D)	**102**	Part (3D)
6	Lines	**106**	Process (3D)

Note

The procedure generated by the START command enables a macro to be called from any level of the hierarchically structured HiCAD menu tree. Start procedure automatically goes to the menu level active when the macro was created. The START command must be on the first line (with the exception of a REM command) and may only be used once in a macro.

Generation with HC-MES

The command is automatically generated when the **Create Macro** function is called.

Example

```
REM           DRAWING
START    1
.
```

```
REM           COMPONENTS
START    2
.
```

```
REM           LINES
START    6
.
```

3.1.27 STRING

Function

Character sequence entry request.

Syntax

STRING *character sequence* / *variable* / RET / # / INT/ ESC

Argument

character sequence
The length of the string depends on the type of entry expected, e.g. Text, a maximum of 60 characters is valid for notations, but only 8 for component names. If, for example, string length exceeds 8 characters, the name is automatically shortened to the maximum permitted length.

variable
Both string and real variables are allowed. However only one variable can be used in an argument. Real variables, beginning with % are automatically converted to a string. The format variable $@ is thereby evaluated.

RET	The given default value (fixed argument) is taken over during macro execution.
#	The value is queried (free argument) during macro execution.
INT	This argument can be used, e.g. to specify objects by identifying lines.
ESC	Cancels text entries

Generation with HC-MES

The command is automatically generated when a text entry is requested.

Example

This example uses the **Insert Text** function to demonstrate different arguments for the STRING command.

```
REM           COMPONENTS
START    2
$A:=variable
%A :=5.25
REM           COMPONENTS            S6= Text
OPTION  16   2
REM            TEXT                 1 = Insert text
OPTION   1  25
POINT   #
REM     Text angle:
WINKEL  RET
REM     Text:
STRING  Constant
POINT   #
REM     Text angle:
WINKEL  RET
REM     Text:
STRING  $A
POINT   #
REM     Text:
STRING  RET
POINT   ESC
END
```

3.1.28 SZAUS / SZEIN

Function

Switches status bar updates off or on

Syntax

SZAUS

SZEIN

Argument

None

Generation with HC-MES

Automatic generation is impossible.

Note

On screen information in the status line is also updated during macro execution, e.g. when the active component is changed. These updates may be irritating during a macro run. SZAUS can be used to temporarily suppress them.

3.1.29 UDA / UDE

- **Function**

  Enables / Disables 3D Undo saves.

- **Syntax**

  UDE

  UDA

- **Argument**

  None

- **Note**

  An internal save is always executed for the function from which HiCAD's Undo function is invoked. This function can be temporarily suppressed for the duration of a macro run by the UDA function. It can be recalled by the UDE function.

- **Generation with HC-MES**

  Automatic generation is impossible.

3.1.30 VAI

Function

The entry of a user value for an integer variable during macro execution

Syntax

VAI % *variable* [*user prompt*]

Argument

% *variable*
A value is assigned to a integer variable. Although a whole number is normally entered, a real variable or an arithmetic expression may also be used. However in the case of real variables and arithmetic expression, returns are rounded to the next whole number.

user prompts
A *user prompt* is a sequence of characters with a maximum length of 50 characters. Although optional, this entry is useful as it signals the user when an entry is expected.

Generation with HC-MES

Automatic generation is impossible.

Example

In this example an optional number of circles with a specific radius are created. The number of circles and the radius is queries by the macro.

```
REM          LINES
START    6
VAI %a No. of circles
VAR %r Radius of circles
FOR  %i:= 1 TO a
REM          LINES          5 = Circle/Arc
OPTION   5   6
REM           Arc           5 = CP-Radius
OPTION   5  11
POINT   #
REM     Radius (INT for point on circle) :
DISTANZ r
POINT   ESC
NEXT I
END
```

3.1.31 VAR

Function

The entry of a user value for a variable during macro execution

Syntax

VAR % / $ *variable* / L*n* / P*n* /[*user prompt*]

Argument

% *variable*
Assigns a value to a real variable. Although a whole number is normally entered, a real variable or an arithmetic expression may also be used.

$*variable*
ASSIGNS a text string to a string variable. The length of the text string depends on the application (cf. STRING command).

Pn, Ln
Assigns point and line variables by identification.

user prompt
A *user prompt* is a sequence of characters with a maximum length of 50 characters. This entry is optional, but useful as it signals the user when an entry is expected.

Note

In comparison to the special function **Variable** (keys: STRG+6) the VAR command considerably simplifies the assignment of values. Enabling individual variables to be queried, it pauses macro execution to allow user inputs. If a variable holding a value has already been defined, it is offered for acceptance. Select RETURN to take the value over.

Generation with HC-MES

Automatic generation is impossible.

Please read the note about bracketed ECHO and VAR commands (cf. 3.1.7).

Example 1

```
REM           LINES
START    6
VAR %a No. of circles
VAR %r Radius of circles
FOR  %i:= 1 TO a
REM           LINES          5 = Circle/Arc
OPTION   5   6
REM            ARC           5 = CP-Radius
OPTION   5  11
POINT   #
REM     Radius (INT for point on circle) :
DISTANZ r
POINT   ESC
NEXT I
END
```

Example 2

```
REM           Point and line variables
START    6
VAR P0 Enter 1st point
VAR P1 Enter 2nd point
VAR L0 Identifiy line
REM           LINES          5 = Circle/Arc
OPTION   5   6
REM            ARC           7 = P-P-GE
OPTION   7  11
POINT   P0
POINT   P1
POINT   L0
REM     Circle  O.K.  (Y/N) ?
ANTWORT 1
POINT   ESC
END
```

3.1.32 Variable Assignment

Function

Value assignment to variables within a macro

Syntax

%*variable*:= *expression*	Numeric variable
$*variable*:= *expression*	String variable

Argument

expression
expression can mean a constant, a variable or an arithmetic expression, it depends on the type of variable to which the value is assigned. In this case you need to differentiate between numeric variables (%) and string variables ($)

Numeric variables
Value assignment to a numeric variable is triggered by a % character. Integer or real constants, numeric variables and arithmetic expressions are accepted for an *expression*. String variables are valid, even if they only contain a real number.

String variables
Value assignments to string variables are triggered by the $ character. To begin with, *expression* may be a character string, it may not however be enclosed in inverted commas (" "), as they would then be taken over with the variable.
Apart from single string variables, two string variables, (only two will be accepted), may be linked as follows $a+$b. Please make sure that a space is not inserted in front or behind the plus sign. A character string is accepted as the first (and only as the first!) string variable, but it must be set in inverted commas.
Finally, a numeric variable can be specified for *expression*. The numeric variable is identified by the % character and automatically converted to a string. As in the STRING command, it is evaluated as the format variable $@.

Note

This type of value assignment enables you to manipulate a macro without calling the special function **Variable** (key combination STRG+6). In this case values may be assigned to all user variables. Although system variables can be changed, it is not advisable!

Generation with HC-MES

Automatic generation is impossible.

Example 1

```
REM          LINES
START    6
$@:=(29hSurface of closed contour:, f10,2, 5h mm^^2)
$a := %z0
OPTION  ESC
REM          COMPONENTS            S6= Text
OPTION  16   2
REM          TEXT                  1 = Insert text
OPTION   1  25
POINT   #
REM     Text angle:
WINKEL  RET
REM     Text:
STRING  $a
POINT   ESC
OPTION  ESC
END
```

```
REM          LINES
START    6
VAR %a No. of circles
%i:=0
WHILE i < a
REM          LINES          5 = Circle/Arc
OPTION   5   6
REM          ARC            5 = CP-Radius
OPTION   5  11
POINT   #
REM     Radius (INT for point on circle) :
DISTANZ #
POINT   ESC
%i:=i+1
WHEND
END
```

3.1.33 WAIT

Function

Displays a user prompt in the text box and pauses macro execution. RETURN resumes execution.

Syntax

WAIT [*user prompt*]

Argument

user prompt
This argument is similar to the ECHO command. In this case, as it may occupy an entire text box line, a *prompt* can have a maximum length of 66 characters. If, in spite of this, a prompt exceeds the reserved length, the screen can be re-formatted with the EGA reset (special **ZOOM** function).

You can also output prompts contained in HiCAD's MAKROTXT.DAT file. To do this, you need to copy the file to the macro directory. Enter the string **$100*n*** to access a line in this file, whereby ***n*** is the number of the corresponding line in the MAKROTXT.DAT file, e.g. **WAIT $10021.**

In this case, the prompt contained in line 21.

Note

The WAIT command is similar to the ECHO command and is also used to display auxiliary information during a macro run. In contrast to the ECHO command, after the output of a prompt, the WAIT command temporarily interrupts the run. This is indicated by an asterisk (*) appended to the prompt. Macro execution is continued with RETURN. No other actions are possible at this point.

Generation with HC-MES

Automatic generation is impossible.

Example

```
REM           DRAWINGS
START    1
WAIT Draw new line
REM           DRAWINGS          3 = Process ZTL
OPTION   3   1
REM           COMPONENTS        1 = LINES
OPTION   1   2
REM           LINES             2 = Polyline
OPTION   2   6
POINT   #
POINT   #
POINT   ESC
END
```

3.1.34 WARTE

Function

Causes macro execution to pause (measured in seconds).

Syntax

WARTE *n*

Argument

n

This value specifies the number of seconds by which the macro run should be paused.

Generation with HC-MES

Automatic generation is impossible.

Example

```
REM           LINES
START    6
WARTE 20
REM           LINES                9 = Preset Par.
OPTION   9   6
REM           LINE PARAM.          6 = Default param.
OPTION   6  18
OPTION ESC
END
```

3.1.35 WAUS/WEIN

Function

Enables / Disables the Wait status

Syntax

WAUS
WEIN

Argument

None

Note

In contrast to SAUS/SEIN, by default the "disable" condition (WAUS) is set. This means that macro execution is not paused after purely informative prompts (to which RETURN is the only response possible) are output to the text box. This type of output is indicated by an asterisk (*) appended to the prompt, e.g. with the function **Set Default Parameter** for diverse functions. If a procedure should be temporarily interrupted, the macro must contain a previous WEIN.

Generation with HC-MES

Automatic generation is impossible.

Example

```
REM           LINES
START    6
WEIN
REM           LINES                9 = Preset Par.
OPTION   9   6
REM           LINE PARAM.          6 = Default param.
OPTION   6  18
OPTION ESC
END
```

The output of error messages for unsuccessful database searches can be influenced by **WEIN/WAUS** commands.

3.1.36 WERT

Function

Checks whether a numeric or text variable is available

Syntax

WERT %*variable*

WERT $*variable*

Argument

%*variable*

$*variable*

Checks whether the variable is defined. If it is, the logical variable VORHD id set to TRUE, otherwise FALSE is set.

3.1.37 WHILE...WHEND

Function

A condition controlled loop is executed continuously and only finished when a particular condition is meet. It always checks the condition at the start of the loop.

Syntax

WHILE *logical expression*
.
. HCGS command
.
WHEND

Argument

logical expression
WHILE loops check whether the result returned by the given expression is true. If the result is false, execution is continued with the HCGS command following WHEND. Please remember that all *logical expressions* must be defined in advance.

Note

WHILE loops can be used where the number of repetitions is unknown. As a WHILE loop tests for its exit before any instructions are executed, it allows for unforeseeable situations, e.g. user inputs during a macro run. WHILE loops can, therefore, be less specific than FOR loops.

Generation with HC-MES

Automatic generation is impossible.

Example

The macro below deletes all named DIN 931 - M 42 X 130 - ST hexagon screws from the current drawing. The system variable VORHD is thereby used in place of a logical expression. This causes the loop to end immediately when a component with the name DIN 931 - M 42 X 130 - ST cannot be found.

```
REM          HiCAD-Next
START   59
REM   HiCAD       3 = 3-D SCENE
OPTION   3  59
REM   3-D SCENE   3 = Process
OPTION   3 101
REM   PARTS      10= Activate
OPTION  10 102
STRING  DIN 931 - M 42 X 130 - ST
WHILE vorhd
REM   PARTS       7 = Delete
OPTION   7 102
REM     O.K. ?
ANTWORT 1
REM   PARTS      10= Activate
OPTION  10 102
STRING  DIN 931 - M 42 X 130 – ST
WHEND
ECHO All DIN 931 Screws deleted or not existing!!
END
```

3.1.38 WINKEL

Function

An angle specification relative to a reference line

Syntax

WINKEL *real constant* / *arithmetic expression* / # / ZEI / ESC

Argument

RET The displayed default value (fixed value) is taken over during macro execution.

\# The value is queried (free argument) during macro execution.

ZEI This argument is automatically entered if **z** is specified during macro creation, i.e. the angle is derived from the drawing. The next macro command is then:

OPTION *number* 24

This is equivalent to the activation of the **Angle** function in the **Information** menu. From now on, the command sequence depends on *number*. In a macro, ZEI simulates the actions taken when you specify "z" at the appropriate time during a normal HiCAD session.

Note

This command works in the same way as the DISTANZ command and is required when a specification is required for an angle, (point options P, PX, PY, WX, WY)

Generation by HC-MES

The command is automatically generated whenever an angle specification is required.

Example

```
REM           LINES
START    6
REM           LINES                2 = Polyline
OPTION   2   6
POINT   #
POINT   P Z
WINKEL  ZEI
REM           ANGLE                3 = Angle GE-axis
OPTION   3  24
POINT   #
REM     Plus right angle (4) - Minus right angle (5)
INTEGER RET
REM     Offset :
DISTANZ 50
POINT   ESC
END
```

3.1.39 ZAA / ZAE

Function

Completely redraws the screen / Cancels "redraw"

Syntax

ZAA

ZAE

Note

The ZAE cancels screen formatting, e.g. for the **View All** option. In the case of large drawings, this enables the speed of execution to be increased. ZAA enables formatting.

Generation with HC-MES

Automatic generation is impossible.

Example 1

This macro disables screen formatting for the **Redraw All Parts** function, but enables the screen to be completely re-drawn for the **View All** function.

```
REM          COMPONENTS
START    2
REM ALT1
ZAE
OPTION  21   0
REM   REDRAW   1 = All parts
OPTION   1  51
REM ALT2
ZAA
OPTION  22   0
REM   ZOOM     5 = View all
OPTION   5  52
END
```

Example 2

This macro disables screen formatting for the **Re-draw All Parts** function, and only allows partial reformatting for the **View All** function.

```
REM          COMPONENTS
START    2
REM ALT1
ZAE
OPTION  21   0
REM   REDRAW     1 = all parts
OPTION   1  51
REM ALT2
OPTION  22   0
REM   ZOOM       5 = View all
OPTION   5  52
END
```

3.2 Example

This section illustrates how HiCAD macro techniques can be implemented. In our sample session, a new main part containing the 2D parts of the current part is created, and subsequently moved.

You need to:

- Open a drawing file
- Delete variable memory

Now take the following actions:

- Invoke the **Create Macro** function (key combination STRG+7)
- Specify a name for the macro e.g. *PARTCOP*
- Call the special function **Redraw**
- ALT-functions and select **All Parts**
- ALT-functions and select the special function, **ZOOM**
- Select the **View All** option
- Activate the **2D ZTL** menu
- Activate the **Process ZTL** menu
- Select the **Create Main Part** option
- Select the **Create Part** option
- Enter a name for the component, e.g. COPY1
- Switch to the **Line** menu
- Select the **Copy GE** option
- Specify a rectangular pick-box to determine the line elements
- Invoke a point option to specify the lower left corner of the box
- Invoke a point option to specify the upper right corner of the box
- **2xEND** to call the component menu
- Select the **Transform** option
- Select the **Shift** option
- Specify a *fitting point* on the part
- Specify a *fitting point* on the drawing
- Select the **End MACRO** function to exit macro creation

These actions generate the following macro:

```
REM   HICAD-Next
REM          HiCAD-Next
START   59
REM ALT1
OPTION  21   0
REM   REDRAW   1 = All parts
OPTION   1  51
REM ALT2
OPTION  22   0
REM   ZOOM   5 = View all
OPTION   5  52
REM   HiCAD   2 = 2-D ZTL
OPTION   2  59
REM   DRAWINGS   3 = Process ZTL
OPTION   3   1
REM   COMPONENTS   8 = New main part
OPTION   8   2
REM   NEW PART   1 = Create part
OPTION   1   3
STRING  COPY1
REM   COMPONENTS   1 = LINES
OPTION   1   2
REM   LINES   S5= Copy GE
OPTION  15   6
REM      in rectangle (4) – inside contour  (5)  -  GE seg-
ment (6) :
INTEGER 4
POINT   #
POINT   #
POINT   ESC
OPTION  ESC
REM   COMPONENTS   2 = transform.
OPTION   2   2
REM   TRANSFORM.   1 = Shift
OPTION   1  27
POINT   #
POINT   #
END
```

We now modify the macro:

- During macro execution, the selection box for the **Copy GE** should be automatically determined and depend on the coordinates of the complete view (View All). These values are assigned to system variables:
  - ZW, ZX for the lower left corner of the complete view, and
  - ZY, ZZ for the upper right corner of the complete view.

  The coordinates of the selection box should then be calculated as follows

  u1 = x-coordinate of the lower, left corner = ZW-1
  u2 = y-coordinate of the lower, left corner = ZX-1
  o1 = x-coordinate of the upper, right corner = ZY+1
  o2 = y-coordinate of the upper, right corner = ZZ+1

  The corresponding macro instructions are:

```
%u1:=ZW -1
%u2:=ZX -1
%o1:=ZY +1
%o2:=ZZ +1
```

  These lines are inserted in the macro in front of the new main part command. Variables u1, u2, o1 and o2 are used to specify the selection box with **POINT** commands and invoke the **ABSOLUTE X,Y** point function.

```
REM   HICAD-NT  : 2.0 VN:07
REM          HiCAD-Next
START   59
REM ALT1
OPTION  21   0
REM   REDRAW   1 = All parts
OPTION   1  51
REM ALT2
OPTION  22   0
REM   ZOOM   5 = View all
OPTION   5  52
REM   HiCAD   2 = 2-D ZTL
OPTION   2  59
REM   DRAWINGS   3 = Process ZTL
OPTION   3   1
%u1:=ZW -1
%u2:=ZX -1
%o1:=ZY +1
%o2:=ZZ +1
REM   COMPONENTS   8 = New main part
OPTION   8   2
REM   NEW PART   1 = Create part
OPTION   1   3
STRING  COPY1
REM   COMPONENTS   1 = Lines
OPTION   1   2
REM   LINES   S5= Copy GE
OPTION  15   6
REM      in rectangle (4) – inside contour  (5)  -  GE seg-
ment (6) :
INTEGER 4
POINT   A u1 u2
POINT   A o1 o2
POINT   ESC
OPTION  ESC
REM   COMPONENTS   2 = transform.
OPTION   2   2
REM   TRANSFORM.   1 = Shift
OPTION   1  27
POINT   #
POINT   #
END
```

- The name of the new component should be variable and queried after the macro start. A **VAR instruction** is therefore set at the start of the:

  VAR $n Name of copy:

  The variable is inserted in the STRING instruction to replace the name "COPY1".

```
REM   HICAD-NT  : 2.0 VN:07
REM          HiCAD-Next
START   59
VAR $N Name of copy:
REM ALT1
OPTION  21   0
REM   REDRAW   1 = All parts
OPTION   1  51
REM ALT2
OPTION  22   0
REM   ZOOM   5 = View all
OPTION   5  52
REM   HiCAD   2 = 2-D ZTL
OPTION   2  59
REM   DRAWINGS   3 = Process ZTL
OPTION 3
%u1:=ZW -1
%u2:=ZX -1
%o1:=ZY +1
%o2:=ZZ +1
REM   COMPONENTS   8 = New main part
OPTION   8   2
REM   NEW PART   1 = Create part
OPTION   1   3
STRING  $N
REM   COMPONENTS   1 = Lines
OPTION   1   2
REM   LINES   S5= Copy GE
OPTION  15   6
REM      in rectangle (4) – inside contour  (5)  -  GE seg-
ment (6) :
INTEGER 4
POINT   A u1 u2
POINT   A o1 o2
POINT   ESC
OPTION  ESC
REM   COMPONENTS   2 = transform.
OPTION   2   2
REM   TRANSFORM.   1 = Shift
OPTION   1  27
POINT   #
POINT   #
END
```

4 Additional Notes

4.1 DXF Files

When DXF files are created by a user macro, the presence of the DXF file cannot be controlled by the variable, VORHD. In this case the variables JA or NEIN should be used, e.g. IF JA THEN.....

Normally DXF layers are allocated to HiCAD layers. To make identification easier in HiCAD 16 and higher, DXF layer names are also taken over and displayed as line parameter. Currently, this only applies when the id 'LAYNA 1' is set for the DXF/DWG take over of ACADHCAD.DAT files.

4.2 Macro variable: ZDSP

In text menu mode, the function sequence:

2D - Process - Interface - cur.FIG -

can be used to activate a component, as long as the macro variable ZDSP holds its index. ZDSP can, for instance, be set with the **Dimension Info** function.

4.3 Select Text

When working in HiCAD Text Menu mode during macro creation or processing the **Select Text** function is available. As the number of text positions saved has been increased for HiCAD 18 and higher, it is reasonable to increase the tolerance set for text height within a text. A tolerance value can be set in the sys\TXTPAR.DAT system file.

This setting is advantageous when reading DXF files.

4.4 Text Tools

The **Increment Text No.** function in **Text/Text-Tools** menu of the macro recorder enables numeric text having a uniform text code to be increased by a given value (e.g. for 2D Item No.).

4.5 True Type Font

If the appropriate font is specified in the TTFONT.DAT system file, it is possible to set True Type Fonts as text parameter for macro runs. If you select Fonts plus INT when you set parameters, TT fonts can be accessed according to the sequence in which they are saved to the TTFONT.DAT system file.

4.6 Objektcursor

If you set the expression %@idt:=0 in front of an info function, the object cursor is switched off for this function during a macro run without user input, this can considerably speed up run time.

4.7 Develop Sheet Metal

The result of the system variable $@AWV selected method for sheet metal development, e.g. F = Faktorenverfahren (DIN). The result ist identical to the ID in the file method.DAT. method is name of the corresponding DIN, for example DIN 6935.DAT (MAKROABW directory)

4.8 Dimensioning

Use the menu entry **BEM.TRANS** in the transformation menu to switch on/off associative dimensioning. This may be usefull if a dimension foot point belongs to different components and you dont't want the corresponding dimension to be shifted.

4.9 Sketching Cursor

The numeric variable @SKZ enables you to specify whether the sketching cursor should be available with functions that are not associated with sketch technique (e.g. rectangle functions, transformation etc.). If this variable is set to 0, the cursor is switched off; if the variable has another value or is not assigned, the sketching cursor is active. You can, for instance, set the variable in the INITAL.MAC start macro.

4.10 Component ID

When you create or activate a 3D part, the system variable **%ZKEN** is occupied by a unique component id. You can then identify the 3D part in component selection by entering the name X__IDKENN (2 underscores) and, in response to the subsequent query, specifying the known id

4.11 Call Operating System

Up to now, when Operating Systems were called the character **/** was converted to **\,** this conversion no longer takes place.

4.12 Material Hatching

To define a material hatching enter 9999 for the first hatching code. HiCAD then asks for further hatching parameters.

4.13 Export Data to STEP/MTA or IGES/CATIA

When using macros to export HiCAD data to STEP/MTA or IGES/CATIA files, please define the macro variable **STYP** at the beginning of the macro.

STEP %STYP:= 1
MTA %STYP:= 2
IGES %STYP:= 1
CATIA %STYP:= 2

4.14 Temporary path

The temporary FILEGRUP.DAT path |: can now be used in macros. This logical path will not be entered in the FILEGRUP.DAT, but can be used to allocate any Windows paths to a HICAD path.

This path does not need to be saved/backed up, but has to be assigned in the macro.

In order to allocate this path, select the MANAGEMENT → Change Path function in the text menu mode during the macro recording. Then, enter **|** as file group ((| on the <,> key) and the required Windows path as a path, e.g. D:\USERDATEN.

```
REM   HICAD-Next VN:1210
REM          HiCAD
START   59
HNEXT
REM   HiCAD   1 = Verwaltung 1
OPTION   1  59
REM   VERWALTUNG   4 = Pfad ändern
OPTION   4   8
REM     Dateigruppe (neuer Default-Pfad) :
STRING  |
REM      Dateiverzeichnis:
STRING  RET
END
```

4.15 Macro Variables for UNDO

New macro variables are available for the Undo functionality.

ZUNG **Global Undo**

0 The UNDO-functionality is switched off, i.e. there is no option to regain a previous state.

When UNDO is switched off, the variables **ZUNA** (UNDO active/inactive) and **ZUNP** (UNDO-break) are irrelevant

1 UNDO-functionality is switched on

ZUNA **UNDO active/inactive**

0 UNDO inactive
Inactive means that no UNDO-backup is made. If UNDO is inactive, the **ZUNP** variable (UNDO break) is irrelevant

1 UNDO active

ZUNP **UNDO-break (UNDO is switched off temporarily)**

0 UNDO does not pause

1 UNDO pauses
The effect of the pause is that there is no UNDO backup. The state before the pause can be restored any time, as a Scene UNDO-backup is carried out before the pause. When the pause is cancelled, UNDO-backups will be made again.

The pause will be activated/deactivated with the **UDA/UDE** macro order.

Please remember:

- Please be careful to use the UDA/UDE macro order, as it needs to be ensured that only 2temporary" parts must be processed/created between UDA and UDE, which have the regain the pre-UDA state after UDE !!
- An UNDO-backup can be carried out if ZUNG = 1, ZUNA = 1 and ZUNP = 0.
- The UNDO-/REDO-backup can be loaded via the menu bar if the appropriate icon is not grey.

5 Error Messages

The following error messages may be issued during macro creation or execution:

Error Code	Cause
-2	Empty HCGS procedure cannot be corrected
-1	HCGS procedure is not available
1	HC ZED expects an OPTION command not the existing command
3	CALL or MAKRO command: Unable to find file
10	Invalid menu level
20	Error in the argument of an ZUSATZ statement
21	HCGS procedure does not include a ZUSATZ statement
30	Unknown data record (file name) in the DATFIL command
40	Error in the argument of a LINE or WIED instruction
41	HCGS procedure does not include a LINE or WIED instruction
99	Unknown HCGS command
100	Illegal argument in the START command
101	HCGS procedure without START command or HCGS command precedes the START command
102	No parameter for START command
103	Faulty argument in END command
110	No argument for HCGS command
111	Faulty argument for OPTION command
201	VAR Px or VAR Lx command in a point entry
202	Unknown variable in VAR command
205	Unknown variable in variable assignment
206	Error in arithmetic expression for variable assignment
999	A compiled macro procedure cannot be revised

Error Code	Cause
1001	Empty macro
1002	Not enough space for HCGS procedure in the given memory area
1003	HCGS procedure too large for revision
1004	More than 99 jump destinations (labels)
1005	Invalid jump destination (label exceeds 9999)
1006	No jump destination (label)
1007	Too many FOR, WHILE or REPEAT loops
1008	False or no variable name for NEXT
1009	Incorrect nesting of FOR loops
1010	NEXT without FOR or unknown loop beginning
1011	UNTIL without REPEAT
1012	WHEND without WHILE
1019	Loop end without start command
1020	Nesting too deep for CALL
1021	Incorrect FOR instruction
1022	Illegal variable assignment
1023	REPEAT loop has no exit
1024	WHILE loop has no end
1025	FOR instruction has no end
1026	IF instruction has no end
1027	Hierarchical error
1028	Invalid macro command
1029	Invalid variable test
1030	HiCAD show/hide functions only allowed immediately in front of 'OPTION'

6 Macro Libraries

HiCAD offers you the possibility of creating macro libraries that can be subsequently integrated in the screen menu. This enables you to call a macro belonging to a library quickly and simply from a pop-up menu.

The functions used to create and process macro libraries can be found in the **Drawing** Tab. Select **User Library** in the **Insert Part** function group.

For a detailed description please refer to the HiCAD Online help (HiCAD Basics > Customising HiCAD).

7 The FILEGRUP.DAT File

The **FILEGRUP.DAT** file is required by several functions, e.g. to create preview images of variants in macros. FILEGRUP.DAT contains all default settings for internally relevant HiCAD directories. When HiCAD is installed the file is saved to the program directory, i.e. in the *exe* subdirectory, and read when HiCAD is started.

```
   C:/HICAD/sys
A:C:/HICAD/ztl
B:C:/HICAD/norm
C:C:/HICAD/ztl
D:C:/HICAD/ega
E:C:/HICAD/makrolib
F:C:/HICAD/ztl
G:C:/HICAD/ztl
H:C:/HICAD/ztl
I:C:/HICAD/ztl
J:C:/HICAD/ray
K:C:/HICAD/material
L:C:/HICAD/ztl
.
.
```

Fig. 9 Extract from he FILEGRUP.DAT file

There is always one valid path specification in each line of the file. The first two characters in the line are the names of the file group, i.e. the short designation of the path in HiCAD, e.g. *A*:, followed by the complete path specification comprising a maximum of 80 characters.

The first line **must** contain the path designation of the HiCAD system directory. In this case, the 2-character description is replaced by blanks. Each following line contains the path for individual HiCAD file directories.

Directories entered in the FILEGRUP.DAT file can be changed by calling the **Settings** menu and selecting **Directories**. (cf. *HiCAD Tutorial*)

You can exchange any directory indicated by an open lock. Click the appropriate filed and select the required directory.

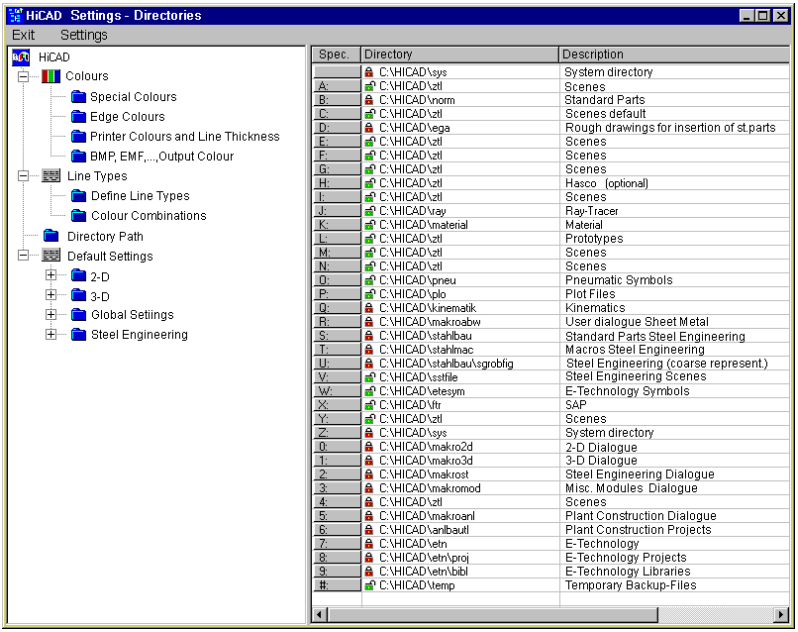


Fig. 10 Change directories

Directory assignments in the **Settings** menu are retained when you exit HiCAD, i.e. when you start HiCAD the next time, they are the default settings These settings are saved in the FILEGRUP.DAT file.

The first time you select the **Directory** option, the original HiCAD directory assignments are saved to the FILEGRUP.ORI file. These assignments can be recalled by selecting **Load Default Settings** from the **Settings** menu.

Please remember:

You are enabled to load/save your scene from/in any directory, independent of the FILEGRUP.DAT file settings, as common for all Windows programs. The default path for HiCAD scenes is used as a default setting when starting HiCAD again. The path you used last will be shown when loading files during a session.

Index

A

Amend Macro 16
ANTWORT 47
APAUS 49
APEIN 49
Argument Groups 23
Arithmetic Operators 33

B

Basic Functions 34

C

Call 50
Call Macro 13
Cancel 18
CLOSE 74
Constants 23
Control Arguments 25
COPY 52
Create Macro 10

D

DEL 54
Delete 18
DISTANZ 55

E

ECHO 57
End 18
END 59
End Macro Creation 13
Error Messages 121
Example Macro 110

F

FAA 60
FAE 60
File Names 23
FILEGRUP.DAT 124
Fixed Arguments 24
FOR 61
Free Argument 12

G

GOTO 63, 65

H

HCGS Commands 21

I

IF 65
IFEND 65
IGNORE 67
INPUT 74
INTEGER 68

L

Label 18

M

Macro
  Compile 20
Macro Commands 11
Macro Creation Mode ... 10
Macro Frames 11
Macro Language 21
Macro Name 11
Macro Structure 11
Macro Variables for
  UNDO 119
MAKRO 70
Mathematical Functions 34
MAUS 72
MEIN 72
MKDIR 73

N

NEXT 61
Numbers 23

O

OPEN 74
OPTION 77

Optional Arguments 23
OUTPUT 74
Overwrite Macro.....14

P

Path
Temporary.....118
Path Specifications 124
PFD 80
POINT 81
Point Option 25
Process Macro 14

R

REAL.....83
REM 84
REPEAT.....86
Run.....17

S

SAUS 87
Save 18
SEIN.....87
Single-Step Mode 17
START 89
STRING.....90
String Expressions.....39
String Functions.....34
String Operations.....40
System File
FILEGRUP.DAT.....124
SZAUS 92
SZEIN.....92

T

Take Over Macro ... 15, 18
Temporary Path 118
Test Macro 18
THEN..... 65

U

UDA..... 93
UDE..... 93

V

VAI..... 94
VAR..... 95
Variable
User..... 26
Variable Assignments .. 97
Variables 19, 24

W

WAIT 99
WARTE 101
WAUS..... 102
WEIN..... 102
WERT..... 103
WHILE..... 104
WINKEL 106

Z

ZAA 108
ZAE 108
Zoom Options..... 17

Issued and edited by: © 2009 ISD ® Software und Systeme GmbH

ISD Software und Systeme GmbH
Hauert 4
44227 Dortmund
Germany
Tel. +49-(0)231-9793-0
Fax +49-(0)231-9793-101
info@isdgroup.de
www.isdgroup.de

ISD Berlin
Paradiesstraße 208a
12526 Berlin
Germany
Tel. +49-(0)30-634178-0
Fax +49-(0)30-634178-10
berlin@isdgroup.de
www.isdgroup.de

ISD Hannover
Ahrensburger Straße 3
30659 Hanover
Germany
Tel. +49-(0)511-616803-40
Fax +49-(0)511-616803-41
hannover@isdgroup.de
www.isdgroup.de

ISD Nürnberg
Nordostpark 7
90411 Nuremberg
Germany
Tel. +49-(0)911-95173-0
Fax +49-(0)911-95173-10
nuernberg@isdgroup.de
www.isdgroup.de

ISD Ulm
Wilhelmstraße 25
89073 Ulm
Germany
Tel. +49-(0)731-96855-0
Fax +49-(0)731-96855-10
ulm@isdgroup.de
www.isdgroup.de

ISD Austria GmbH
Hafenstraße 47-51
4020 Linz
Austria
Tel. +43-(0)732-9015-1800
Fax +43-(0)732-9015-1829
info@isdgroup.at
www.isdgroup.at

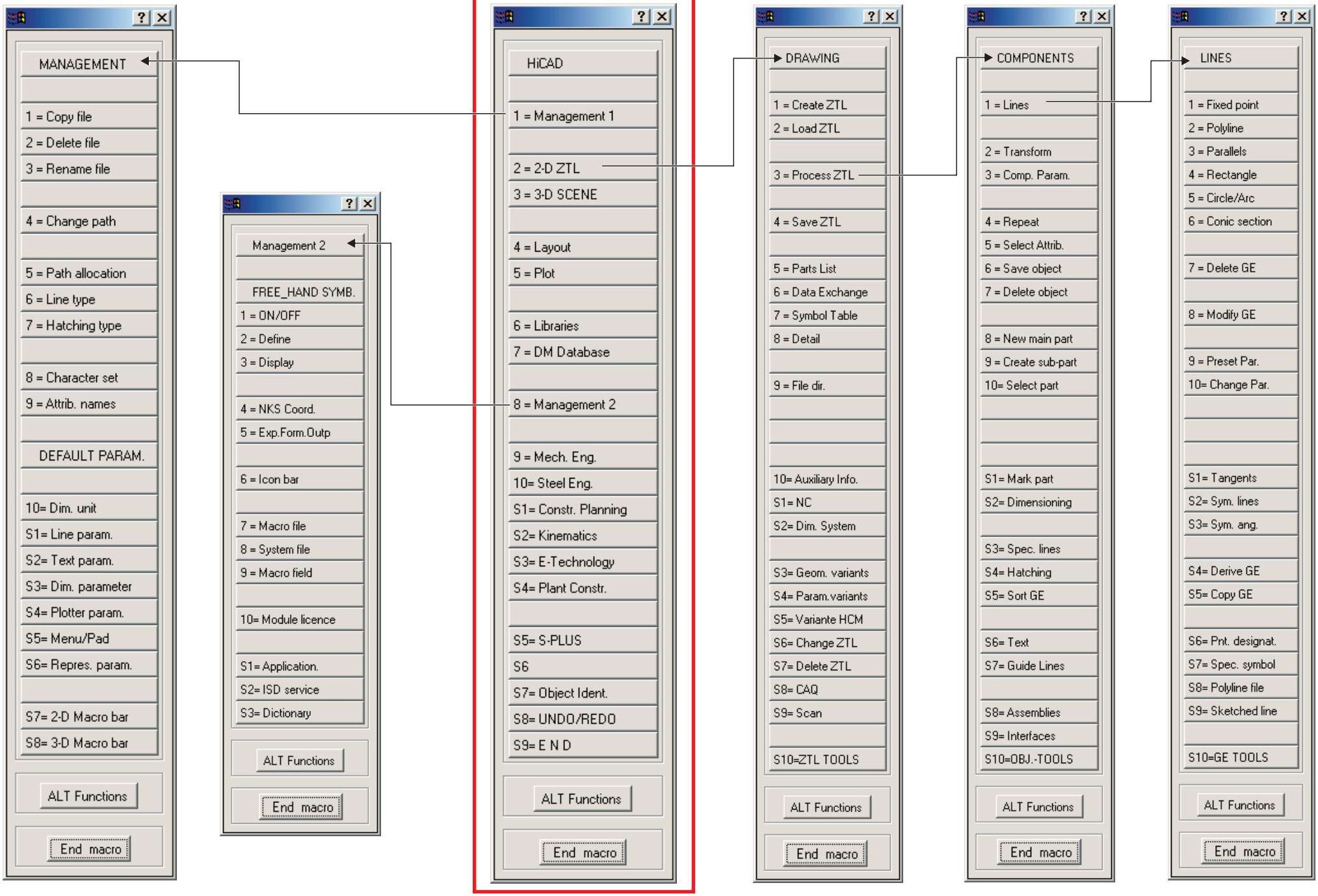
ISD Benelux b.v.
Het Zuiderkruis 33
5215 MV 's-Hertogenbosch
The Netherlands
Tel. +31-(0)73-61538-88
Fax +31-(0)73-61538-99
info@isdgroup.nl
www.isdgroup.nl

ISD East Europe Sp.z.o.o.
Ul. Fortuny 6B
01-339 Warszawa
Poland
Tel. +48-(0)22-86205-29
Fax +48-(0)22-86205-30
info@isdgroup.pl
www.isdgroup.pl

ISD Italia s.r.l.
Viale Zanotti, 76
27027 Gropello Cairoli (Pavia)
Italia
Tel. +39-(0)382-815772
Fax +39-(0)382-826112
info@isdgroup.it
www.isdgroup.it

ISD Japan Co. Ltd.
Daiwajisho-Building 4th floor
74-1 Yamashita-cho Naka-Ku,
Yokohama, Kanagawa, 231-0023
Japan
Tel. +81-(0)45-315-9605
Fax +81-(0)45-315-9607
info@isdgroup.jp
www.isdgroup.jp

ISD Schweiz AG
Rosenweg 2
4500 Solothurn
Switzerland
Tel. +41-(0)32-62413-40
Fax +41-(0)32-62413-42
info@isdgroup.ch
www.isdgroup.ch


MANAGEMENT
1 = Copy file
2 = Delete file
3 = Rename file
4 = Change path
5 = Path allocation
6 = Line type
7 = Hatching type
8 = Character set
9 = Attrib. names
DEFAULT PARAM.
10= Dim. unit
S1= Line param.
S2= Text param.
S3= Dim. parameter
S4= Plotter param.
S5= Menu/Pad
S6= Repres. param.
S7= 2-D Macro bar
S8= 3-D Macro bar
ALT Functions
End macro
Management 2
FREE_HAND SYMB.
1 = ON/OFF
2 = Define
3 = Display
4 = NKS Coord.
5 = Exp.Form.Outp
6 = Icon bar
7 = Macro file
8 = System file
9 = Macro field
10= Module licence
S1= Application.
S2= ISD service
S3= Dictionary
ALT Functions
End macro
HiCAD
1 = Management 1
2 = 2-D ZTL
3 = 3-D SCENE
4 = Layout
5 = Plot
6 = Libraries
7 = DM Database
8 = Management 2
9 = Mech. Eng.
10= Steel Eng.
S1= Constr. Planning
S2= Kinematics
S3= E-Technology
S4= Plant Constr.
S5= S-PLUS
S6
S7= Object Ident.
S8= UNDO/REDO
S9= E N D
ALT Functions
End macro
DRAWING
1 = Create ZTL
2 = Load ZTL
3 = Process ZTL
4 = Save ZTL
5 = Parts List
6 = Data Exchange
7 = Symbol Table
8 = Detail
9 = File dir.
10= Auxiliary Info.
S1= NC
S2= Dim. System
S3= Geom. variants
S4= Param.variants
S5= Variante HCM
S6= Change ZTL
S7= Delete ZTL
S8= CAQ
S9= Scan
S10=ZTL TOOLS
ALT Functions
End macro
COMPONENTS
1 = Lines
2 = Transform
3 = Comp. Param.
4 = Repeat
5 = Select Attrib.
6 = Save object
7 = Delete object
8 = New main part
9 = Create sub-part
10= Select part
S1= Mark part
S2= Dimensioning
S3= Spec. lines
S4= Hatching
S5= Sort GE
S6= Text
S7= Guide Lines
S8= Assemblies
S9= Interfaces
S10=OBJ.-TOOLS
ALT Functions
End macro
LINES
1 = Fixed point
2 = Polyline
3 = Parallels
4 = Rectangle
5 = Circle/Arc
6 = Conic section
7 = Delete GE
8 = Modify GE
9 = Preset Par.
10= Change Par.
S1= Tangents
S2= Sym. lines
S3= Sym. ang.
S4= Derive GE
S5= Copy GE
S6= Pnt. designat.
S7= Spec. symbol
S8= Polyline file
S9= Sketched line
S10=GE TOOLS
ALT Functions
End macro

HiCAD Macro technique