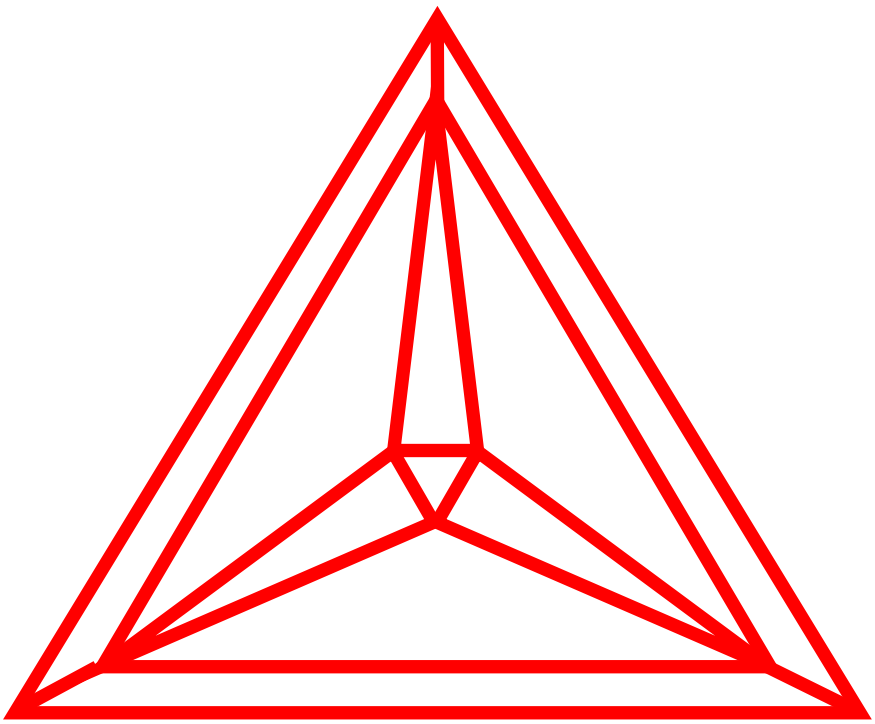
DATAPLOT

(Graphical Language)

For Thermo-Calc™ and DICTRA™ Software

User's Guide & Examples



Thermo-Calc Software AB
Norra Stationsgatan 93, 5 tr.
SE-113 64 Stockholm, Sweden

Copyright:

Disclaimers:

Acknowledgement of Copyright and Trademark Names:

Availability of This Document:

For the purpose of environment-friendlessness, this ***DATAPLOT User's Guide & Examples*** and all other operational manuals (User's Guides and Examples Books), as well as Reference Lists and other technical documentations, for the TCS-provided software, databases and programming interfaces are provided along the delivered TCS Standard Products CDs and installed on each of designated installation, which can be reviewed and accessed easily and conveniently. If desired and preferred, a user can locally print such a manual BUT it is only for the purpose of the user's internal use. For a hard copy of such a manual physically printed and delivered by TCS, a certain fee shall be applied.

To make manual updating more prompt and efficient, the later manual revisions or additions will be made available on the Internet. Our users may therefore download such revised documents from our company's web site www.thermocalc.com.

Editors of This Document:

Dr. Pingfang Shi
Thermo-Calc Software AB (TCS)
Norra Stationsgatan 93, 5 tr.
SE-113 64 Stockholm, SWEDEN
E-Mail: pingfang@thermocalc.se

Prof. Bo Sundman
Dept. of Materials Science & Engineering
Royal Institute of Technology (KTH)
SE-100 44 Stockholm, SWEDEN
E-Mail: bosse@mse.kth.se

Contents

1 Introduction

1.1 DATAPLOT Graphical Language

In order to obtain graphical output of any numerical data and informative strings, a graphical language called **DATAPLOT** has been developed in connection with the graphical software **DIGLIB** [1]. Using this graphical language, a user may store various information in a normal textual file (`*.EXP`), which can be plotted as graphical symbols, lines, texts or Greek letters on any plot device support by DIGLIB.

It is possible to create a DATAPLOT (`*.EXP`) file directly from diagrams published in journals by using a graphical tablet and the program **DIGPAD** [2].

A user may generate and plot DATAPLOT (`*.EXP`) files together with various calculation and/or experimental results from, for example, phase and property diagrams calculated by the **Thermo-Calc** software [3,4,5], analytical functions defined by the **FuncOptPlot** software [6], or composition profiles simulated by the **DICTRA** software [7,8].

The Thermo-Calc and DICTRA software have completely implemented the DATAPLOT language and relevant graphical interface in the POST-processor. Therefore, various calculation and simulation results from these software are always interpreted in the comprehensive DATAPLOT language that ensures efficient graphical presentations on screen, graphical files or hard copies in a professional and high-quality graphical standard.

An `EXP` file automatically generated by the POST-processor (using the `MAKE_EXPERIMENTAL_DATAFILE` command) in the Thermo-Calc and DICTRA software is a DATAPLOT (`*.EXP`) file and contains all types of legal DATAPLOT commands and their parameters. With a simple textual editor, the user may also modify or add some DATAPLOT commands and related parameters in an existing `EXP` file. This is very useful when appending experimental information on calculated/simulated plots, and when specifying user-desired texts, symbols, colors, fonts, filled patterns, diagram types, diagram sizes, symbol/character sizes, titles, special characters, *etc*.

However, if a PostScript copy is generated (either saved as a PostScript graphical file or printed on a PostScript-supporting device), some normal DATAPLOT commands may appear in a strange way or do not work properly. Therefore, some special formatting codes must be used for such PostScript outputs; this is achieved by using the **LTEXT Text Formatting Program** and/or the ordinary PostScript commands.

1.2 Related References

1. *DIGLIB User's Guide*
2. *DIGPAD User's Guide*
3. *Thermo-Calc Software System*
4. *Thermo-Calc User's Guide*
5. *FOP User's Guide*
6. *DICTRA User's Guide*
7. *DICTRA Examples Book*

1.3 About This Document

This ***DATAPLOT User's Guide & Examples*** document is a supplementary part of the following manual sets:

- ***Thermo-Calc 3.0 Console Mode Manual Set User's Guide***
- ***DICTRA26 Manual Set (DICTRA26 User's Guide*** and ***DICTRA26 Examples Book***).

The ***DATAPLOT User's Guide*** gives all the details on the graphical language **DATAPLOT** that a phase diagram or property diagram calculated by Thermo-Calc or a kinetic profile simulated by DICTRA26 can be defined.

In this User's Guide, the important features of the DATAPLOT language are overviewed in *Chapter 2*. Then, all commands are described in detail, for defining `PROLOGUE` (in *Chapter 3*) and `DATASET` (in *Chapter 4*), respectively. The LTEXT Text Formatting Program for editing PostScript codes is summarized in *Chapter 5*. Finally, a number of instructive examples and the standard codes for various formatting purposes are given in *Chapter 6*.

A section (*i.e., Section 2.13*) introduce the method of formatting DIGLIB symbols in LaTeX documents, for the purpose of necessarily/appropriately referring to the corresponding LaTeX symbols (closest to those DIGLIB symbols which have been plotted on a Thermo-Calc or DICTRA figure using the DATAPLOT Graphical Language) in the texts of LaTeX documents for publications/reports.

The section *Example 6 in Section 6.10* demonstrate how to use different available color codes for setting colors in plotted Thermo-Calc and DICTRA graphs.

Editors of This Document:

Dr. Pingfang Shi
Thermo-Calc Software AB (TCS)
Norra Stationsgatan 93, 5 tr.
SE-113 64 Stockholm, SWEDEN
E-Mail: pingfang@thermocalc.se

Prof. Bo Sundman
Dept. of Materials Science & Engineering
Royal Institute of Technology (KTH)
SE-100 44 Stockholm, SWEDEN
E-Mail: bosse@mse.kth.se

2 Important Features of the DATAPLOT Graphical Language

2.1 DATAPLOT File STRUCTURE

A DATAPLOT file is a normal textual file with the extension of EXP that can be created with a textual editor or by a program. The file must contain one or more DATASETS and possibly also one or more PROLOGUES. Each PROLOGUE/DATASET is an entity that can be individually selected for plotting.

A PROLOGUE/DATASET is identified by a unique positive number in the file. A PROLOGUE normally contains various DATASET commands for defining information about axis scaling, axis text, axis length, title and so on. A PROLOGUE is terminated by another PROLOGUE or by the first DATASET. This means that all PROLOGUES must be placed at the beginning of the file, before the first DATASET.

A DATASET normally contains various DATASET commands that are associated some separate data points, as well as with one or more BLOCKS of data (calculated or experimental). A DATASET is terminated by another DATASET command or the end of file.

2.2 DATAPLOT Language Syntax

The DATAPLOT language consists of commands with or without parameters:

COMMAND {parameter(s)}

The basic graphical command consists simply of an X/Y coordinate pair and a Graphical Operation Code (GOC). With other commands, the interpretation of this basic command can be modified in many ways. There are separate commands to draw a polygon or a function, and various facilities to obtain texts in many different fonts.

For convenience in editing a DATAPLOT file, the graphical commands can be abbreviated.

Note that a command (with parameters) must not exceed 80 characters. If it is too long (normally as writing necessary codes in a command's parameters for a complex expression), two or more lines can be edited.

2.3 Coordinate Systems

The DATAPLOT language accepts coordinates in three different coordinate systems, which are called word, virtual and normalized, respectively.

The *word* coordinates are those selected by the user, that may represent any kind of data and be of "any" magnitude.

The *normalized* coordinate system goes from zero to one. When plotting, the user must interactively scale each axis by selecting the minimum and maximum word coordinates on the axis. In the normalized coordinate system, the minimum axis value is represented by zero and the maximum by one. DIGLIB will draw a square between the four points (of the X and Y axes) that are determined by the coordinates zero and one in the normalized coordinates. Note that it is also possible to draw triangular plots as described below. However, in most places, we only reference square diagrams. All data points within the minimum and maximum word coordinates will be plotted inside this square. DIGLIB will also write tic marks and corresponding word values at such tic marks.

The *virtual* coordinate system uses centimeter as units. However, the actual size of one unit is dependent on the implementation of the device driver in DIGLIB. It is not recommended to use this coordinate system if different output devices are used for preliminary and final plots.

It may be convenient to use normalized coordinates to draw boxes and texts. The user may give normalized coordinates outside zero and one if the user wishes to write texts outside the area enclosed by the square. To ensure proper operation outside the normalized box, the clipping must be turned off.

2.4 Graphical Operation Codes

The Graphical Operation Code (GOC) determines how the coordinates will be interpreted and also what shall be done at the point determined by the coordinates. For an individual data point, its GOC codes must be given. For each data BLOCK, the user may define a default GOC that will be used for the whole BLOCK, unless a GOC is explicitly defined for a specific point.

The GOC is a combination of the following characters:

Character	***Meaning***
W, V or N	To use Word, Virtual or Normalized coordinates
A or R	To use Absolute or Relative values
M or D	To perform a Move or Draw operation
B	To apply "soft" splines on the line drawn between the coordinate pairs
S	To plot default symbol
S#	To change default symbol to No # and to plot it
'TEXT	To plot any text "TEXT" at the current position N.B. The ' command should be the last operation in a GOC code.

If S is omitted, no symbol will be plotted. A number selecting the symbol can optionally follow the character S. The DIGLIB software determines which symbol the number represents. See *Section 2.10* for more details on plotting symbols.

If any of the other characters are omitted, the default defined for the BLOCK will be used. Absolute values imply the current coordinates; relative values mean that they shall be added to the current coordinates. The GOC must be written without any space between various characters, but the order for the characters is irrelevant.

If no GOC is defined for a BLOCK, the system default is MWA, which means Move Word Absolute, *i.e.*, move to the given point that is interpreted as absolute word coordinates. The GOC=DNA means draw a line from the current point to the new one interpreting the coordinates as normalized. After each draw or move operation, the so-called current point will be the new point.

The TEXT following the ' command is interpreted and expanded by the DIGLIB software, if it includes any ^ operator or STRING names using the ~ descriptor (see *Sections 4.14* and *4.15*). For obtaining PostScript outputs, the TEXT must be edited by the LTEXT's Text Formatting Program instead (see *Chapter 5*).

2.5 Tables or Blocks

In many cases, the user has (calculated or experimental) data in form of tables, and would like to plot one or several columns as X-axis and one or more columns as Y-axis. A single table may contain many sets of data records of the same kind of information for a specific data BLOCK. There can be many data BLOCKS in a DATASET.

By enclosing such a table in a data BLOCK, the user can select which column(s) is the X-axis and which is the Y-axis. For instance,

```
BLOCK X=C1; Y1=C3; Y2=C2; GOC=C4,DAW
```

Where the X-axis values are in column 1, the Y-axis values are in columns 2 and 3, and any GOC codes will be in column 4. The default GOC for this BLOCK is Draw Absolute Word. The GOC code inside the table is only necessary if the default GOC is not applicable.

A more elaborate use of the table is shown in this example:

```
BLOCK X=C3; Y=1E3/(C3+273); GOC=C8,MAWS1
```

Where the X-axis values are in column 3, the Y-axis values are in column 1 but here they are transformed by first adding 273 and then divide by 1000 by the result. Any GOC is in column 8, and the default GOC is Move Absolute Word and plot the Symbol No. 1.

Columns in a table must be separated by one or more space characters. Thus, they do not have to be justified.

It is possible to have tables with mixed text and numbers, but the user must be aware of the fact that each word followed by a space is counted as one column. Of course, the columns used for plotting must be numerical. An example of a legal line in a table is

```
298.15 This_is_the_second_column 11.5 This_is_the_fourth_column
```

Note that a line in the table must not exceed 80 characters. A `BLOCK` must be terminated by a line with the `BLOCKEND` command.

2.6 Drawing a Polygon

Normally, each point is written on a separate line. But in order to draw a line in a more compact way, the use may use the command `DRAWLINE`. `DRAWLINE` is followed by a couple of X/Y pairs of numbers. The X/Y pairs must be separated by a space, and there must be a comma sign between the X and Y values. `DRAWLINE` makes a *move* operation to the first pair of X/Y coordinates, and then draw a line among all pairs up to the last one. All pairs must fit on one line of 80 characters, but the user may of course have several consequent `DRAWLINE` commands.

2.7 Drawing an Analytical Function

As was shown earlier, it is possible in a `BLOCK` to set an axis to a function. It is not even necessary to use a value from any column in order to compute the function value to be plotted. In order to plot a function with an even increment of the independent variable, there is a command `FUNCTION`.

2.8 Painting of an Enclosed Area

It is possible to paint or fill an area in a specified pattern in the plot with the `PAINT` command. Available patterns are determined by the DIGLIB software. Related command to `PAINT` is `PCFUNCTION`. At present, the `PAINT` command only works on PostScript devices; see *Section 5.9* and *Figure 5* for all types of possible patterns available for various fonts in the PostScript format.

2.9 Writing a Text

In order to write a text, the user may use the `TEXT` command. This will write the text at the current point.

It is also possible to write a text at any X/Y pair by appending a single quote followed by the text on the same line. For example,

```
1.1 1.0 NAM'This is a text
```

will cause the text "`This is a text`" to be written at the normalized coordinates (`1.1, 1.0`).

The user may select the font used for the text by the `FONT` command, and the size of the characters by the `CHARSIZE` command. This size of the symbols can be set with a `SYMBOLSIZE` command.

If a text or a single character should be of a different font than all the other text, or if the user would like to use subscripts or superscripts in a text, it is necessary to use the `^` operators or `STRING` command to create the text (see *Sections 4.14* and *4.15*). The `STRING` command will store, in a variable specified by the user, the text including all text-formatting information defined by the DIGLIB software; this is shown in *Example 3* (*Section 6.3*) and *Example 5* (*Section 6.5*).

However, if a graphical output is done on a PostScript device using the PostScript hardcopy fonts, special text formatting codes as presented in *Chapter 5* (the LTEXT Text Formatting Program) should be used. Note that the above-mentioned `STRING` formatting syntax will then not be valid.

2.10 Plotting a Symbol

As described in *Section 2.4*, a GOC code in a data BLOCK may contain an S option to plot a symbol for a X/Y pair or the same symbols for the while data BLOCK. A number selecting the symbol may optionally follow the character S. Like writing texts, it is also possible to plot a symbol at any current X/Y position by appending a quote specified by the symbol number in the GOC code (*e.g.*, 1.1 1.0 MANS5'This is a text).

Figure 1 summarizes all standard symbols available in the DIGLIB software (*Figure 1a* as printed from the Thermo-Calc/DICTRA Graph window, and *Figure 1b* as saved in EMF format). Note that a default symbol is the current symbol in the current run of the Thermo-Calc/FOP/DICTRA software (it is usually the No 1 symbol if the POST-processor is switched on for the first time). S (*i.e.*, # is not specified) means that the current symbol is plotted. S0 (*i.e.*, #=0) means that no symbol is plotted.

Note that all the DIGLIB symbols work properly and give very nice output results for the PostScript format (as illustrated in *Figure 1c*, graphical files viewed by PostScript-supporting graphical software, or hardcopies printed on PostScript-supporting devices).

One can also insert DIGLIB symbols into LaTeX-edited documents; for details, refer to *Section 2.13* and *Figure 1d*.

2.11 Other Commands and Miscellaneous

Line type can be solid, long dashed, short dashed or dotted. This can be selected by the LINETYPE command. If the user has a color device, it is also possible to change color on the lines with the COLOR command. On some non-color devices, colors are simulated with different width and dashing of the lines.

By default, all data outside the normalized coordinates zero and one are not plotted. This can be changed by using the CLIP command.

When plotting symbols representing various experimental data, it is important that the symbols are centered around the coordinate values.

When writing a text, the user would instead often like to give the coordinates of the lower left corner of the first character in the text. This is the default case, but the user may change it by the ATTRIBUTE command.

The user may create libraries with *e.g.* texts and include these in many similar plots by using the INCLUDE command.

The dollar sign $ as the first character of a line stands for a comment character, and thus the whole line will be ignored when plotting.

2.12 Interactive Plotting

The DATAPLOT file is read into the POST-processor of the Thermo-Calc, FuncOptPlot or DICTRA workspaces by the APPEND_EXPERIMENTAL_DATA or QUICK_EXPERIMENTAL_PLOT commands. These commands will ask for the name of the DATAPLOT file and also which PROLOGUE(S) and DATASET(S) to be plotted.

By giving the PROLOGUE/DATASET number as −1, the user will obtain a list of the available PROLOGUES/DATASETS in the file. Note that if "DATASET 0" is present in a DATAPLOT file, its data will always be used independent on which other DATASETS have been chosen.

2.13 Formatting DIGLAB Symbols in LaTeX Documents

When writing papers using the LaTeX editor, one may sometimes refer to use the DIGLIB symbols in texts, in addition to in figures (which are described in other sections of this document). Naturally, this is very useful necessarily/appropriately referring to the corresponding LaTeX symbols (closest to those DIGLIB symbols

which have been plotted on a Thermo-Calc/DICTRA figure using the DATAPLOT Graphical Language) in the texts of LaTeX documents for publications/reports.

This section shows how to generate some DIGLIB symbols in texts, through the attached LaTeX source file (`DIGLIB_Sym.tex` listed below) and its converted jpg file (`DIGLIB_Sym.jpg` as in *Figure 1d*).

```
\documentclass[dvips,12pt]{article}
\textwidth 165mm
\textheight 225mm
\oddsidemargin 1mm
\evensidemargin 1mm
\topmargin 1mm
%%\usepackage{amssymb}
%% next replace amssymb and to get udtimes
\usepackage[utopia]{mathdesign}
\usepackage{rotating}
\usepackage[latin1]{inputenc}
\usepackage{graphics}
\usepackage{graphicx,subfigure}   % with figures
%\usepackage[draft]{graphicx}     % without figures
\usepackage{subfigure}            % with figures
\topmargin 1mm
\oddsidemargin 1mm
\evensidemargin 1mm

\begin{document}

{\Large \bf Diglib symbols and their corresponding LaTeX symbols}
\vspace{5mm}
```

The table below gives the closest corresponding LaTeX symbol. All symbols (except for +) must be generated in math mode. Most of these require the package **amssymb**, *i.e.*, one must have a directive `usepackage{amssymb}` in the preamble. Two of the symbols require the more extensive **mathdesign** which can be included with `usepackage[utopia]{mathdesign}`.

```
\vspace{5mm}
{\Large
\begin{tabular}{llll}
Diglib & Latex & Latex name & Note\\
1 & $\vartriangle$     & $\backslash$vartriangle & amssymb \\
2 & $\square$          & $\backslash$square\\
3 & {\Huge $\diamond$} & $\backslash$diamond & size $\backslash$Large \\
4 & $\udtimes$         & $\backslash$udtimes & mathdesign \\
5 & $\triangledown$    & $\backslash$triangledown & amssymb\\
6 & +                  & normal + \\
7 & $\ast$             & $\backslash$ast & amssymb\\
8 & $\times$           & $\backslash$times \\
9 & {\Huge $\circ$}    & $\backslash$circ & size $\backslash$Large \\
10 & {\Huge $\star$}   & $\backslash$star & size $\backslash$Large, amssymb\\
11 & $\curlyvee$       & $\backslash$curlyvee & amssymb\\
12 & $\Join$           & $\backslash$Join\\
13 &                   & - & nothing similar, overlapping $> <$\\
14 &                   & - & nothing similar, 10-edged star \\
15 & $\maltese$        & $\backslash$maltese  & mathdesign \\
16 &                   & - & nothing similar, a pentagon \\
17 & $\curlywedge$     & $\backslash$curlywedge & amssymb\\
\end{tabular}}

\end{document}
```

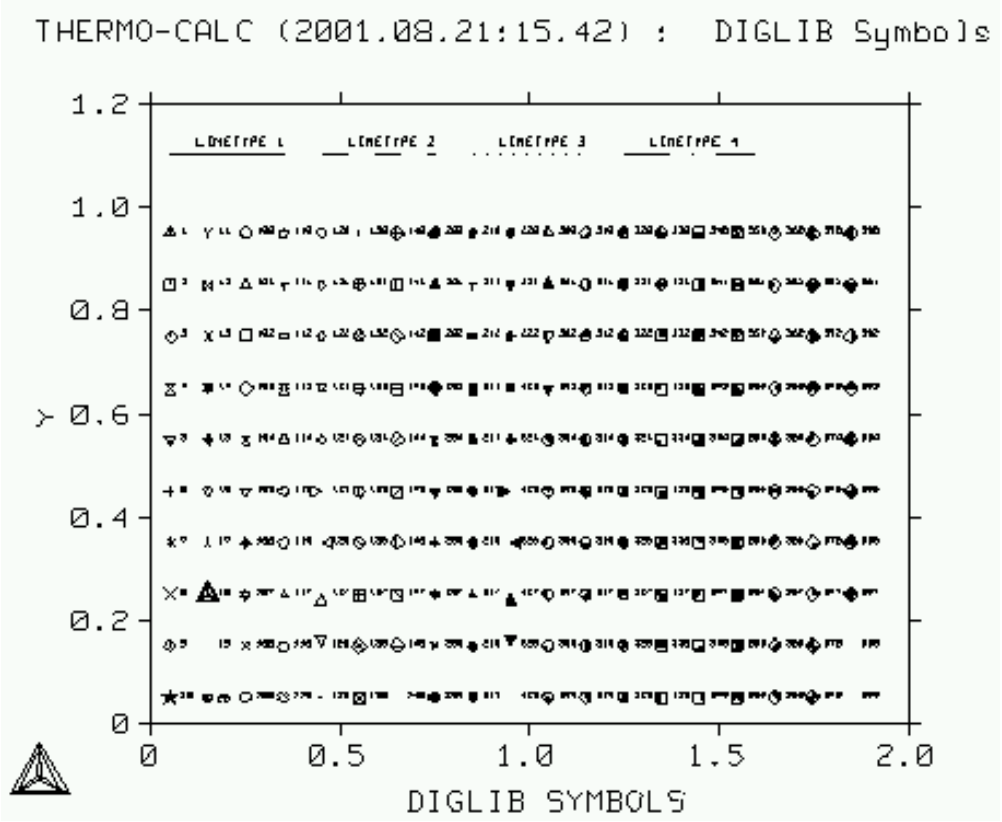


Figure 1a. Examples of All the DIGLIB Symbols (printed from TC Graph Window)

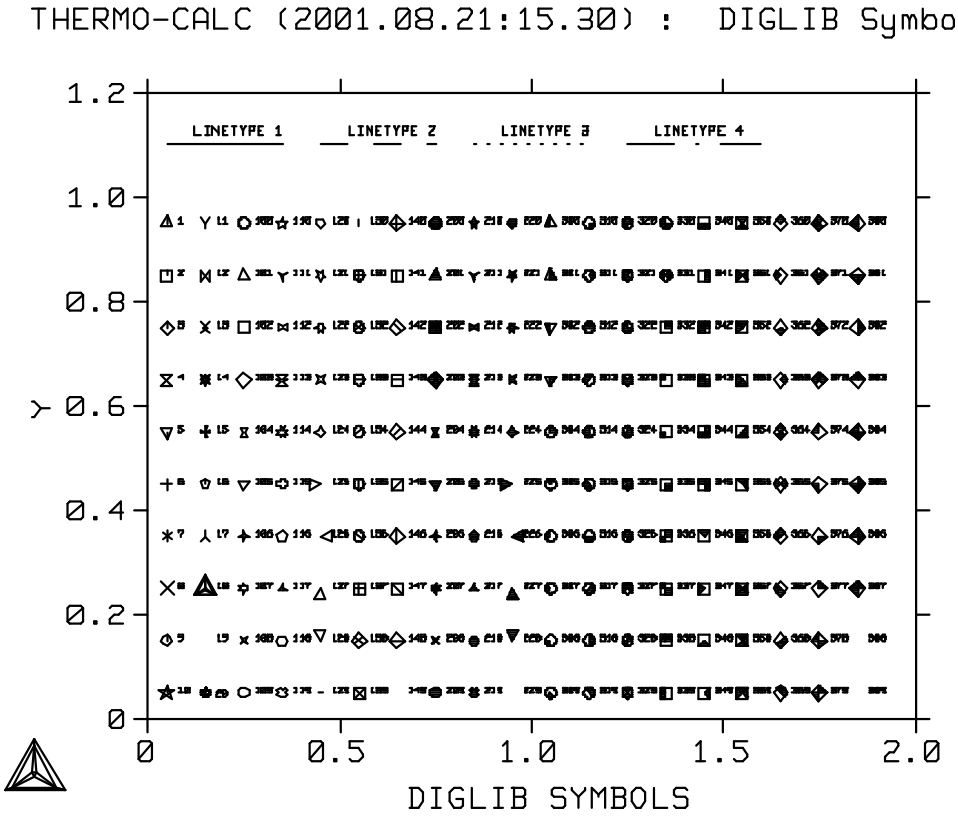


Figure 1b. Examples of All the DIGLIB Symbols (in EMF Format)

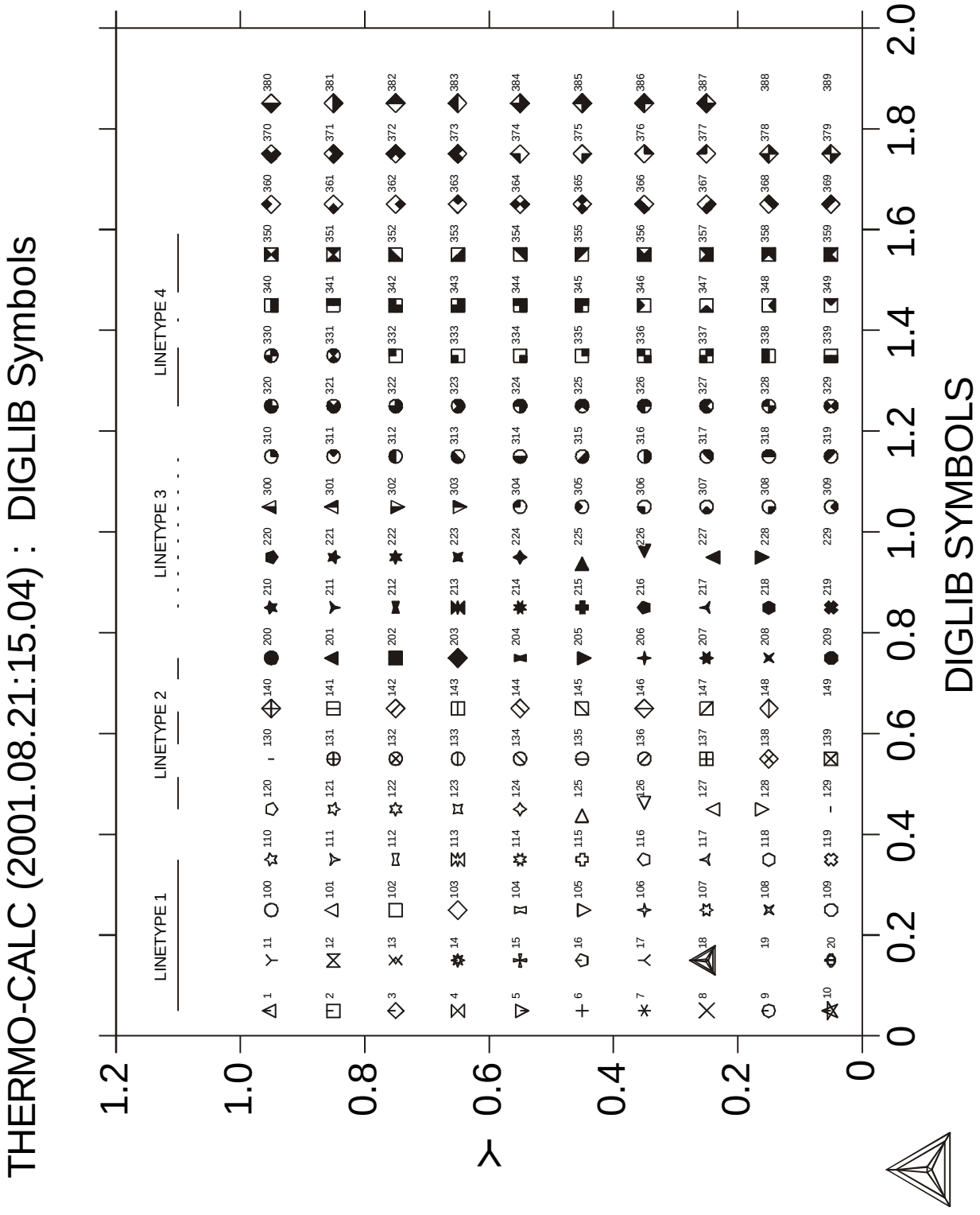


Figure 1c. Examples of All the DIGLIB Symbols (in PostScript Format)

Diglib symbols and their correponding LaTeX symbols

Diglib	Latex	Latex name	Note
1	△	vartriangle	amssymb
2	□	square	
3	◇	diamond	
4	⧖	udtimes	mathdesign
5	▽	triangle down	amssymb
6	+	plus	
7	∗	ast	amssymb
8	×	times	
9	∘	circ	
10	⋆	star (5 edges)	amssymb
11	⋎	curlyvee	amssymb
12	⋈	Join	
13		-	nothing similar, combined ><
14		-	nothing similar, 10-edged star
15	✠	maltese	mathdesign
16		-	nothing similar, a pentagon, almost like ci
17	⋏	curlywedge	amssymb

Test
>< ⋊ ⋉
⋖

Figure 1d. Examples of Some DIGLIB Symbols (in LaTeX Format)

3 PROLOGUE Commands

Below follows a presentation of the PROLOGUE command and subsequently a list of all legal commands in a PROLOGUE.

3.1 PROLOGUE

Description: The PROLOGUE command indicates the beginning of a number of consequent lines of user-defined diagram layout manipulating commands. The PROLOGUE lines are displayed on the terminal along with the text "optional text" when using the -1 option in the POST-processor as prompted for the PROLOGUE number in the APPEND_EXPERIMENTAL_DATA or QUICK_EXPERIMENTAL_PLOT command.

Synopsis: PROLOGUE # optional text

Notes: # is an unsigned integer identifying the PROLOGUE.

3.2 XSCALE

Description: The XSCALE command sets the scaling in word coordinates of the X-axis.

Synopsis: XSCALE min max

Notes: min and max are real numbers.

3.3 YSCALE

Description: The YSCALE command sets the scaling in word coordinates of the Y-axis.

Synopsis: YSCALE min max

Notes: min and max are real numbers.

3.4 XTEXT

Description: The XTEXT command sets the X-axis text.

Synopsis: XTEXT text

Notes: text is an arbitrary text string that may contain text-formatting codes.

3.5 YTEXT

Description: The YTEXT command sets the Y-axis text.

Synopsis: YTEXT text

Notes: text is an arbitrary text string that may contain text-formatting codes.

3.6 XTYPE

Description: The XTYPE command sets the X-axis type as linear (default), logarithmic or inverse.

Synopsis: XTYPE type

Notes: type is a character string reading LIN, LOG or INV.

3.7 YTYPE

Description: The YTYPE command sets the Y-axis type as linear (default), logarithmic or inverse.

Synopsis: YTYPE type

Notes: type is a character string reading LIN, LOG or INV.

3.8 XLENGTH

Description: The XLENGTH command sets the X-axis length to approximately # centimeters.

Synopsis: XLENGTH #

Notes: # is a positive real number (the approximate X-axis length in centimeters).

3.9 YLENGTH

Description: The YLENGTH command sets the Y-axis length to approximately # centimeters.

Synopsis: YLENGTH #

Notes: # is a positive real number (the approximate Y-axis length in centimeters).

3.10 TIC_TYPE

Description: The TIC_TYPE command sets the relative length of the tic marks. Default value is 1. Negative number gives tics on the inside of the diagram frame. 0 gives no tics.

Synopsis: TIC_TYPE #

Notes: # is a real number.

3.11 TITLE

Description: The TITLE command sets the title text string to be printed above the diagram.

Synopsis: TITLE text

Notes: text is an arbitrary text string that may contain text-formatting codes

3.12 DIAGRAM_TYPE

Description: The DIAGRAM_TYPE command sets the diagram type to square (which is default) or triangular.

Synopsis: DIAGRAM_TYPE type plot_3rd_axis clip_along_third-axis

Notes: type is a character string reading SQUARE (default) or TRIANGULAR. If type reads TRIANGULAR, then two additional parameters should be given namely: "plot_3rd_axis" and "clip_along_third-axis" that are characters strings reading YES or NO.

4 DATASET Commands

Below follows a presentation of the DATASET command and subsequently a list of all legal commands in a DATASET.

4.1 DATASET

Description: The DATASET command indicates the beginning of a number of consequent lines comprising a set of user-defined data. The DATASET lines are displayed on the terminal along with the text "optional text" when using the -1 option in the POST-processor as prompted for the DATASET number in the APPEND_EXPERIMENTAL_DATA or QUICK_EXPERIMENTAL_PLOT command.

Synopsis: DATASET # optional text

Notes: # is an unsigned integer identifying this set of data.

4.2 BLOCK

Description: The BLOCK command defines how the following numeric data block shall be interpreted. The definitions of X and Y coordinates may also be expressed as a function of the column values, making it possible to perform transformations.

Synopsis: BLOCK X&=C#; ...; Y&=C#; ...; GOC=C#,@@@...

Notes: &' are optional unsigned integers that make it possible to plot several (maximum 9) X- or Y-axis columns. #' are unsigned integers identifying the column numbers. The column number # in "GOC=C#" is the location of any possible GOC codes in the current data BLOCK; @@@ stands for the default Graphical Operation Code (GOC) for the current BLOCK. The GOC code inside the current table is only necessary if the current default GOC is not applicable.

Legal GOC characters include:

Character	***Meaning***
W	Word coordinates *(default)*
V	Virtual coordinates
N	Normalized coordinates
A	XY are absolute values *(default)*
R	XY are relative values
M	Move to XY *(default)*
D	Draw to XY
B	Apply "soft" splines on a line drawn (used only on BLOCK data)
S	Plot current symbol at XY
S#	Change current symbol to No # symbol, and plot it at XY
'TEXT	Plot the text "TEXT" at XY (it must appear last in the GOC code)

Note that the TEXT following the ' command is interpreted and expanded by the DIGLIB software, if it includes any ^ operator or STRING names using the ~ descriptor. For obtaining PostScript outputs, the TEXT must be edited by the LTEXT Text Formatting Program instead (see *Chapter 5*).

4.3 BLOCKEND

Description: The BLOCKEND command terminates the local definition of the graphical operation code defined by the earlier BLOCK command.

Synopsis: `BLOCKEND`

4.4 DATAPOINT

Description: DATAPOINT is not actually a DATASET command, but the basic DATAPLOT command (see *synopsis* below) performs an action at the current point determined by the specified X/Y-coordinates.

A DATASET may contain various data points, in addition to one or more data BLOCKS (see *Section 4.2*). Such data points are separated and independent on each other.

Synopsis: `X Y GOC`

Notes: `X` and `Y` are unsigned real numbers identifying the X/Y-coordinates for the current data point. `GOC` stands for Graphical Operation Code (`GOC`) for the current point. Legal GOC characters are listed in *Section 4.2*.

Examples:
```
0.7 0.95 N'Example 6
0.5 0.08 MNA'E^FS18^SQ(^SK^FS10A+5#8*C#^FS10 -!a^FS18)^FS11+B^DIa#b#$
```

4.5 CLIP

Description: The `CLIP` command turns clipping on or off. If it is `OFF`, it allows output outside the ordinary plot area defined by normalized coordinates zero and one.

Synopsis: `CLIP clp`

Notes: `clp` is a character string reading `ON` or `OFF`.

4.6 ATTRIBUTE

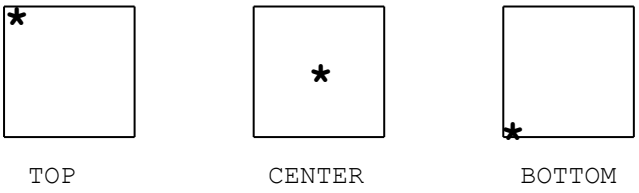
Description: The `ATTRIBUTE` command specifies where the current XY position is in the character or symbol plotbox.

Synopsis: `ATTRIBUTE attribute`

Notes: `attribute` may be `TOP`, `CENTER` or `BOTTOM`.

`ATTRIBUTE CENTER` is default for symbols;
`ATTRIBUTE BOTTOM` is default for characters.

Examples: Character plotboxes



4.7 LINETYPE

Description: The `LINETYPE` command redefines the current linetype in the plot.

Synopsis: `LINETYPE #`

Notes: `#` must be an unsigned integer.

Legal linetypes are (for both normal graphical outputs and for PostScript formats, as illustrated in *Figures 1* and *3*):

Number	***Linetype***
1	solid *(default)*
2	long dashed
3	short dashed
4	dotted

4.8 DRAWLINE

Description: The DRWALINE command draws a line starting at (x1,y1) to (xn,yn) through (x2,y2) → (x(n-1), y(n-1)).

Synopsis: DRAWLINE x1,y1 x2,y2 ..... xn,yn

Notes: x and y may be reals or integers of any value.

This is identical to connecting all points in a table:

```
x1   y1    M
x2   y2    D
...  ...   D
xn   yn    D
```

4.9 CHARSIZE

Description: The CHARSIZE command redefines the default size of the characters in the plot. The character size has an initial default value, which may vary with the current font setting and the output device (the plot format).

Synopsis: CHARSIZE $$

Notes: $$ must be an unsigned real.

4.10 SYMBOLSIZE

Description: The SYMBOLSIZE command redefines the current symbol size setting. The symbol size has an initial default value.

Synopsis: SYMBOLSIZE $$

Notes: $$ must be an unsigned real.

4.11 GLOBALSIZE

Description: The GLOBALSIZE command redefines the default global size of the plot. The global size has an initial default value.

Synopsis: GLOBALSIZE $$

Notes: $$ must be an unsigned real.

4.12 COLOR

Description: The COLOR command redefines the current color setting.

Synopsis: COLOR code

Notes: code is an unsigned integer number (from 0 to 21) identifying the color, or a character string specifying the color, as demonstrated in *Example 10* (*Section 6.10*).

There are 22 legal color codes (from 0 to 21) as listed below, and their color effects are shown in *Figure 2*.

Code	***Color***	***Equivalent Character***
0	*BACKGROUND*	*INVISIBLE*
1	BLACK	NORMAL
2	RED	VERY_THICK
3	GREEN	THIN
4	BLUE	THICK
5	YELLOW	VERY_THIN
6	MAGENTA	DASHED
7	CYAN	DOTTED

8	PURPLE	
9	GOLD	
10	TURQUOISE	
11	PINK	
12	GRAY	
13	ORAGNERED	
14	MAROON	
15	PLUM	
16	SEAGREEN	
17	OLIVEDRAB	
18	SIENNA	
19	ORANGE	
20	CORAL	
21	UserDef	

Any digit larger than 21 will have the color effect same as 1 (black).

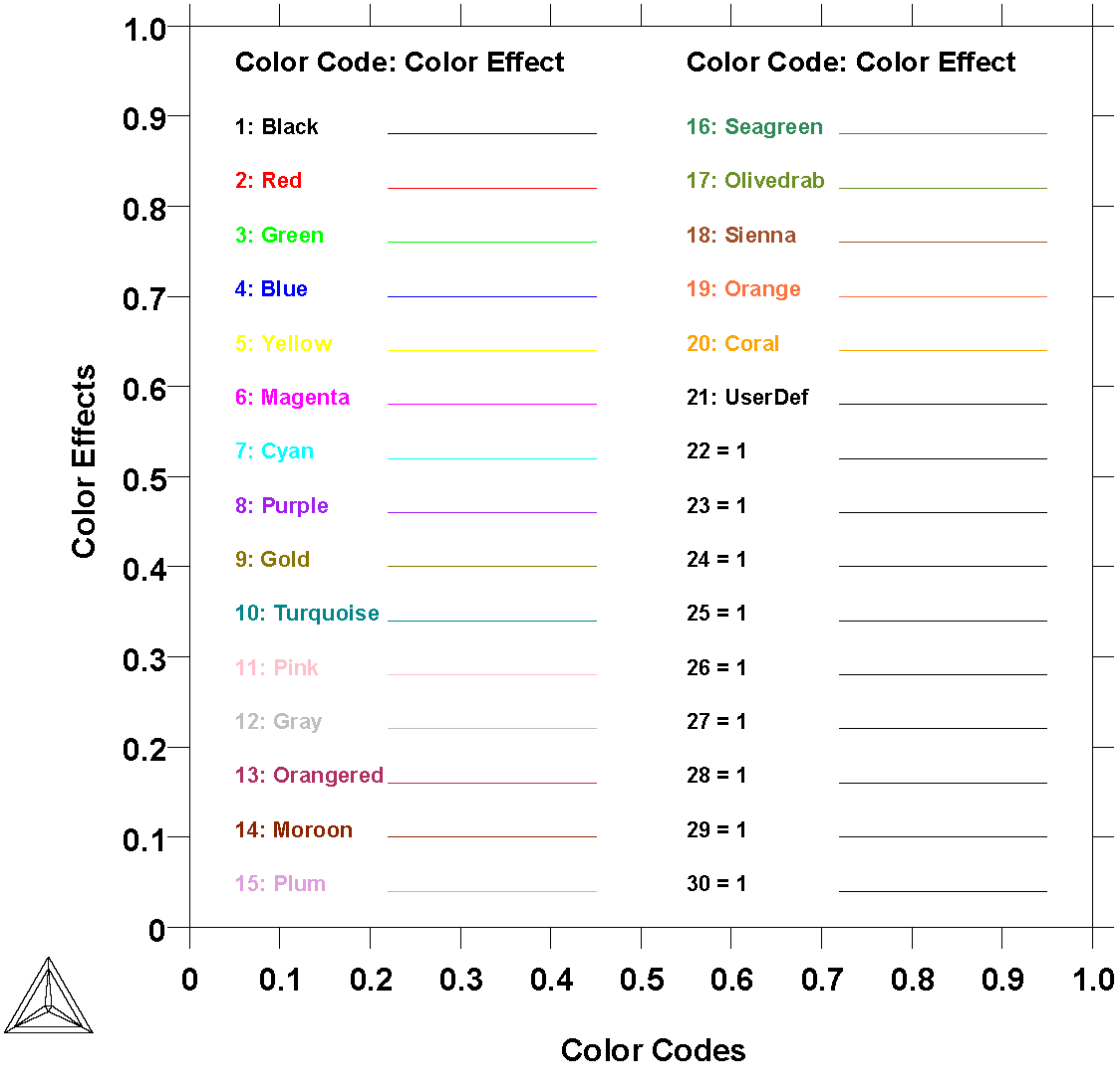


Figure 2. All the DIGLIB Color Codes and Their Color Effects

4.13 FONT

Description: The FONT command redefines the default font setting in the POST-processor.

Synopsis: FONT #

Notes: # is an unsigned integer.

Legal fonts are:

Number	*Font Name*
1	Cartographic Roman *(default)*
2	Bold Roman Script
3	Bold Roman
4	Bold Italic
5	Script
6	Bold Script
7	UNCIAL
8	Bold Greek
9	Gothic English
10	Gothic Greek

Their outputs are illustrated in *Figure 3*.

Note: For the use of the PostScript hardcopy fonts, the font assignments are different and more fonts are optionally available; see *Section 5.7* and *Table 1*.

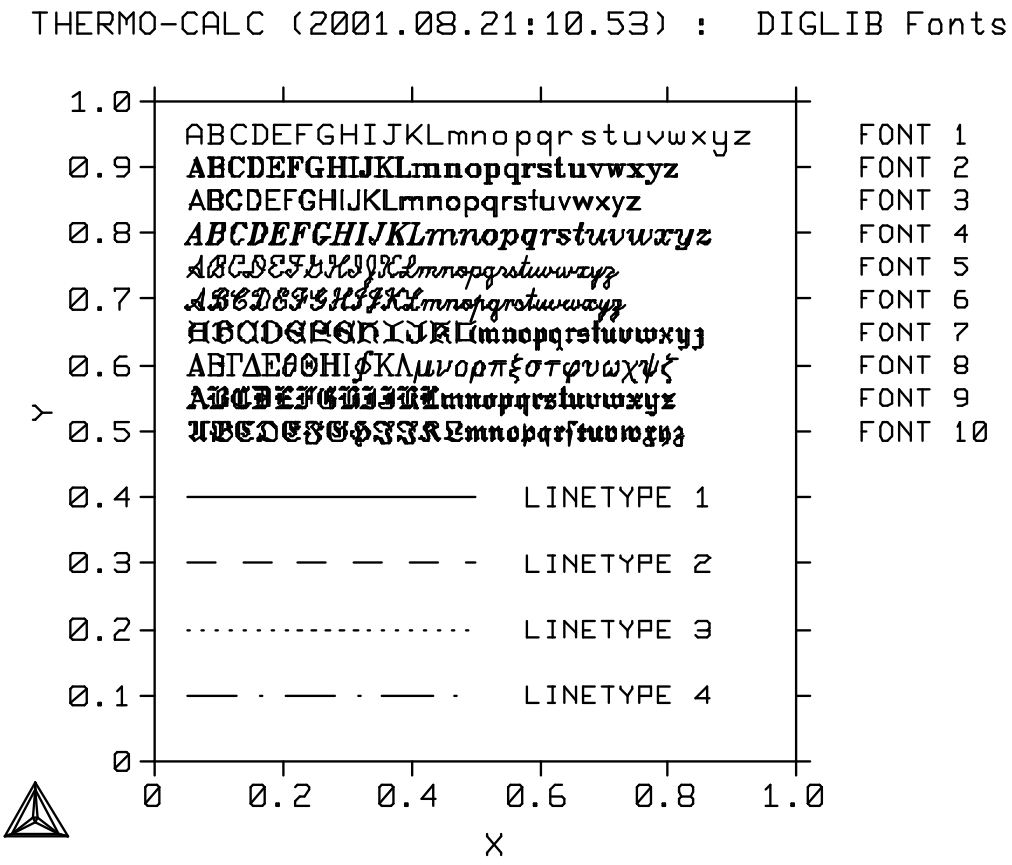


Figure 3. Examples of All the DIGLIB Fonts and Line Types

4.14 STRING

Description: The `STRING` command defines a string containing a text and operation codes (*e.g.*, to change the default font settings).

Synopsis: `STRING name text_with_each_character_in_^S#^G^F#^U#^D#^R#^L#^N`

Notes: `name` is a valid alphanumeric name (variable) to represent the `text` including all text-formatting codes. It is highly recommended that all the characters in the `name` shall be written in CAPITAL CASE (such as ACA2SO4); otherwise, the defined STRING may sometimes be incorrectly plotted subsequently.

"`text_with_each_character_in_^S#^G^F#^U#^D#^R#^L#^N`" means the text is coded with each of its characters (and/or numeric numbers) that are formatted with various operators (`^S#`, `^G`, `^F#`, `^U#`, `^D#`, `^R#`, `^L#` and/or `^N`). `#`' are unsigned integers. `^` is the caret character and does not mean a control character. Between an `#` (in an operator) and a numeric number (as a part of the text), there must always be a comma sign "`,`"; otherwise, the number will not be plotted in the text, because the `#` with the number will be interpreted as another incorrect `#`.

Legal string operation codes in the DIGLIB software are:

Operator	***Operation***
`^S#`	Set character size to size #
`^G`	Set font to Greek
`^F#`	Set font to font number #
`^U#`	Move up # units
`^D#`	Move down # units
`^R#`	Move right # units
`^L#`	Move left # units
`^N`	Not move, remain at current position

In some of the commands, `#`=0 resets the option to previous (or default) value.

Examples:

```
STRING Alpha1 ^Ga^F0^D0^S8,1^S0^U0                          ➔ α1
STRING M23C6 M^D0^S8,23^S0^U0C^D0^S8,6^S0^U0                ➔ M23C6
STRING ACA2CO3 ^Ga^F0^D0^S8Ca^D0^S4,2^S0^U0^S8CO^D0^S4,3^S0^U0
                                                            ➔ αCa2CO3
```

When using the PostScript hardcopy fonts, the above operation codes are NOT valid; see instead *Chapter 5* (the LTEXT Text Formatting Program).

4.15 TEXT

Description: The `TEXT` command outputs, at the current position, the `text` following the keyword `TEXT` or the text in the string "`string_name`" that has previously been defined with the `STRING` command.

Synopsis: `TEXT text` or `~string_name`

Notes: It is possible to mix `^` operators and previously defined string names using the `~` operator.

When using the PostScript hardcopy fonts, the above `^` operators are NOT valid; see instead *Chapter 5* (the LTEXT Text Formatting Program).

4.16 FUNCTION

Description: The FUNCTION command defines and plots a user-defined function.

Synopsis 1: `FUNCTION Y=f(X); start end number_of_XY_pairs; GOC;`

Synopsis 1: `FUNCTION X=f(Y); start end number_of_XY_pairs; GOC;`

Notes: `f(X)` or `f(Y)` are legal mathematical functions of `X` or `Y`, understandable by a FORTRAN program. `start` and `end` are unsigned real numbers, and `number_of_XY_pairs` an unsigned integer. `GOC` is a legal graphical operation code as defined in `BLOCK` command definition (see *Section 4.2*).

4.17 PCFUNCTION

Description: The `PCFUNCTION` command appends a user-defined function to the current path. It is used together with the `PAINT` command (see *Section 4.16*).

Synopsis 1: `PCFUNCTION Y=f(X); start end number_of_XY_pairs; GOC;`

Synopsis 1: `PCFUNCTION X=f(Y); start end number_of_XY_pairs; GOC;`

Notes: `f(X)` or `f(Y)` are legal mathematical functions of `X` or `Y`, understandable by a FORTRAN program. `start` and `end` are unsigned real numbers, and `number_of_XY_pairs` an unsigned integer. `GOC` is a legal graphical operation code as defined in `BLOCK` command definition (see *Section 4.2*).

4.18 PAINT

Description: The `PAINT` command paints the area enclosed by the current path in the current pattern. The current path starts at the last `MOVETO` given and includes all subsequent `DRAW`s. Also see `PCFUNCTION` command (in *Section 4.17*).

However, this command only works for the PostScript format (as graphical files or on printed hardcopy) at present.

Synopsis: `PAINT <code> <video> <mode>`

Notes: `<>` denotes optional parameters. To set a new current pattern, supply any or all of the optional parameters.

`<code>` is a single letter `0-9`, `A-Z` or `a-t` (if `<code>=t`, also supply a space and a number in the range 0.00-1.00).

`<video>` is a string reading `NORMAL` or `INVERSE`.

`<mode>` is a string reading `TRANSPARENT` or `OPAQUE`.

Default parameters are: `<code>=0`, `<video>=NORMAL`, `<mode>=TRANSPARENT`.

All PostScript paint patterns are presented in *Section 5.9* and *Figure 5*.

4.19 INCLUDE

Description: The `INCLUDE` command includes a file into the current input stream (for a demonstration, see *Example 5* in *Section 6.5*).

Synopsis: `INCLUDE filename`

Notes: `filename` is a legal filename (with its correct path) for the operation system.

5 LTEXT's PostScript Formatting Codes

When using PostScript hardcopy fonts, a special set of text formatting codes is available. These codes are in principle identical to those used by the ***LTEXT Text Formatting Program***. The text formatting codes may be introduced into your character strings in order to modify the appearance of your texts on the printed sheet or on the PostScript-format graphical file (which can then be imported into your documents for various purposes).

Normally, a CODE starts with the character caret ^ (ASCII 94) and is followed by a two-letter mnemonic for identification. Note that the code mnemonic may be written in UPPER or lower cases.

To write a ^ on the printed copy or PostScript-format file, type in ^^. The most common CODES are those for setting font type (^fo**), font size (^fs**) and for making superscript (^up...$) or subscript (^do...$) indexes.

The CODES may be divided into the following five different categories:

1. Those taking no argument;
2. Those taking one argument;
3. Those taking one string argument;
4. Those taking two string arguments;
5. Those taking more arguments.

5.1 Codes Taking No Argument

The following codes do not need any argument for formatting texts:

Code	***Operation***	***Resulting Text***
^HX	Write 15h-bar (*i.e.*, Plank constant divided by two pi)	
^LN	Write Natural logarithm	***ln***
^LB	Start large parenthesis	**(**
^EB	End large parenthesis	**)**
^LC	Start large curled brace	**{**
^EC	End large curled brace	**}**
^LS	Start large squared bracket	**[**
^ES	End large squared bracket	**]**
^VB	Write large vertical bar	|
^SL	Write large slash	/
^QS	Start large square root sign	√
^QE	End large square root sign (*i.e.*, draws a line to the root sign)	√‾‾‾‾‾

The codes can be written in either UPPER, or lower case, or mixed. Note that the code ^HX may not be working properly for some PostScript fonts.

Example 8 gives an illustration on how such codes are written in a DATAPLOT (EXP) file, and how the resulting texts look like in a PostScript graphical file (as viewed by any PostScript-supporting graphical program) or on a printed copy.

There is a special codes taking on argument (it is just an operation, without any resulting text):

CTRL-M Move to the leftmost position on the next line (*i.e.*, Carriage return)

Note that CTRL-M is the CONTROL_M option (not as written characters) in the DATAPLOT textual file, so it is equal to a <RETURN> action. It starts a new line for coding the text formatting and/or X/Y coordinates.

5.2 Codes Taking One Argument

The following codes need one argument for formatting texts:

Code	*Operation*	*Example*
^FO**	Put font ** as current font (default **=05)	^FO27
^IF**	Put font ** as current index font (default **=05)	^IF9
^FS**	Put font size ** as current font size (default **=12)	^FS5
^RO**	Rotate the line ** degree counter-clockwise (-90≤**≤90)	^RO30
^CCxY	Compose character is used to create letters that are not normally accessible, but can be composed from an ordinary letter and a sign. Compose character is used for signs above short lower case letters and signs below all letters. The first character after ^CC is the base letter (x) and the second character (Y) is the code of the sign according to the table below.	(see the table below)
^CUXY	Compose character in UPPER case is used to create letters that are not normally accessible, but can be composed from an ordinary letter and a sign. Compose character in UPPER case is used for signs above upper case letters and tall lower case letters. The first character after ^CU is the base letter (X) and the second character (Y) is the code of the sign according to the table below.	(see the table below)

The following table lists the sign codes that format the second characters in the codes ^CCxY and/or ^CUXY:

Code	*Sign*	*Example* (in ^CCaY and ^CUAY, where Y = corresponding sign code)
a	acute accent	***á, Á***
b	breve	***b***
c	circumflex	***â, Â***
d	dotaccent	***d***
e	dieresis	***ä, Ä***
g	grave accent	***à, À***
j	caron	***j***
m	macron	***m***
o	ogonek	***o***
r	ring	***å, Å***
s	cedilla	***ç, Ç*** (*i.e.*, the French character ç)
t	tilde	***ã, Ã***
u	Hungarian dots	**u**

There are some special codes that take one argument, as given in the following table:

Code	*Operation*	*Example* (Codes → **Result**)
^^	Write ^ in the text.	a^^b → **a^b**
^_	State or end underlining of characters.	a^_bc^_ → **abc**
^#	Write # in the text.	a^#b → **a#b**
^$	Write $ in the text.	a^$b → **a$b**
^!	Write ! in the text.	a^!b → **a!b**
!C	Write the specified character (C = any character in UPPER or lower case) in the symbol font (*i.e.*, the font 29).	a!Db → **aΔb** a!db → **aδb**

CTRL-H%	Backspace of length (0.333*DS*%) where DS is the default font size. The number % is optional and only one digit is significant. If % is absent or equals 0, then % is set to 1.	
CTRL-O%%%	The character corresponding to the octal code %%% in the current ENCODING vector corresponding to the current font (see *Tables 2a, 2b* & *2c*) will be printed. %%% must be three digits; otherwise, the code CTRL-O%%% is ignored.	

5.3 Codes Taking One String Argument

The following codes need one string argument (that is ended by the sign $ or #) for formatting texts:

Code	***Operation***	***Example*** (Codes → **Result**)
^GR text $	Greek letters and symbols.	^GRA=b$ → A=β
^UP index $	Index up	B^Upa3$ → B^{a3}
^DO index $	Index down	D^DO5f$ → D_{5f}
^BI index $	Large-bracket index up	B^BIa3$ → $B^{[a3]}$
^BD index $	Large-bracket index down	D^BD5f$ → $D_{[5f]}$
^SQ text $	Square root sign with text	E^SQa3s$ → E$\sqrt{a3s}$

Note that the special character $ is used as terminator. The character # may also be used for this purpose. See *Section 5.6*. Also see *Section 5.2* (Special codes taking one argument) about writing $ or # in the text.

5.4 Codes Taking Two String Arguments

The following codes need two string arguments (each of them is ended by the sign $ or #) for formatting texts:

Code	***Operation***	***Example*** (Codes → **Result**)
^DI index up $ index down $	Double index	B^DIab → B_b^a
^SU upper limit $ lower limit $	Summation sign	^SUf=1$10$B^DOf$ → $\sum_{f=1}^{10} B_f$
^IN upper limit $ lower limit $	Integral	^INx=0$20$X^UP5$ → $\int_{x=0}^{20} X^5$
^KV dividend $ divisor $	Division	^KVA+5$8*C$ → $\frac{A+5}{8*C}$
^SK dividend $ divisor $	Small division	^SK5$8$ → $\frac{5}{8}$

Note that the special character $ is used as terminator. The character # may also be used for this purpose. See *Section 5.6*. Also see *Section 5.2* (Special codes taking one argument) about writing $ or # in the text.

5.5 Codes Taking Many Arguments

Many arguments that are legal PostScript commands/codes may be used in the ^PS ..... code:

Code	*Operation*	*Note*
^PS	The rest of the line is assumed to contain only legal PostScript codes	This command should be placed on a single line.

Example:

```
0.2 0.3 MNA'^PS xm ym 80 (\238) putsymbol)
```

(For the relevant output, as well as other such codes and their outputs, see *Example 7* in *Section 6.7*).

5.6 Nesting of Codes

It is permitted to nestle different codes inside each other. If a code is nestled, it may be necessary to replace $ with #, because that $ will close all nestlings while # only closes the latest nestling. For instance,

```
0.2 0.3 MNA'E^FS18^SQ(^SK^FS10A+5#8*C# -6^FS18)^FS11*B^DIa#b#$
```

5.7 PostScript Fonts

A PostScript font is identified by its font number. All possible PostScript fonts are listed in *Table 1* and illustrated in *Figure 4*.

To choose a specific PostScript font, use the LTEXT Text Formatting Code ^FO** (for texts) and ^IF** (for indices), where ** is a two digit font number (in case written as a one digit number, it is secure to leave a space after the number; otherwise the following text might be printed incorrectly). Not that some fonts may be absent on a specific PostScript device. If a font is not available, the Courier font (*i.e.*, **=05) is selected if possible. The current font size for texts can be set by the code ^FS**), where ** is a one or two digit font number.

Due to the differences in implementation strategies of the PostScript interpreter made by different manufactures, some PostScript devices will fail to print if the user tries to use a non-present font.

5.8 PostScript Vectors

One may write some special characters and/or symbols that are encoding from corresponding PostScript vectors (see *Example 7* in *Section 6.7*), through the following LTEXT codes or PostScript commands:

```
0.2 0.3 MNA'CTRL-O238
0.2 0.3 MNA'^PS xm ym 80 (\238) putsymbol)
```

A PostScript vector is identified by its vector code that is presented as an octal number %%%. Note that %%% must be three digit; otherwise, the command CTRL-O%%% code or the PostScript ^PS command will be ignored.

However, the outputs might be different, depending upon what font has been switched on. All possible PostScript vectors with the PostScript fonts 1-20 and 30-42 are listed in *Table 2a*; with the font 29 in *Table 2b*; and with the font 43 in *Table 2c*.

5.9 PostScript Paint Patterns

One may paint a specific area with a PostScript paint pattern (see *Example 7* in *Section 6.7*), through LTEXT codes or PostScript commands. A PostScript paint patterns is identified by its pattern code that is a single letter 0-9, A-Z or a-t (if <code>=t, also supply a space and a number in the range 0.00-1.00).

However, the outputs for the case with NORMAL video setting are different from those for the case with INVERSE video setting. All the possible PostScript paint patterns are listed in *Figures 5a* (NORMAL) and *5b* (INVERSE).

Table 1. List of the Available PostScript Fonts

Number	*Font Name*	*Resulting Text*
01	AvantGarde-Book	AvantGarde-Book
02	AvantGarde-Book Oblique	*AvantGarde-Book Oblique*
03	AvantGarde-Demi	**AvantGarde-Demi**
04	AvantGarde-Demi Oblique	***AvantGarde-Demi Oblique***
05	Courier	Courier
06	Courier-Oblique	*Courier-Oblique*
07	Courier-Bold	**Courier-Bold**
08	Courier-Bold Oblique	***Courier-Bold Oblique***
09	Helvetica	Helvetica
10	Helvetica-Oblique	*Helvetica-Oblique*
11	Helvetica-Bold	**Helvetica-Bold**
12	Helvetica-Bold Oblique	***Helvetica-Bold Oblique***
13	LubalinGraph-Book	LubalinGraph-Book
14	LubalinGraph-Book Oblique	*LubalinGraph-Book-Oblique*
15	LubalinGraph-Demi	LubalinGraph-Demi
16	LubalinGraph-Demi Oblique	*LubalinGraph-Demi Oblique*
17	NewCenturySchlbk-Roman	NewCenturySchlbk-Roman
18	NewCenturySchlbk-Italic	*NewCenturySchlbk-Italic*
19	NewCenturySchlbk-Bold	**NewCenturySchlbk-Bold**
20	NewCenturySchlbk-Bold Italic	***NewCenturySchlbk-Bold Italic***
21	Souvenir-Light	Souvenir-Light
22	Souvenir-Light Italic	*Souvenir-Light Italic*
23	Souvenir-Demi	**Souvenir-Demi**
24	Souvenir-Demi Italic	***Souvenir-Demi Italic***
25	Time-Roman	Time-Roman
26	Time-Italic	*Time-Italic*
27	Time-Bold	**Time-Bold**
28	Time-Bold Italic	***Time-Bold Italic***
29	Symbol	Σψμβολ
30	Helvetica-Narrow	Helvetica-Narrow
31	Helvetica-Narrow-Bold	**Helvetica-Narrow-Bold**
32	Helvetica-Narrow-Oblique	*Helvetica-Narrow-Oblique*
33	Helvetica-Narrow-Bold Oblique	***Helvetica-Narrow-Bold Oblique***
34	Bookman-Demi	**Bookman-Demi**
35	Bookman-Demi Italic	***Bookman-Demi Italic***
36	Bookman-Light	Bookman-Demi
37	Bookman-Light Italic	*Bookman-Light Italic*
38	Palatino-Roman	Palatino-Roman
39	Palatino-Bold	**Palatino-Bold**
40	Palatino-Italic	*Palatino-Italic*
41	Palatino-Bold Italic	***Palatino-Bold Italic***
42	ZapfChancery-Medium Italic	*ZapfChancery-Medium Italic*
43	ZapfDingbats	&✓⑦✍☚⓪⑤✑✓❶⑩

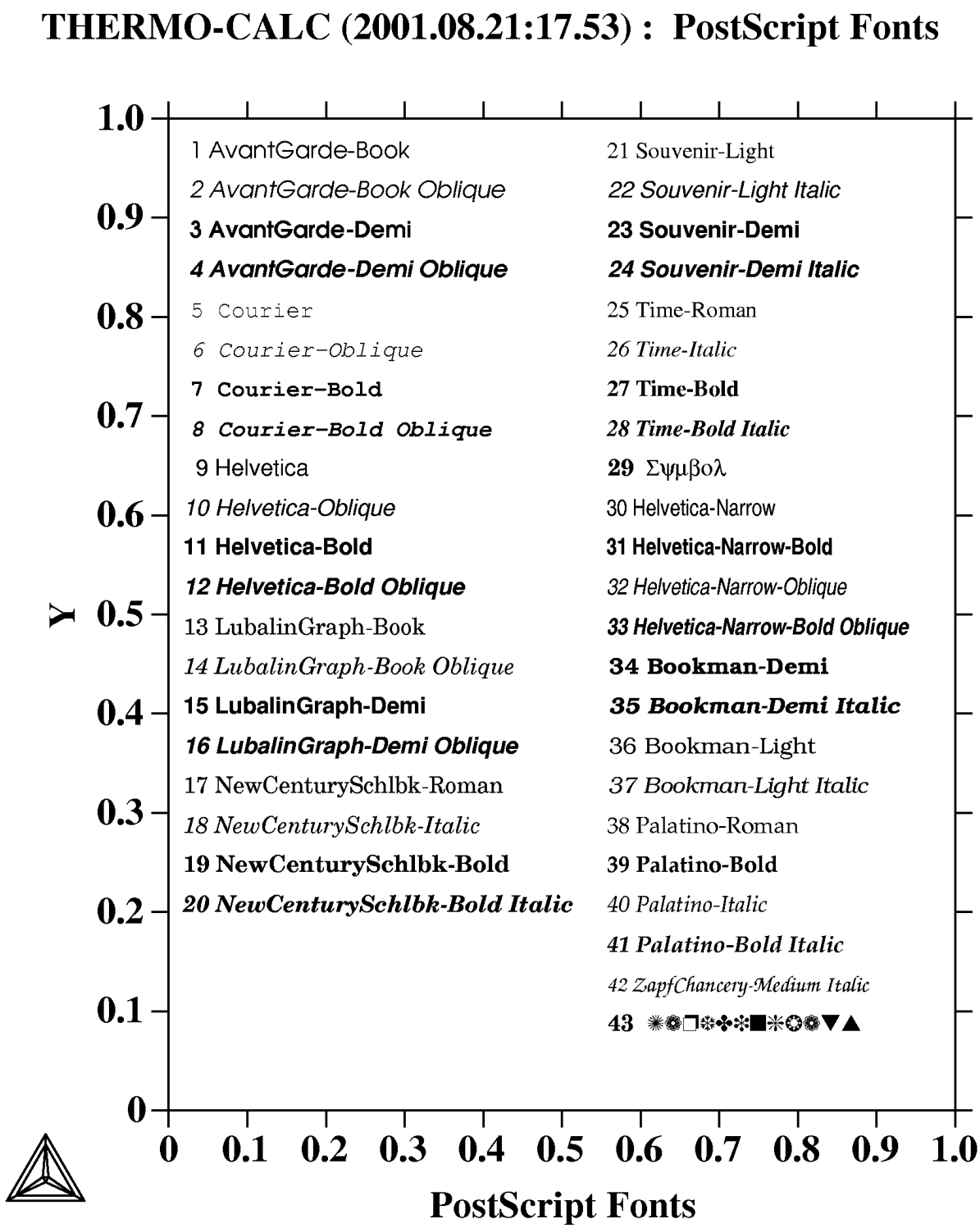


Figure 4. PostScript Outputs of the Available PostScript Fonts

Table 2a. Current ENCODING Vectors used by PostScript Fonts 1-28 & 30-42

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	”	#	$	%	&	’
\05x	(	)	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	_
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	
\20x	™	‰	•	*f*	†	‡	Ł	ł
\21x	{	[	}	]	\	|	Œ	œ
\22x	ı	`	´	ˆ	˜	¯	˘	·
\23x	¨		°	¸		˝	˛	ˇ
\24x	Ÿ	¡	¢	£	¤	¥	¦	§
\25x	…	©	ª	«	¬	-	®	/
\26x	°	±	‹	›	Ž	µ	¶	·
\27x	Š	š	º	»	ž	□	□	¿
\30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
\31x	È	É	Ê	Ë	Ì	Í	Î	Ï
\32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×
\33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
\34x	à	á	â	ã	ä	å	æ	ç
\35x	è	é	ê	ë	ì	í	î	ï
\36x	ð	ñ	ò	ó	ô	õ	ö	÷
\37x	ø	ù	ú	û	ü	ý	þ	ÿ

No printable characters

Only on LN03R not the QMS-PS810

Table 2b. Current ENCODING Vectors used by PostScript Font 29

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		∫	∀	#	∃	%	&	϶
\05x	(	)	*	+	,	−	.	/
\06x	∂	≈	⇔	⇐	⇒	∝	*f*	√
\07x	∞	∇	:	;	≤	≠	≥	?
\10x	≅	←	→	↔	Δ	·	Φ	Γ
\11x	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
\12x	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω
\13x	Ξ	Ψ	≡	[	∴	]	⊥	_
\14x	‾	α	β	χ	δ	ε	ϕ	γ
\15x	η	ι	φ	κ	λ	μ	ν	ο
\16x	π	θ	ρ	σ	τ	υ	ϖ	ω
\17x	ξ	ψ	ζ	{	|	}	~	
\20x								
\21x								
\22x								
\23x								
\24x		ϒ	′	≤	⁄	∞	*f*	♣
\25x	♦	♥	♠	↔	←	↑	→	↓
\26x	°	±	″	≥	×	∝	∂	•
\27x	÷	≠	≡	≈	…	|	—	↵
\30x	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩
\31x	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
\32x	∠	∇	®	©	™	∏	√	·
\33x	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
\34x	◊	⟨	®	©	™	Σ	⎛	⎪
\35x	⎝	⎡	⎪	⎣	⎛	⎧	⎝	⎪
\36x		⟩	⌠	⌡	⎪	⌡	⎠	⎪
\37x	⌡	⎤	⎪	⌋	⎠	⎩	⌡	

Table 2c. Current ENCODING Vectors used by PostScript Font 43

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x								
\05x								
\06x								
\07x								
\10x								
\11x								
\12x								
\13x								
\14x								
\15x								
\16x								
\17x								
\20x								
\21x								
\22x								
\23x								
\24x								
\25x								
\26x								
\27x								
\30x								
\31x								
\32x								
\33x								
\34x								
\35x								
\36x								
\37x								

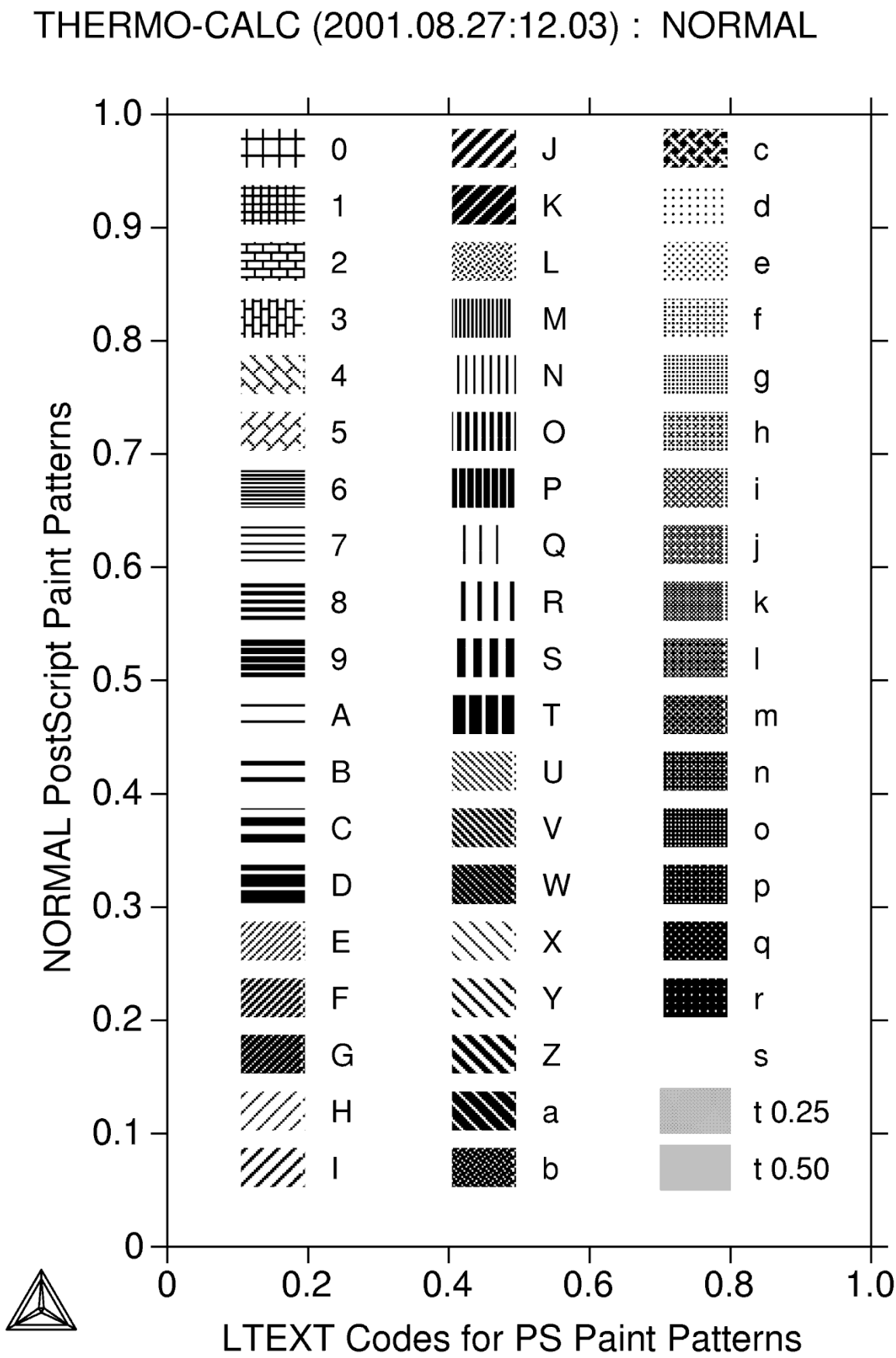


Figure 5a. **Current ENCODING PostScript Paint Patterns (NORMAL Video Status)**

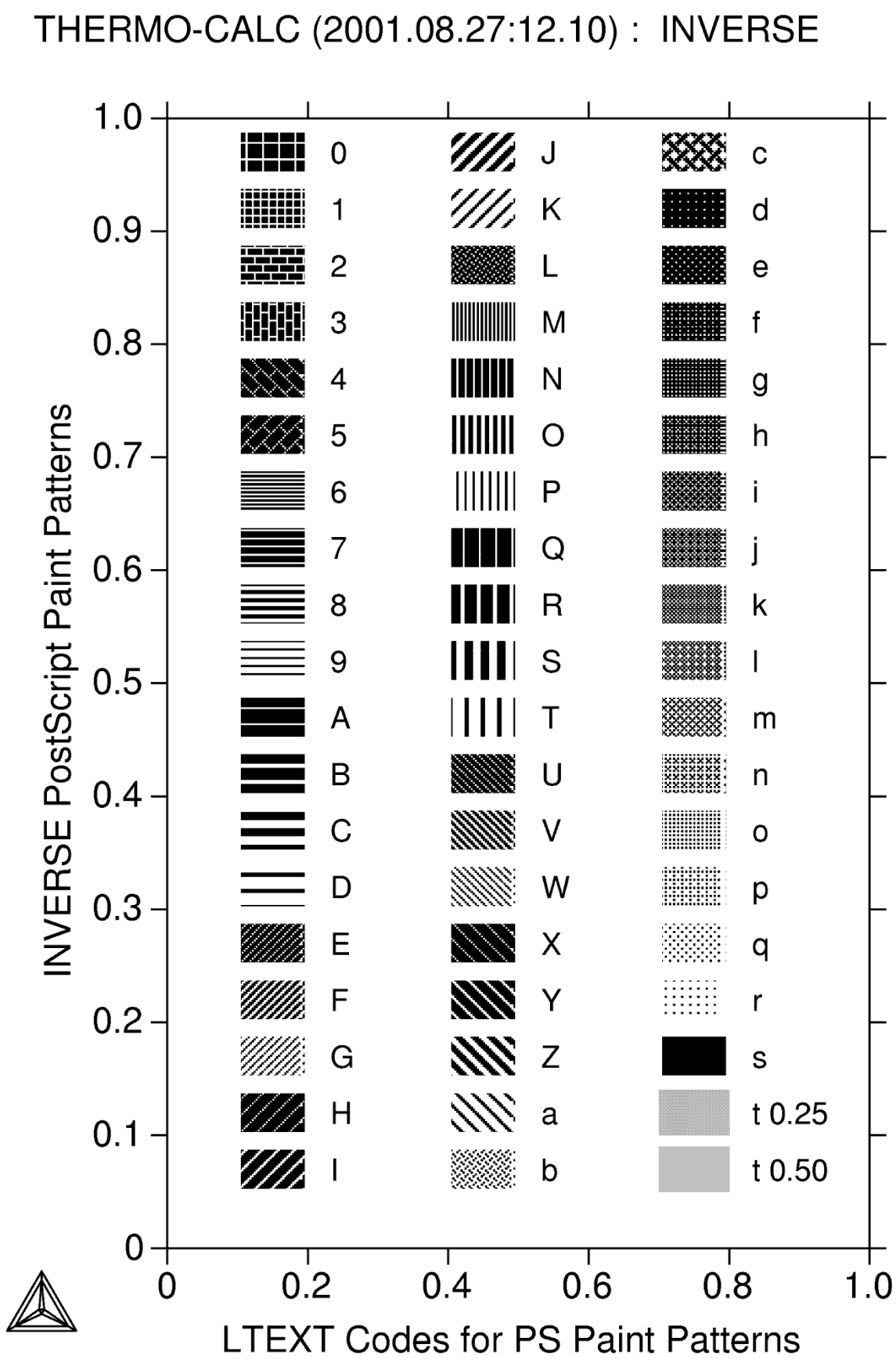


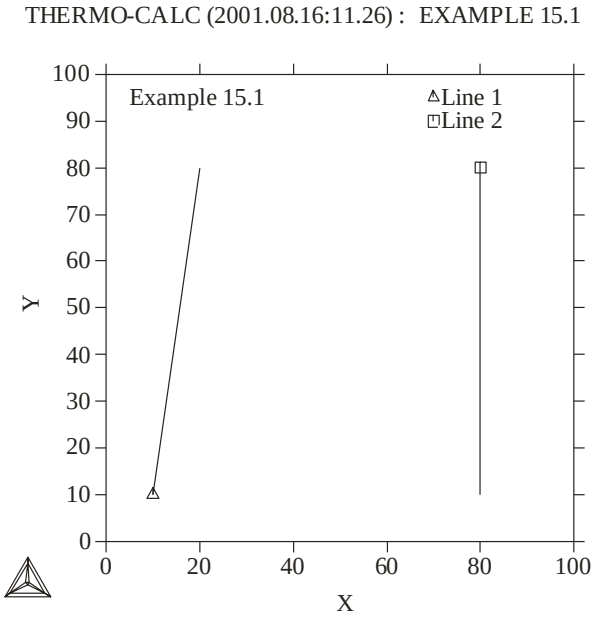
Figure 5b. **Current ENCODING PostScript Paint Patterns (INVERSE Video Status)**

(This page is intended to be empty)

6 Examples of DATAPLOT Files and Their Resulting Outputs

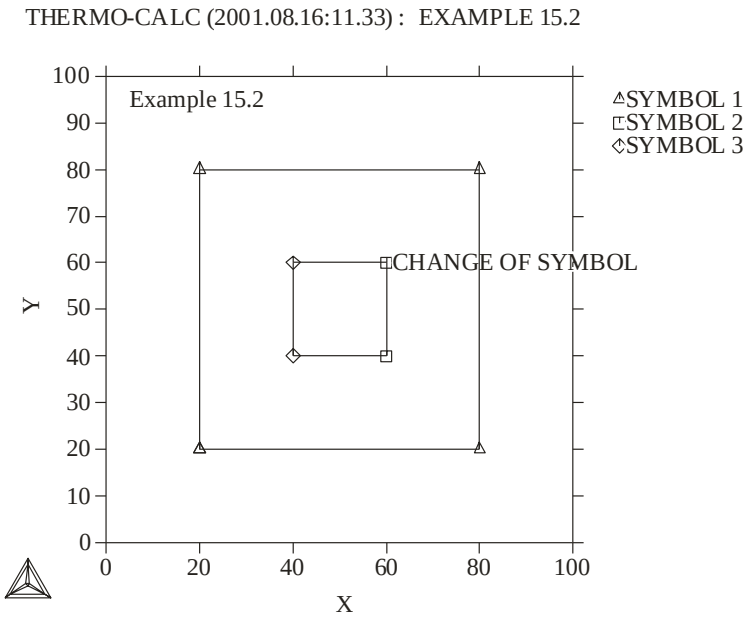
6.1 Example 1 – Draw Lines and Symbols

```
$DATAPLOT Example 1

PROLOG 1 EXAMPLE 1  0<X<100, 0<Y<100
 XSCALE        0.00000       100
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  EXAMPLE 1
 XTEXT  X
 YTEXT  Y

DATASET 1 Two lines started with two symbols
ATTRIBUTE CENTER
0.05  0.95 N'Example 1
0.7   0.95 NS'Line 1
0.7   0.90 NS2'Line 2
10    10  S1
20    80  D
80    80  S2
80    10  D
50    60
```



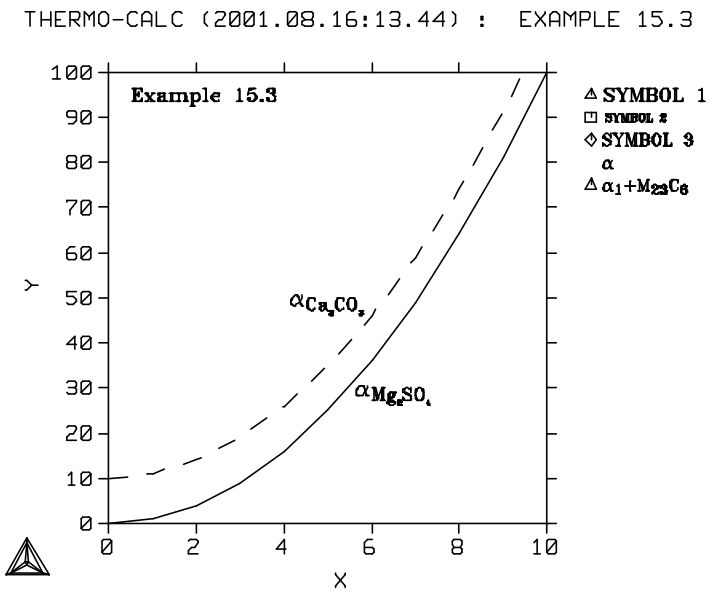
6.2 Example 2 – Draw Polygons and Symbols

```
$DATAPLOT Example 1

PROLOG 2 EXAMPLE 2  0<X<100, 0<Y<100
 XSCALE        0.00000       100
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  EXAMPLE 2
 XTEXT  X
 YTEXT  Y

DATASET 2 Two ploygons with three types of symbols
ATTRIBUTE CENTER
CLIP OFF
0.05  0.95 N'Example 2
1.1   0.95 NS1'SYMBOL 1
1.1   0.90 NS2'SYMBOL 2
1.1   0.85 NS3'SYMBOL 3
BLOCK X=C1; Y=C2; GOC=C3,DSWA
40    40  M
40    60
60    60  S2'CHANGE OF SYMBOL
60    40
40    40  S0
BLOCKEND
BLOCK X=C1*100; Y=C2*100; GOC=C3,DSWA
0.2   0.2 MS1
0.2   0.8
0.8   0.8
0.8   0.2
0.2   0.2
BLOCKEND
```



6.3 Example 3 – Using String and Various Line Types

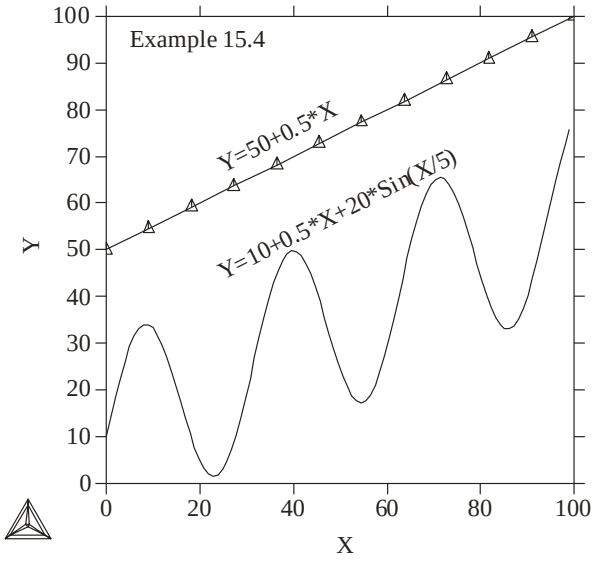
```
$DATAPLOT Example 3
PROLOG 3 EXAMPLE 3  0<X<10, 0<Y<100
 XSCALE        0.00000       10
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  EXAMPLE 3
 XTEXT  X
 YTEXT  Y
DATASET 3 Draw curvers; plot formatted texts and symbols
$Define some strings:
STRING BCC ^Ga^F0
STRING BCC1 ^Ga^F0^D0^S8,1^S0^U0
STRING M23C6 M^D0^S8,23^S0^U0C^D0^S8,6^S0^U0
STRING ACA2CO3 ^Ga^F0^D0^S8Ca^D0^S4,2^S0^U0^S8CO^D0^S4,3^S0^U0
STRING AMG2SO4 ^Ga^F0^D0^S8Mg^D0^S4,2^S0^U0^S8SO^D0^S4,4^S0^U0
$ Note: if as PostScript output:
$STRING BCC !a
$STRING BCC1 !a^do1$
$STRING M23C6 M^do23$C^do6$
$STRING ACA2CO3 !a^doCa^do2$^doCO^do3$
$STRING AMG2SO4 !a^doMg^do2$^doSO^do4$
ATTRIBUTE CENTER
CLIP OFF
FONT 2
0.05  0.95 N'Example 3
1.1   0.95 NS1'SYMBOL 1
CHARSIZE 0.2
1.1   0.90 NS2' SYMBOL 2
CHARSIZE 0.3
1.1   0.85 NS3'SYMBOL 3
1.1   0.80 N' ~BCC
1.1   0.75 NS1'~BCC1+~M23C6
SYMBOLSIZE 0.4
CHARSIZE 0.4
0.41  0.50 N'~ACA2CO3
0.56  0.30 N'~AMG2SO4
CLIP ON
LINETYPE 1
BLOCK X=C1; Y=C1*C1; GOC=C2, DWA
0 M
1
2
3
4
5
6
7
8
9
10
BLOCKEND
LINETYPE 2
BLOCK X=C1; Y=C1*C1+10; GOC=C2, DWA
0 M
1
2
3
4
5
6
7
8
9
10
BLOCKEND
```



6.4 Example 4 – Draw Curves Defined by Functions

```
$DATAPLOT Example 4

PROLOG 4 EXAMPLE 4  0<X<100, 0<Y<100
 XSCALE        0.00000       100
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH         11.5000
 YLENGTH         11.5000
 TITLE  EXAMPLE 4
 XTEXT  X
 YTEXT  Y

DATASET 4 Plot two functions as lines:
ATTRIBUTE CENTER
0.05  0.95 N'Example 4
$ Draw two lines defined by FUNCTIONS:
FUNCTION Y=10+0.5*X+20*Sin(X/5); 0 100 100; DWA;
FUNCTION Y=50+0.5*X; 0 100 10; DS1WA;
$ Write funtions beside the lines:
$    Note the real rotation angle (27 degree) can be seen
$      only on the PostScript hardcopy!
0.25  0.68 N'^RO27Y=50+0.5*X
0.25  0.45 N'^RO27Y=10+0.5*X+20*Sin(X/5)
```

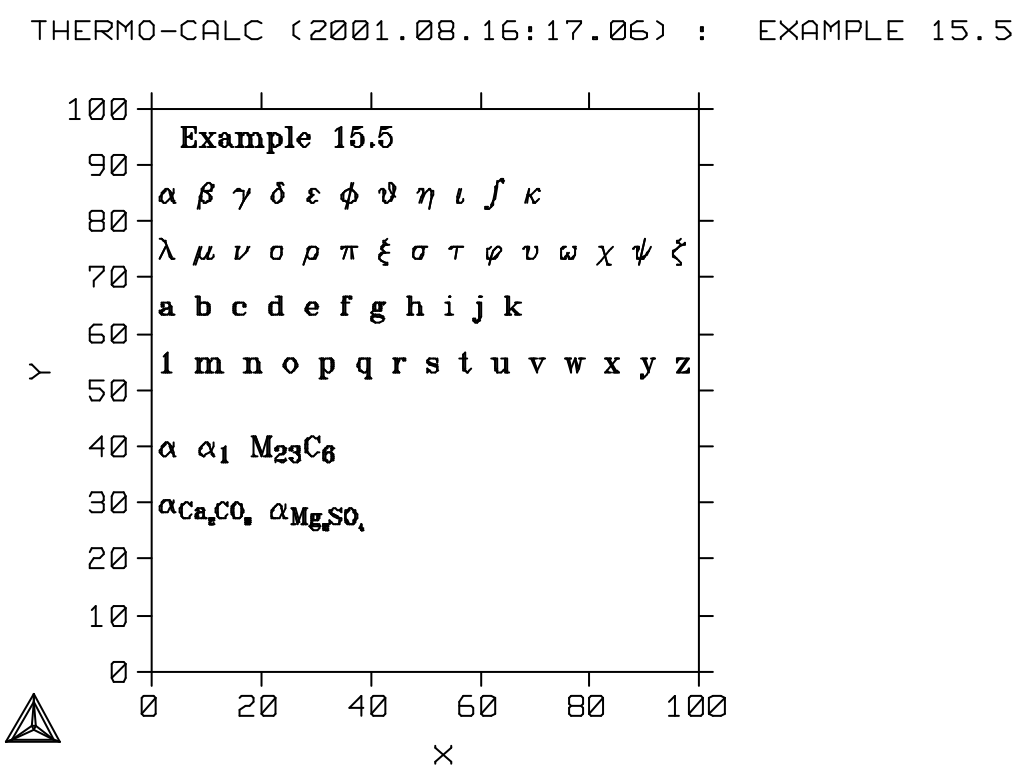
THERMO-CALC (2001.08.16:16.32) : EXAMPLE 15.4



(Note that this is a PostScript copy of the output. On Thermo-Calc Graph window and resulting EMF file, the angle will not appear correctly.)

6.5 Example 5 – Use Included Files for Predefined Symbols

```
$DATAPLOT Example 5

PROLOG 5 EXAMPLE 5  0<X<100, 0<Y<100
 XSCALE        0.00000       100
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  EXAMPLE 5
 XTEXT  X
 YTEXT  Y

DATASET 5 Write characters in various fonts, and defined symbols
FONT 2
INCLUDE <DATAPLOT-EXAMPLE-PATH>INCLUDE.EXP
ATTRIBUTE CENTER
0.05  0.95 N'Example 5
CLIP OFF
0.01  0.85 N'~TEST1A
0.01  0.75 N'~TEST1B
0.01  0.65 N'~TEST2A
0.01  0.55 N'~TEST2B
0.01  0.40 N'~BCC ~BCC1 ~M23C6
0.01  0.30 N'~ACA2CO3 ~AMG2SO4
```



The INCLUDE.EXP file has the following content:

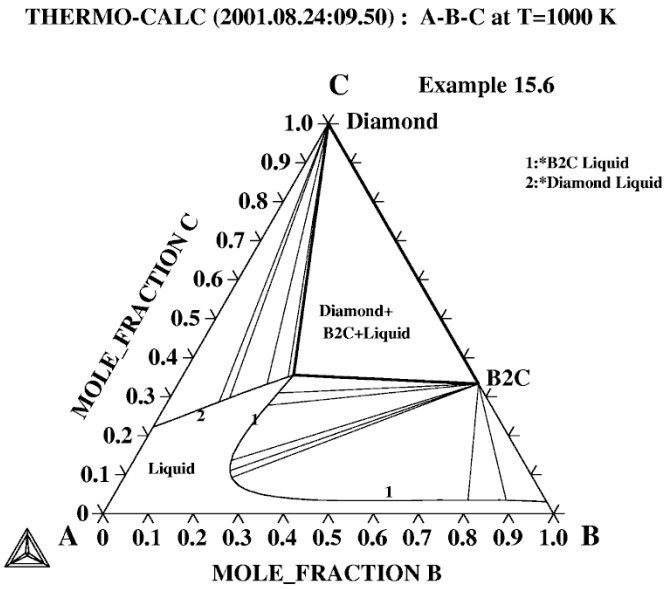
```
------------------------------------------------
STRING TEST1A ^Ga b c d e f g h i j k^F0
STRING TEST1B ^Gl m n o p q r s t u v w x y z^F0
STRING TEST2A   a b c d e f g h i j k
STRING TEST2B   1 m n o p q r s t u v w x y z
STRING BCC ^Ga^F0
STRING BCC1 ^Ga^F0^D0^S8,1^S0^U0
```

```
STRING BCC2 ^Ga^F0^D0^S8,2^S0^U0
STRING FCC ^Gc^F0
STRING FCC1 ^Gc^F0^D0^S8,1^S0^U0
STRING FCC2 ^Gc^F0^D0^S8,2^S0^U0
STRING L Liquid
STRING SIGMA ^Gs^F0
STRING MU ^Gm^F0
STRING LAVES ^Gl^F0
STRING CHI ^Gx^F0
STRING KSI ^Gr^F0
STRING MCETA ^S12^Gh^F0^S0
STRING M2C  M^D0^S8,2^S0^U0C
STRING M2CT M^D0^S10,2^S0^U0C
STRING M6C  M^D0^S8,6^S0^U0C
STRING M23C6 M^D0^S8,23^S0^U0C^D0^S8,6^S0^U0
STRING M7C3 M^D0^S8,7^S0^U0C^D0^S8,3^S0^U0
STRING M3C2 M^D0^S8,3^S0^U0C^D0^S8,2^S0^U0
STRING MC1-X MC^D0^S8,1-x^S0^U0
STRING XC x^D0^S7,C^S0^U0
STRING XFE x^D0^S7,Fe^S0^U0
STRING XMO x^D0^S7,Mn^S0^U0
STRING XW x^D0^S7,W^S0^U0
STRING UW ~XW/(~XMO+~XW)
STRING ac a^D0^S7,C^S0^U0
STRING ACC a^D0^S7,C^S0^U0
STRING ACA2CO3 ^Ga^F0^D0^S8Ca^D0^S4,2^S0^U0^S8CO^D0^S4,3^S0^U0
STRING AMG2SO4 ^Ga^F0^D0^S8Mg^D0^S4,2^S0^U0^S8SO^D0^S4,4^S0^U0

$ Note: if as PostScript output:
$STRING TEST1A !a !b !c !d !e !f !g !h !i !j !k^fo27
$STRING TEST1B !l !m !n !o !p !q !r !s !t !u !v !w !x !y !z^fo27
$STRING TEST2A   a b c d e f g h i j k
$STRING TEST2B   1 m n o p q r s t u v w x y z
$STRING BCC !a
$STRING BCC1 !a^do1$
$STRING BCC2 !a^do2$
$STRING FCC !c
$STRING FCC1 !c^do1$
$STRING FCC2 !c^do2$
$STRING L Liquid
$STRING SIGMA !s
$STRING MU !m
$STRING LAVES !l
$STRING CHI !x
$STRING KSI !r
$STRING MCETA !h
$STRING M2C  M^do2$C
$STRING M2CT M^do2$C
$STRING M6C  M^do6$C
$STRING M23C6 M^do23$C^do6$
$STRING M7C3 M^do7$C^do3$
$STRING M3C2 M^do3$C^do2$
$STRING MC1-X MC^do1-x$
$STRING XC x^doC$
$STRING XFE x^doFe$
$STRING XMO x^doMo$
$STRING XW x^doW$
$STRING UW ~XW/(~XMO+~XW)
$STRING ac a^doC$
$STRING ACC a^doC$
$STRING ACA2CO3 !a^doCa^do2$^doCO^do3$
$STRING AMG2SO4 !a^doMg^do2$^doSO^do4$
```

6.6 Example 6 – Plot Triangular Diagrams for Ternary Systems

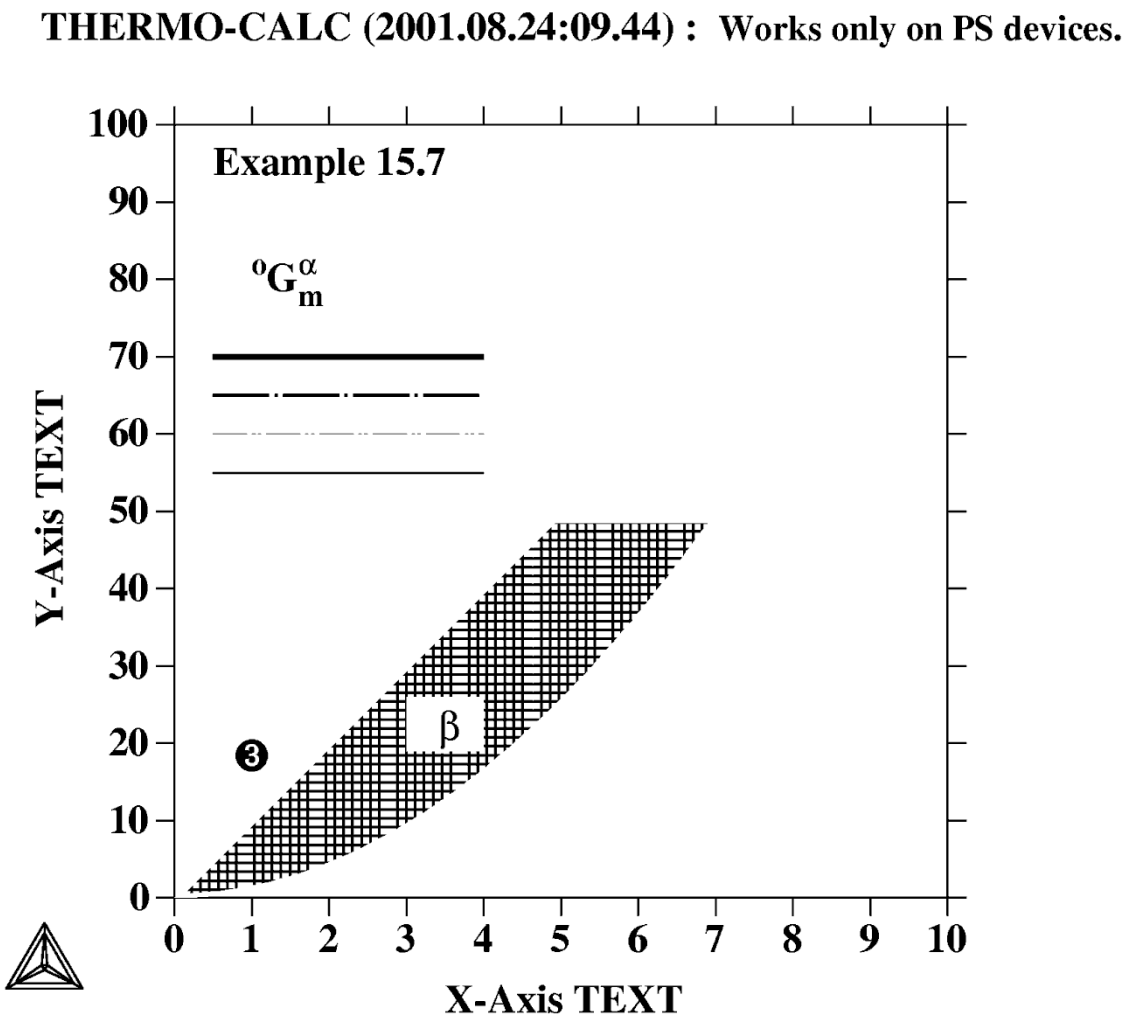
```
$DATAPLOT Example 6

PROLOG 6 EXAMPLE 6  0<X<0.969224, 0<Y<1.00000
 XSCALE        0.00000          0.969224
 YSCALE       0.307492E-01       1.00000
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  A-B-C at T=1000 K
 XTEXT  MOLE_FRACTION B
 YTEXT  MOLE_FRACTION C
 DIAGRAM_TYPE TRIANGULAR YES YES

DATASET 6 Plot a ternary phase diagram
CLIP OFF
0.70  0.95 N'Example 6
0.85  0.30 N'B2C
0.54  0.87 N'Diamond
CHARSIZE 0.25
  1.4E+01  1.10E+01 MVA'1:*B2C Liquid
  1.4E+01  1.05E+01 MVA'2:*Diamond Liquid
  0.10  0.10 N'Liquid
  0.48  0.45 N'Diamond+
  0.48  0.40 N' B2C+Liquid
  5.80E-01  5.40E-02 MWA'  1
  1.90E-01  2.40E-01 MWA'  1
  0.65E-01  2.50E-01 MWA'  2
CHARSIZE 0.45
-0.10  -0.05 N'A
 1.06  -0.05 N'B
 0.50   0.95 N'C
CHARSIZE 0.35
$$ Calculated A-B-C Phase Equilibrium Data:
 $ PHASE REGION FOR:
 $F0 LIQUID
 $E DIAMOND_A4
 $F0 B2C
$ INVARIANT EQUILIBRIUM
COLOR   2
BLOCK X=C1; Y=C2;       GOC=C3,WAD;
   2.4555855989E-01   3.5568857193E-01   M
   0.0000000000E+00   9.9999523163E-01
   2.4555855989E-01   3.5568857193E-01   M
   6.6666668653E-01   3.3333334327E-01
   0.0000000000E+00   9.9999523163E-01   M
   6.6666668653E-01   3.3333334327E-01
COLOR   1
BLOCKEND
 $ PHASE REGION FOR:
 $E LIQUID
 $F0 B2C
BLOCK X=C1; Y=C2;       GOC=C3,WAD;
$ PLOTTED COLUMNS ARE : X(LIQUID,B) and X(LIQUID,C)
   2.2030337155E-01   1.2340000272E-01   M
   2.2632879019E-01   1.1058768630E-01
   2.3371633887E-01   9.9345825613E-02
   2.4253317714E-01   8.9345827699E-02
   2.6429468393E-01   7.2744041681E-02
   2.8429466486E-01   6.2814079225E-02
   2.9617273808E-01   5.8319382370E-02
   3.2811737061E-01   4.9470417202E-02
   3.6353862286E-01   4.3130427599E-02
   3.9895987511E-01   3.8979098201E-02
   4.5209178329E-01   3.5266116261E-02
   5.2293431759E-01   3.3152002841E-02
   6.1148744822E-01   3.3077053726E-02
   6.4690870047E-01   3.3490389585E-02
   6.8232995272E-01   3.4017231315E-02
   7.3546189070E-01   3.4814555198E-02
   7.5317251682E-01   3.5033416003E-02
   8.0630439520E-01   3.5373892635E-02
```

```
   8.5943627357E-01   3.4983776510E-02
   9.1256815195E-01   3.3575300127E-02
   9.6747112274E-01   3.0857827514E-02
   9.6922445297E-01   3.0749246478E-02
   2.2030337155E-01   1.2340000272E-01   M
   2.1294665337E-01   1.5308913589E-01
   2.1171525121E-01   1.8851040304E-01
   2.1532440186E-01   2.2393165529E-01
   2.2180187702E-01   2.5935292244E-01
   2.2992117703E-01   2.9477417469E-01
   2.3888295889E-01   3.3019542694E-01
   2.4555855989E-01   3.5568857193E-01
$ PLOTTED COLUMNS ARE : X(B2C,B) and X(B2C,C)
   6.6666668653E-01   3.3333334327E-01   M
   6.6666668653E-01   3.3333334327E-01
$ TIELINES
COLOR   3
   6.6666668653E-01   3.3333334327E-01   M
   8.7775242329E-01   3.4625384957E-02
   6.6666668653E-01   3.3333334327E-01   M
   7.9250496626E-01   3.5342670977E-02
   6.6666668653E-01   3.3333334327E-01   M
   2.4555824697E-01   3.5568737984E-01
   6.6666668653E-01   3.3333334327E-01   M
   2.3944084346E-01   9.2542596161E-02
   6.6666668653E-01   3.3333334327E-01   M
   2.3359020054E-01   3.0954307318E-01
   6.6666668653E-01   3.3333334327E-01   M
   2.2585247457E-01   2.7766343951E-01
   6.6666668653E-01   3.3333334327E-01   M
   2.1618695557E-01   1.3621240854E-01
   2.2632879019E-01   1.1058768630E-01   M
   6.6666668653E-01   3.3333334327E-01
COLOR   1
BLOCKEND
 $ PHASE REGION FOR:
 $F0 DIAMOND_A4
 $E B2C
BLOCK X=C1; Y=C2;       GOC=C3,WAD;
$ PLOTTED COLUMNS ARE : X(DIAMOND_A4,B) and X(DIAMOND_A4,C)
   0.0000000000E+00   9.9999523163E-01   M
   0.0000000000E+00   9.9999976158E-01
BLOCKEND
 $ PHASE REGION FOR:
 $E LIQUID
 $F0 DIAMOND_A4
BLOCK X=C1; Y=C2;       GOC=C3,WAD;
$ PLOTTED COLUMNS ARE : X(LIQUID,B) and X(LIQUID,C)
  2.45558E-01  3.55688E-01  M
  2.02635E-01  3.34830E-01
  1.62439E-01  3.13753E-01
  1.22439E-01  2.91531E-01
  8.24390E-02  2.68542E-01
  4.24390E-02  2.45480E-01
  2.43905E-03  2.23138E-01
  2.49999E-07  2.21816E-01
$ TIELINES
COLOR   3
  0.00000E+00  9.99995E-01  M
  2.35291E-01  3.50859E-01
  0.00000E+00  9.99994E-01  M
  1.98265E-01  3.32609E-01
  0.00000E+00  9.99994E-01  M
  1.32400E-01  2.97160E-01
  0.00000E+00  9.99993E-01  M
  1.14399E-01  2.86953E-01
BLOCKEND
```



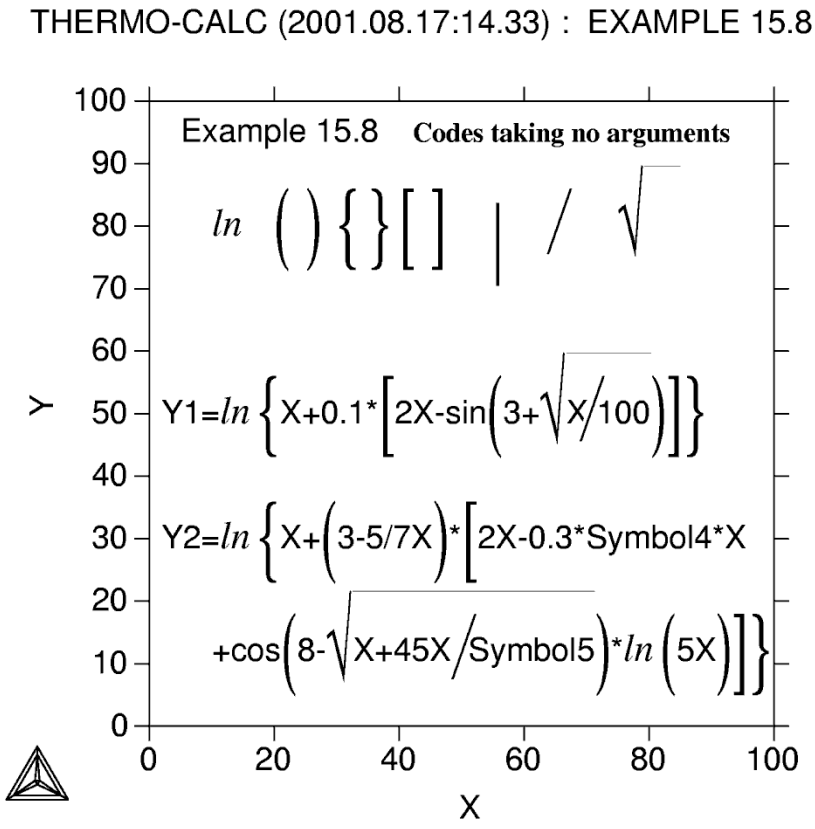
6.7 Example 7 – PostScript Characters/Symbols/Patterns/Lines

```
$DATAPLOT Example 7

PROLOG 7 EXAMPLE 7  0<X<10, 0<Y<100
 XSCALE        0.00000       10
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  ^fs16Works only on PS devices.^fs8
 XTEXT  X-Axis TEXT
 YTEXT  Y-Axis TEXT

DATASET 7 PostScript Output
ATTRIBUTE CENTER
0.05  0.95 N'Example 7

$Use LTEXT formatting codes to write a complex expression
$  at the coordinates (0.1, 0.8):
0.1  0.8 MNA'^upo#G^di!a#m#

$Paint a selected area (with Pattern 1) bound
$  by a square function and two line segments:
0    0   MWA
PCFUNCTION Y=X**2; 0 7 100; DWA
7   49   DWA
4.9 49   DWA
0    0   DWA
PAINT 1

$Put a text (beta) in a white box (Pattern s)
$  on the painted field:
3   19   MWA
4   19   DWA
4   26   DWA
3   26   DWA
3   19   DWA
PAINT s OPAQUE
3.28 22   MWA' !b

$Use ZapfDingbats font from PostScript to plot additional symbols:
$  The octal value 314 was selected from the ENCODING vectors.
0.1  0.20 MNA' ^ps xm ym 80 (\314) putsymbol

$Set some different linetypes using PostScript Codes:
$  Draw a very thick line
1.1  0.0  MNA' ^ps 10.0 slw
0.5  70   MWA
4    70   DWA
$  Draw a thick dash-dotted line
1.1  0.0  MNA' ^ps [100 10 5 10] 0 sd 5 slw
0.5  65   MWA
4    65   DWA
$  Draw a thin dash-double-dotted line
1.1  0.0  MNA' ^ps [60 10 5 5 5 10] 0 sd 1.0 slw
0.5  60   MWA
4    60   DWA
$  Reset to normal-thick-solid line
1.1  0.0  MNA' ^ps [] 0 sd 3.0 slw
0.5  55   MWA
4    55   DWA
```


THERMO-CALC (2001.08.24:09.44) : Works only on PS devices.
Example 15.7
$^{o}G_{m}^{\alpha}$
β
❸
Y-Axis TEXT
X-Axis TEXT
0
10
20
30
40
50
60
70
80
90
100
0
1
2
3
4
5
6
7
8
9
10

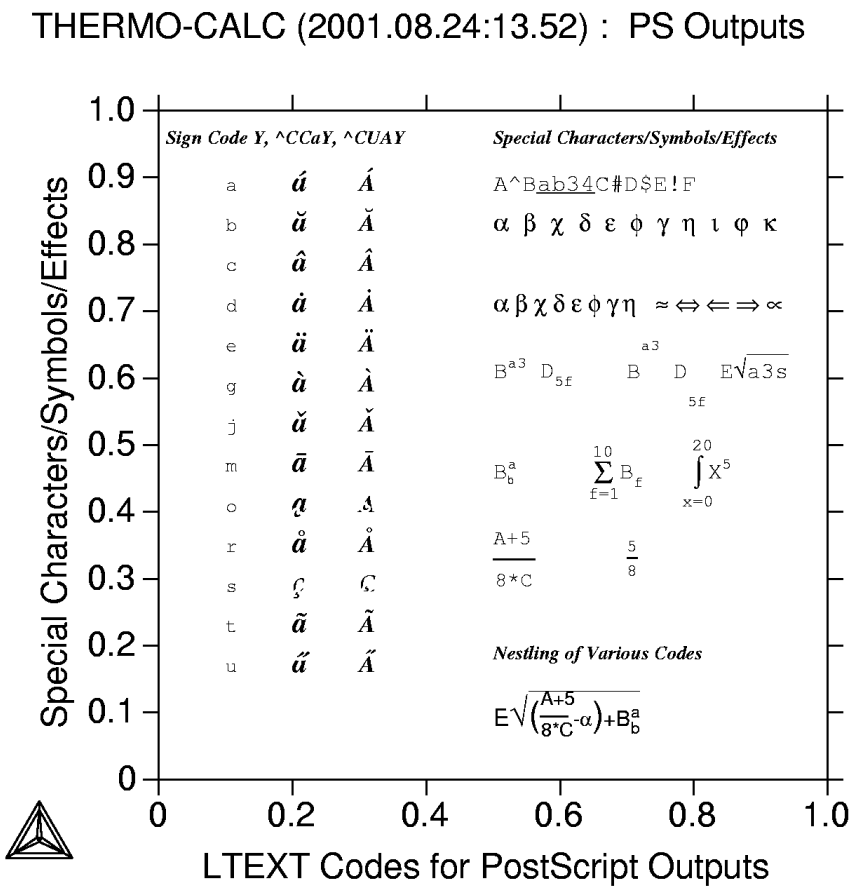
6.8 Example 8 – LTEXT Codes Taking No Arguments

```
$DATAPLOT Example 8

PROLOG 8 EXAMPLE 8  0<X<100, 0<Y<100
 XSCALE        0.00000       100
 YSCALE        0.00000       100
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  EXAMPLE 8
 XTEXT  X
 YTEXT  Y

DATASET 8 Various LTEXT Formatting Codes:
0.05  0.95 N'Example 8   ^fo27^fs15 Codes taking no arguments
$ Codes taking no arguments:
$0.05  0.80 MNA'^hx
0.1   0.80 MNA'^ln
0.2   0.80 MNA'^lb
0.25  0.80 MNA'^eb
0.3   0.80 MNA'^LC
0.35  0.80 MNA'^EC
0.4   0.80 MNA'^LS
0.45  0.80 MNA'^ES
0.55  0.80 MNA'^VB
0.65  0.80 MNA'^SL
0.75  0.80 MNA'^QS
0.85  0.80 MNA'^QE
$ Write expressions using codes taking on arguments:
0.02 0.50 MNA'Y1=^ln^lcX+0.1*^ls2X-sin^lb3+^qsX^sl100^qe^eb^es^ec
0.02 0.30 MNA'Y2=^ln^lcX+^lb3-5/7X^eb*^ls2X-0.3*Symbol4*X
0.10 0.12 MNA'+cos^lb8-^qsX+45X ^SLSymbol5^qe^eb*^ln^lb5X^eb^es^ec
```



6.9 Example 9 – LTEXT Codes Taking Various Types of Arguments

```
$DATAPLOT PostScript Codes

PROLOG 9 PostScript Codes  0<X<1.0, 0<Y<1.0
 XSCALE        0.0       1.0
 YSCALE        0.0       1.0
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  PS Outputs
 XTEXT  LTEXT Codes for PostScript Outputs
 YTEXT  Special Characters/Symbols/Effects

DATASET 9 Various LTEXT Codes for PS characters/symbols/effects:
$ PostScript Codes:
CHARSIZE 0.2
0.01  0.95 MNA'^FO28Sign Code Y, ^^CCaY, ^^CUAY
0.5   0.95 MNA'^FO28Special Characters/Symbols/Effects
0.5   0.18 MNA'^FO28Nestling of Various Codes
0.1   0.88 MNA'^FO05a
0.1   0.82 MNA'^FO05b
0.1   0.76 MNA'^FO05c
0.1   0.70 MNA'^FO05d
0.1   0.64 MNA'^FO05e
0.1   0.58 MNA'^FO05g
0.1   0.52 MNA'^FO05j
0.1   0.46 MNA'^FO05m
0.1   0.40 MNA'^FO05o
0.1   0.34 MNA'^FO05r
0.1   0.28 MNA'^FO05s
0.1   0.22 MNA'^FO05t
0.1   0.16 MNA'^FO05u
CHARSIZE 0.3
0.2   0.88 MNA'^FO28^CCaa
0.2   0.82 MNA'^FO28^CCab
0.2   0.76 MNA'^FO28^CCac
0.2   0.70 MNA'^FO28^CCad
0.2   0.64 MNA'^FO28^CCae
0.2   0.58 MNA'^FO28^CCag
0.2   0.52 MNA'^FO28^CCaj
0.2   0.46 MNA'^FO28^CCam
0.2   0.40 MNA'^FO28^CCao
0.2   0.34 MNA'^FO28^CCar
0.2   0.28 MNA'^FO40^CCcs
0.2   0.22 MNA'^FO28^CCat
0.2   0.16 MNA'^FO28^CCau
CHARSIZE 0.25
0.3   0.88 MNA'^FO28^CUAa
0.3   0.82 MNA'^FO28^CUAb
0.3   0.76 MNA'^FO28^CUAc
0.3   0.70 MNA'^FO28^CUAd
0.3   0.64 MNA'^FO28^CUAe
0.3   0.58 MNA'^FO28^CUAg
0.3   0.52 MNA'^FO28^CUAj
0.3   0.46 MNA'^FO28^CUAm
0.3   0.40 MNA'^FO28^CUAo
0.3   0.34 MNA'^FO28^CUAr
0.3   0.28 MNA'^FO40^CUCs
0.3   0.22 MNA'^FO28^CUAt
0.3   0.16 MNA'^FO28^CUAu
```




```
0.5  0.88 MNA'^FO05A^^B^_ab34^_C^#D^$E^!F
0.5  0.82 MNA'!a !b !c !d !e !f !g !h !i !j !k
$0.5  0.76 MNA'^FO05 CTRL-H5 CTRL-O234
0.5  0.70 MNA'^GRa b c d e f g h   1 2 3 4 5$
0.5  0.60 MNA'^FO05B^UP^FS08a3^FS12$ D^DO^FS085f^FS12$
0.7  0.60 MNA'^FO05B^BI^FS08a3^FS12$ D^BD^FS085f^FS12$ E^SQa3s$
0.5  0.45 MNA'B^DI^FS08a$b$^FS12
0.65 0.45 MNA'^SU10#f=1#B^DO^FS08f$^FS12
0.80 0.45 MNA'^IN20#x=0#X^UP^FS085$^FS12
0.5  0.32 MNA'^KVA+5#8*C#
0.7  0.32 MNA'^SK5$8$
0.5  0.08 MNA'E^FS18^SQ(^SK^FS10A+5#8*C#^FS10 -!a^FS18)^FS11+B^DIa#b#$
```

6.10 Example 10 – Color Codes and Their Color Effects

```
$DATAPLOT Color Codes & Color Effects

PROLOG 10 Color Codes  0<X<1.0, 0<Y<1.0
 XSCALE        0.0       1.0
 YSCALE        0.0       1.0
 XTYPE LINEAR
 YTYPE LINEAR
 XLENGTH        11.5000
 YLENGTH        11.5000
 TITLE  Color Outputs
 XTEXT  Color Codes
 YTEXT  Color Effects

DATASET 10 Various Color Codes for Color Outputs:
CHARSIZE 0.3
LINETYPE 1
BLOCK X=C1; Y=C2;       GOC=C3,WAD;
0.05  0.95 MNA'Color Code: Color Effect
0.55  0.95 MNA'Color Code: Color Effect
CHARSIZE 0.25

COLOR  1
  0.05  0.88 MNA'1: Black
  CLIP ON
  0.22  0.88 M
  0.45  0.88
  CLIP OFF
COLOR  2
  0.05  0.82 MNA'2: Red
  CLIP ON
  0.22  0.82 M
  0.45  0.82
  CLIP OFF
COLOR  3
  0.05  0.76 MNA'3: Green
  CLIP ON
  0.22  0.76 M
  0.45  0.76
  CLIP OFF
COLOR  4
  0.05  0.70 MNA'4: Blue
  CLIP ON
  0.22  0.70 M
  0.45  0.70
  CLIP OFF
COLOR  5
  0.05  0.64 MNA'5: Yellow
  CLIP ON
  0.22  0.64 M
  0.45  0.64
  CLIP OFF
COLOR  6
  0.05  0.58 MNA'6: Magenta
  CLIP ON
  0.22  0.58 M
  0.45  0.58
  CLIP OFF
COLOR  7
  0.05  0.52 MNA'7: Cyan
  CLIP ON
  0.22  0.52 M
  0.45  0.52
  CLIP OFF
COLOR  8
  0.05  0.46 MNA'8: Purple
  CLIP ON
  0.22  0.46 M
  0.45  0.46
  CLIP OFF
COLOR  9
  0.05  0.40 MNA'9: Gold
  CLIP ON
  0.22  0.40 M
  0.45  0.40
  CLIP OFF
COLOR  10
  0.05  0.34 MNA'10: Turquoise
  CLIP ON
  0.22  0.34 M
  0.45  0.34
```

```
  CLIP OFF
COLOR  11
  0.05  0.28 MNA'11: Pink
  CLIP ON
  0.22  0.28 M
  0.45  0.28
  CLIP OFF
COLOR  12
  0.05  0.22 MNA'12: Gray
  CLIP ON
  0.22  0.22 M
  0.45  0.22
  CLIP OFF
COLOR  13
  0.05  0.16 MNA'13: Orangered
  CLIP ON
  0.22  0.16 M
  0.45  0.16
  CLIP OFF
COLOR  14
  0.05  0.10 MNA'14: Moroon
  CLIP ON
  0.22  0.10 M
  0.45  0.10
  CLIP OFF
COLOR  15
  0.05  0.04 MNA'15: Plum
  CLIP ON
  0.22  0.04 M
  0.45  0.04
  CLIP OFF
COLOR  16
  0.55  0.88 MNA'16: Seagreen
  CLIP ON
  0.72  0.88 M
  0.95  0.88
  CLIP OFF
COLOR  17
  0.55  0.82 MNA'17: Olivedrab
  CLIP ON
  0.72  0.82 M
  0.95  0.82
  CLIP OFF
COLOR  18
  0.55  0.76 MNA'18: Sienna
  CLIP ON
  0.72  0.76 M
  0.95  0.76
  CLIP OFF
COLOR  19
  0.55  0.70 MNA'19: Orange
  CLIP ON
  0.72  0.70 M
  0.95  0.70
  CLIP OFF
COLOR  20
  0.55  0.64 MNA'20: Coral
  CLIP ON
  0.72  0.64 M
  0.95  0.64
  CLIP OFF
COLOR  21
  0.55  0.58 MNA'21: UserDef
  CLIP ON
  0.72  0.58 M
  0.95  0.58
  CLIP OFF
COLOR  22
  0.55  0.52 MNA'22 = 1
  CLIP ON
  0.72  0.52 M
  0.95  0.52
  CLIP OFF
COLOR  23
  0.55  0.46 MNA'23 = 1
  CLIP ON
  0.72  0.46 M
  0.95  0.46
  CLIP OFF
... ...
BLOCKEND
```