

Manual

WOLF-RAINER HAMANN

with earlier support of Lars Koesterke and many more

Contents

1	Program installation and execution	4
1.1	Computers within the Astro I cluster at the Univ. Potsdam	4
1.2	WRplot installation on other linux systems	4
1.3	Position and size of the WRplot X-window	5
1.4	Input-line commands	6
2	The structure of a WRplot file	9
3	The KASDEF commands	11
3.1	Preface	11
3.1.1	Three types of KASDEF commands	12
3.2	Settings	12
3.3	Instructions	14
3.3.1	Output in the Terminal/command window and System calls	15
3.3.2	Variables	15
3.3.3	IF (conditional) constructions	18
3.3.4	DO-Loops and GOTO	18
3.3.5	Write into/read from a file	19
3.4	Plotcommands	19
3.4.1	Changing line thickness, color and font	19
3.4.2	Graphics: drawing actions	20
3.4.3	Hatched Areas	23
3.4.4	Text output	23
3.4.5	Text output options	24
3.4.6	Itemizing	26
3.4.7	Tables	26
3.4.8	Special characters, Umlauts etc.	27
3.4.9	L ^A T _E X formulae	27
3.4.10	Text attributes	28
3.4.11	Spectral line identifications	28
3.4.12	Include PostScript files/figures	30
3.4.13	Objects from L ^A T _E X code	31
4	The coordinate box	33
5	The plot data	35
5.1	Datasets	35
5.2	Headline and plot attributes	35
5.3	Data tables	36
5.4	COMMAND lines	39
5.5	COMMAND-Functions	45
6	Creating WRplot files with FORTRAN programs	48
7	Include plots into TeX documents	48
	Literature/References	50

A	Appendix: Examples for fonts	51
B	Appendix: Examples for colors	52
C	Appendix: The Plot symbols	53
	Index	56

1 Program installation and execution

1.1 Computers within the Astro I cluster at the Univ. Potsdam

At the Linux Workstation-Cluster `astro.physik.uni-potsdam.de`, the WRplot system is presently installed in the user directory of W.-R. Hamann: `/home/corona/wrh/linux-wrplot.dir`. This installation is cluster-wide.

C-shell users must define the proper path variable, and load a couple of aliases, by inserting the following two lines in his/her `.login` or `.cshrc` script:

```
setenv PATH_WRPLOT /home/corona/wrh/linux-wrplot.dir/wrplot.dir
source $PATH_WRPLOT/proc.dir/wrplot-aliases
```

Bash users have to add the following two lines to their `.profile` (for login shells) as well as to `.bashrc`:

```
PATH_WRPLOT=/home/corona/wrh/linux-wrplot.dir/wrplot.dir
source $PATH_WRPLOT/proc.dir/wrplot-aliases_bash
```

Alternatively, there is another installation with Helge's version of the program in `PATH_WRPLOT /home/crater/htodt/linux-wrplot.dir`.

1.2 WRplot installation on other linux systems

For an automatic installation, download the installation script

```
wget ftp://ftp.astro.physik.uni-potsdam.de/pub/wrhamann/wrplot-install.com
```

and execute

```
wrplot-install.com
```

WRplot will be installed in the user's home directory, i.e. root rights are not required. The script sets up the required environment and also provides the syntax highlighting files for KDE's editor *kate*, if the user has *kate* installed.

In case of problems with this skript, one may alternatively download (e.g. via anonymous ftp):

```
ftp ftp.astro.physik.uni-potsdam.de
cd pub/wrhamann
get wrplot.README
get wrplot.tgz
```

and follow the README instructions.

The executable of WRplot comes in three different versions:

wrplot.gnu.exe

This version is linked **statically** and should therefore run on **any** linux computer. This executable is my reference version and should be referred to when trouble-shooting. However, this version has only 32-bit numbers and is compiled without optimisation, i.e. it is slow.

wrplot.X11.exe

This version is optimized, and hence fast. Since the X11 window is progressively written, this version can be used for WRplot animations. However, depending on your Xserver abilities, it might be that the content of the X-window is not restored after it has been covered by another window, iconized, re-sized, or moved.

wrplot.motif.exe

In this version, the WRplot window is only filled when the plot is complete. Hence, this version cannot be used for animations. However, the X-window is more robust; it is restored after it had been covered, iconized, re-sized, or moved.

Trouble shooting: For the *motif* version of WRplot, the corresponding library must be installed on your system. If WRplot crashes with some error like:

```
libXm.so.4: no such file
```

this library is not installed yet at your system. On ubuntu linux, for instance, you need to type

```
sudo apt-get install libxm4
```

or

```
sudo apt-get install libmotif4
```

– or you switch to the X11 version (see below).

In your private installation, you can choose between these versions by setting a corresponding link from the generic wrplot.exe in \$PATH_WRPLOT/exe.dir. The default has been changed in January 2023 and is now the motif version:

```
ln -sf ../exe64.dir/wrplot.motif.exe wrplot.exe
```

An easy change to the X11 version (i.e. for wrplot-movies, or for debugging a WRplot script) is provided with the help of a newly introduced environment variable (only in 64bit installations): When setting

```
setenv WRPLOT_WINDOWMANAGER 'X11'
```

or, in bash,

```
export WRPLOT_WINDOWMANAGER='X11'
```

the X11 version is taken.

1.3 Position and size of the WRplot X-window

Optionally, the user can influence the size and position of the X-window by setting specific environment variables, with

```
export <var>=<value>
```

in the bash and

```
setenv <var> <value>
```

in the tcsh, respectively:

```
setenv WRPLOT_WSCALE x.x
```

scales the X-Window by the factor x.x

```
setenv WRPLOT_XALIGN LEFT
```

aligns the X-Window with the left border of the screen; default is aligning with the right border

```
setenv WRPLOT_YALIGN BOTTOM
```

aligns the X-Window with the lower border of the screen; default is aligning with the top border

`setenv WRPLOT_XOFFSET n`
adds n pixels to the x-position of the X-window

`setenv WRPLOT_YOFFSET n`
adds n pixels to the y-position of the X-window (note: pixel count from top to bottom)

If two or more displays are connected to the computer, the window manager considers the whole combined area as one screen. In this case, the above ALIGN variables can define on which of the monitors the WRplot X-window will pop up.

1.4 Input-line commands

After successful installation, the following commands are available:

wrplot-update

For keeping your WRplot installation up-to-date, you should enter this command from time to time.

manwrplot

displays this manual with ghostview at your screen.

wrplot

enters an interactive mode where the source file can be specified later.

wrplot filename[.plot] [parameters]¹

opens the WRplot-window on your screen and shows the first plot contained in the source file *filename*. Entering the *return* key in that (active) window opens the next plot in that file or closes the window if the end is reached.

Note that the WRplot-X11-window is designed for a quick inspection of data; it has no nice (PostScript) fonts, and cannot display encapsulated PS files (but indicates the BoundingBox of EPSF files in the place they would appear).

wrplot filename[.plot] + [parameters]¹

creates a PostScript file *wrplot.ps.1* with the plot of the source file. If *wrplot.ps.1* already exists, the trailing number is incremented. If the file contains more than one plot, each plot is written into an own *wrplot.ps*-file with incrementing trailing numbers. **psappend** combines all **.ps.i(i)* files (*i* or *ii* being the one or two character trailing numbers) found in the present work directory into one PS file *total.ps* with subsequent pages. The **dps** command removes all *wrplot.ps.** files in the present work directory.

wrps filename[.plot] [parameters]¹ means “WRplot-to-PS”. The extension of the WRplot-file must be *.plot* and can be omitted in the call. A PostScript file is generated which has the name of the source file, the extension being replaced by *.ps*. The WRplot-file may contain a number of plots, which then are concatenated to PS file with each plot on a separate page.

¹Additional parameters are written into a temporary file *~/_WRplot-argument* in the form of WRplot variable definitions *WRplot_arg1 [WRplot_arg2 ...]* which then can be read from the WRplot script via an `\INCLUDE` command.

wrpdf *filename[.plot]* [*parameters*]¹ means “WRplot-to-pdf”; similar to wrps, but creating a pdf file. As intermediate product, a postscript file /tmp/scratch_wrplot.ps is written and finally removed.

wrmult *filename[.plot]* *nn* [+] *[*parameters*]*¹ is especially useful for quickly viewing a WRplot file which contains a number of plots, each of about a “standard” size of 20 cm × 15 cm in landscape orientation. The first *nn* plots (*nn* = 2, 4, 6, 8, 10 or 12) of that file are scaled down and mounted onto *one* page. The trailing “+” option creates the PostScript file *filename.plot.multi.plot*.

ttw *filename* [f77 as132]

is a WRplot-based ascii-to-ps converter. The option “f77” supports the classical FORTRAN format by marking the columns 6 and 73. Option “as132” allows for long lines with 132 characters by producing landscape format. The output is written into the file *ttw.ps*. The command **dttw** (including the **dps** command) removes all ttw debris.

psappend

concatenates all ps-files in the current directory into one file *total.ps* which then may contain many pages. This works only for sufficiently simple PostScript files, such as those generated by WRplot.

new *filename*

is equivalent to wrps *filename*; psappend

newall

performs wrps for all WRplot-Files in the current directory, and finally a psappend.

insertpage <*pagename*>

helps in constructing presentations; the *pagename* is assumed to consist of a fixed part followed by a two-digit running number. Example: insertpage page11 renames all files *pagenn.plot* with *nn*=11 and higher by incrementing the counter *nn*, so that the gap at *page11.plot* can be filled by an additional page. Note that you need to execute newall to get the ps-files updated accordingly.

renumberpages <*fixed-part-of-pagename*>

is another tool for constructing presentations; after you have removed pages (i.e. .plot files) from the directory, renumberpages will rename the remaining .plot files in subsequent order, thus removing any gaps. Again, it is assumed that the filenames consist of a fixed part followed by a two-digit running number. Example: renumberpages page renames all files *pagenn.plot* if needed. Note that you need to execute newall to get the ps-files updated accordingly. If there are .ps files left from removed pages, you must remove those ps-files before by hand.

ps2pdf_portrait *filename[.ps]*

converts *filename.ps* into *filename.pdf* while enforcing portrait orientation, i.e. preventing any automatic rotation which might happen otherwise by the standard ps2pdf. Media format is A4.

ps2pdf_landscape *filename[.ps]*

same as above, but for landscape orientation.

ps2pdf_screen *filename[.ps]*

converts PostScript files that are formally in portrait orientation without rotating them; this procedure works for all SCREEN_{*n*} formats to produce pdf files for beamer presentations.

2 The structure of a WRplot file

Preface:

A WRplot file may contain several PLOTS and/or MULTILOTS. The rule is: One PLOT per page, unless it is in a MULTIPLOT (also one MULTIPLOT per page).

All WRplot commands need to be written in CAPITALS; small letters may, of course, within strings. Names of variables are case-sensitive.

Comment lines start with an * and may start (nearly) everywhere. Blanks are allowed in (nearly) all places, too.

Parameters can be separated as desired: a delimiter (comma, equality sign or colon), one or more blanks, or a delimiter and blanks. The TAB-character is interpreted as blanks.

Lines may be up to 132 characters long. KASDEF-commands can have continuation lines (see Ch. 3).

Attention, EMACS users! WRplot is a FORTRAN program. FORTRAN reads in formatted files line by line. Each line must end with a *carriage-return* symbol (CR, “Enter”-button). Some editors, especially EMACS in its default settings, do *not* close the last line of a file with a CR but directly with an *end-of-file* (EOF). The last line can not be read by FORTRAN, then. To avoid this in EMACS or XEMACS, add this line

(setq require-final-newline t) ; always add a newline at EOF

into the configuration file *.emacs* or *.xemacs/custom.el*, respectively. Alternatively one always needs to add a blank line (CR) at the end of the file.

MULTIPLOT START

Optional; all following plots in the file until the occurrence of MULTIPLOT END are being added into a *single* plot.

PAPERFORMAT <Format: string> [Keyword]

Optional; unless specified the default (A4Q) format is used.

Allowed formats: A4Q (default), A4H, A3Q, A3H etc. until A0; the trailing letter “H” stands for german *Hochkant* meaning *Portrait* orientation, “Q” stands for *Quer* meaning *Landscape* orientation.

Furthermore, there are formats defined that are suitable for fullscreen presentations:

- SCREEN has an aspect ratio of 4:3, and implies landscape orientation; this format is not recommended anymore (see below);
- SCREEN2 alias SCREEN4T03 also has an aspect ratio of 4:3, but implies formally a portrait orientation; PostScript files created with this format can be converted to pdf with the tool *ps2pdf_screen* which is available among the WRplot tools;
- SCREEN3 alias SCREEN16T09 has an aspect ratio of 16:9, i.e. corresponding to many wide-screen beamers. Again, the output is formally in portrait orientation and co-operates with *ps2pdf_screen*;
- SCREEN4 alias SCREEN16T010 is similar to SCREEN3, but has an aspect ratio of 16:10.

The optional second keyword might be:

BBNOROTATE, BBROTATE, EPS

The optional keywords *BBNOROTATE* (default) and *BBROTATE* allow the user to rotate the BondingBox. With BBROTATE landscape plots can e.g. be placed in L^AT_EX portrait pages.

When using the optional keyword *EPS*, the produced PostScript file will be marked as an EPS file in its first line without suppressing the showpage at the end (if needed, one can remove this manually from the ps file). This feature was added to improve the integration of WRplot-made PS files into Microsoft “WORD” documents.

FORMATEFACTOR <*scale factor:real*>

(Default=1., optional), scales the following plot or all plots of the multi-plot by the given *scale factor*.

PLOT : [<*plot name:string*>]

Start of a plot. *Plot name* (optional) appears in the console window and may help to identify different plots in the same file, e.g. in the interactive mode.

KASDEF commands (see Ch. 3)

Some KASDEF commands may be placed between the datasets but they are executed *before* the box and datasets are plotted. By KASDEF PAUSE (see below) the execution of the following KASDEF commands is delayed until the datasets are plotted. With KASDEF LATEBOX the plotting of the box is delayed to the end.

HEADER : <*title:string*>

Upper caption of the plot. Following this line a sequence of five lines is mandatory (they describe the plot box, see Ch. 4).

N=... [*Parameter*]

In this line the description of the dataset starts (see Ch. 5).

END or ENDE: End of the plot

MULTILOT END or MULTILOT ENDE finishes the multi-plot started with MULTILOT START.

3 The KASDEF commands

3.1 Preface

The term KASDEF has historical reasons (german “Kastendefinition”, i.e. definition of the box). KASDEF commands start with the word KASDEF. It’s more modern to use the Backslash (\) instead (both are valid, though). Error messages (and partly this manual) still use the old naming KASDEF.

KASDEF commands may have *continuation lines*. They start with
`\> <continuation...>`

and work as if the previous line’s KASDEF command were continued behind the > character.

(Nearly) all KASDEF commands that have a parameter with the meaning in *units* may have these replaced by predefined constants **XMIN**, **XMAX**, **YMIN**, **YMAX** and **PXMIN** etc., where XMIN etc. are the coordinates of the BoundingBox and PXMIN etc. are those of the page. This option is however partly obsolete and should not be used anymore. It’s more convenient to use (proper *variables* instead (see below).

KASDEF commands may contain **variables** which are replaced by their current content during the program execution. Variables (see Ch. 3.3.2) are marked by a preceding dollar sign (\$, like in Unix). In principal they are *Strings*, but one can also calculate with them if their contents are numbers (see, e.g., \CALC). The command \PREDEFINE-VARIABLES (see page 15) creates a whole set of standard variables, e.g. \$XMIN.

Note that variables are expanded (i.e. names are replaced by their content) nearly everywhere. When printing text, each \$ is interpreted as the beginning of a variable and the program tries to replace it. In case a dollar sign shall be printed, it needs to be escaped by \\$ (also see Table 1). *Exception:* As long as a plot has no variables defined, the substitution of variables is switched off.

For convenience, some of the plot commands now allow to write arithmetic expressions instead of a simple argument. Such expressions must be enclosed in parantheses. If the expression contains delimiters, it might also be necessary to enclose the whole argument in quotation marks.

Example:

```
\SYM (0.5*cos($alpha)) (0.5*sin($alpha)) 5 5 .2 4
```

Furthermore, in many cases it’s possible (try!) to use the code word SAME in place of repeated parameters, e.g.:

```
\SYM 3. 4. 0. 0. 1. 4
```

```
\SYM SAME SAME 0. 0. 1. 8
```

plots a square and a circle centered at the same coordinates.

In many KASDEF commands, parameters may be specified to be taken logarithmically, or to be raised to a power of ten. For example, one can write LOG35000 (without blank!) for $\log(35000)$ and DEX5. for 10^5 . Arithmetic expressions instead of parameters are not yet possible but planned for future versions of the program (so far one has to use the CALC command to prepare parameters).

3.1.1 Three types of KASDEF commands

The execution of WRplot is split into three phases:

- Reading the input file
- Manipulation of data (only in the *Interactive mode*)
- Printing the plot

WRplot has three types of KASDEF commands:

Settings are interpreted during the read-in phase of the WRplot file. They set certain specifications for the plot which cannot be changed later. The `\INCLUDE` command, which inserts the KASDEFS from an external file, belongs to the *Settings*, too.

Instructions are KASDEF commands that do *not* perform plotting actions, but, e.g., deal with variables. Instructions can be arranged inside INSTRUCTION EXPORT blocks; such blocks are already executed during the read-in phase of the plot file. Any KASDEF commands outside such blocks are first stored in a buffer and executed later during the phase when the plot is created.

Plot commands create the output of text and graphics or relate to such actions (e.g. definition of the current color by `\COLOR`). As Plot commands are not executed in the read-in phase, and are therefore *not allowed* inside an INSTRUCTION EXPORT block.

Because some old WRplot scripts may violate this strict rule, WRplot enters a *compatibility mode* if a *Plot command* is found inside an INSTRUCTION EXPORT block. In this case, all commands of the INSTRUCTION EXPORT block are executed a second time during the Plot phase. A WARNING is issued, because this might cause spurious effects.

Settings and *Instructions* may only appear in the preamble, e.g. *before* the block that describes the box (starting with HEADER). *Plot commands* may in contrast also appear further below, i.e. between the data blocks.

3.2 Settings

KASDEF commands of this type are read first, as they pre-specify certain values which cannot be modified later anymore. These commands must appear *before* the HEADER line (otherwise they are ignored and produce a non-fatal error message).

`\INTERACTIVE`

enters the interactive mode; the same mode can be reached by the interactive menus, when calling `wrplot` without filename. Clearly, it only works with the X11 window (i.e. not when a PostScript file shall be produced).

The INTERACTIVE mode offers many possibilities, especially by means of *graphic input* with the mouse. Most of the menus are self-explaining. Possible applications are:

- zooming into the data;
- rectifying spectra.
- measuring equivalent widths; this works as follows:
 1. display the spectrum in INTERACTIVE mode;
 2. press button POINTS

3. Option a: define by clicking *two* points on the continuum; then the EW will be measured between these two points;
Option b: define *more than two points*, starting and ending on the continuum; then these points define the area (profile) for which the EW will be measured;
4. klick ADD POINTS
5. klick EQUIV
6. only if option a): klick (on the right-hand side) LAST or, if the plot shows more than one dataset, the number of the dataset to be measured;
7. → the measured area becomes black (option a) or is shown by a polygon; EW and line flux are displayed on the plot and also printed in the text window from which wrplot was called;
8. before a new measurement, first klick DEL ALL P – the plot then refreshes, and the points which have been set before disappear.

\PENDEF *<pen:integer>*

Defines the default value of the line thickness for datasets (in units of typographic points, i.e. 1/72 inch).

\DEFAULTCOLOUR *<n:integer>*

Defines the default color of the datasets. If not specified, it is 1 (black).

\OFS *<x_{offset}:cm> <y_{offset}:cm>*

Sets the offset of the lower left corner of the plot box relative to the corner of the page. The default is (4., 3.). This option is especially needed to arrange multiple plots on a single page (MULTILOT).

\SKL *<Scale factor f:real>*

Scales the following plot by the given *scale factor f*. The default is 1. This is often needed to arrange multiple plots on a single page.

\INBOX

Datasets are clipped to the inside of the box. Graphical objects plotted by KASDEF commands are not subject to this clipping, unless the corresponding KASDEF command stands after a PAUSE command and is hence plotted after the datasets. Filled areas which are partly clipped might suffer from spurious effects.

\NOBOX

The plot box (including tick marks and their labels) are suppressed. The header (plot caption) and axes descriptions are not affected.

\LATEBOX

The data box (including tick marks etc.) are plotted *after* the datasets, i.e. can overplots other objects.

\DEFAULTS

defines a coordinate system with scales of 1 cm per unit in both axes. The origin is at the lower left corner of the page (i.e. it implies \OSF 0 0), and the axes are not shown (i.e. it implies NOBOX). This DEFAULTS setting is mutually exclusive with the definition of the coordinate box as described in Sect. 5.

\LETTERSIZE <size *r:real*>

Size of the characters for labels of the plot box (tick labels, axes descriptions, plot caption).
Default: 0.4

\TICKSIZE <size *r:real*>

Size of the (small) ticks at the plot box (the numbered ones have the double length); by default their size is harmonic to the LETTERSIZE (same numerical value).

\SET_NDATMAX <*n.points n:integer*>

Changes the maximum number of data pairs in a dataset. The default is NDATMAX = 100000. The maximum value allowed is 10 times this default. Increasing NDATMAX leads to a corresponding decrease of the maximum number of datasets, NSETMAX.

\SET_NSETMAX <*n.sets n:integer*>

Change the maximum number of datasets. The default is NSETMAX = 100. The maximum value allowed is 10 times the default. Increasing NSETMAX leads to a corresponding decrease of the maximum number of data pairs in a dataset, NDATMAX, such that $NSETMAX \times NDATMAX \leq 10^7$.

\INCLUDE <filename> [*INCKEY*=<string>]

Reads and includes all KASDEF commands from the given file at that point in the WRplot file. Lines are checked for KASDEF commands (i.e. for lines starting with \ or KASDEF); other lines are ignored.

filename can denote a local file, or a filename preceded by an absolute or relative path. If the path name starts with ~<username>, this will be properly expanded. If *filename* is not found, but there exists a file with that name plus the additional extension “.gz”, WRplot assumes that this input file has been compressed with gzip. In this case, WRplot writes automatically an unzipped copy of that file to /tmp/wrplot_scratch_gunzip_<username> and reads from this copy.

If the keyword **INCKEY**=<string> is set, the reading starts below the first line which starts with the given *string*; reading ends when a line “END” is encountered (or at the end of the file).

If the include-key is a variable, this variable must be defined inside an INSTRUCTION EXPORT block (see below). If this variable contains blanks (or other delimiters), the variable name must be enclosed in double (!) double quotes, for instance: **INCKEY**=<“\$varkey”>.

\INSTRUCTION EXPORT

until

\INSTRUCTION NO-EXPORT

All *Instructions* between these two lines constitute an “INSTRUCTION EXPORT” block. All commands inside such blocks are already executed during the read-in phase. This is needed, for instance, to set variables or create data which are used when reading the datasets.

3.3 Instructions

3.3.1 Output in the Terminal/command window and System calls

\ECHO *string*

Writes the String to the Terminal/command window

\SYSTEM *commands-to-be-passed*

WRplot allows shell calls, i.e.

`\SYSTEM echo The file $file has `cat $file | wc -l` lines.`

The line's content after SYSTEM is passed to a **FORTAN** shell call, i.e. to Bash. Note: Variables are expanded before they are passed to the shell. To use system variables, use `\$` (e.g. `\$PWD`). Variables are not handed back to WRplot (see the example at `\READ`, e.g. Sect. 3.3.5).

3.3.2 Variables

Variables are used by WRplot in a similar syntax as in UNIX. A dollar sign (\$) preceding a variable name refers to the *content* of this variable. Note that, like in UNIX, variable names are case-sensitive. The length of a variable name, as well of a variable's content, is limited to 132 characters. The end of the variable's name is indicated by a blank or another delimiter out of the set `,=:+-*/^{}()"$` — these characters cannot be part of the variable's name, thus. For compatibility with older versions, the arithmetic operators `+*/` are still — until further notice? — allowed, *unless* this variable is used in `\CALC` operations. To separate a variable's name from directly following text, enclose it in double quotes, e.g. `"$Mv"mag` (the quotes are interpreted as delimiters and removed).

Square brackets ([,]) are allowed at the end of variables' names to indicate/emulate *indexed variables*. In case the index is a variable (with preceding \$), it is first replaced by its value, then all other variables are interpreted. Example:

```
\CALC x2[$i] = $x[$i]**2
```

Blanks are not allowed between the square brackets. Variables with multiple indices are not allowed yet.

\VAR *A = ...*

Fill a variable (here *A*) with the given content.

\PREDEFINE-VARIABLES

Defines a couple of standard variables, e.g.:

- the minimum and maximum value of the plot's coordinate box, as well as the midpoint values of both axes (in their respective units) (`XMIN`, `XMAX`, `XMID`, `YMIN`, `YMAX`, `YMID`)
- the coordinates (in units) of the paper borders and center of the page (`PXMIN`, `PXMAX`, `PXCENTER`, `PYMIN`, `PYMAX`, `PYCENTER`)
- the scale factors to convert units to cm (`XSCALE`, `YSCALE`)
- filename, date and time (`FILENAME`, `DATE`, `TIME`). Attention, only these three variables are immediately accessible after the *predefine* command; all others are defined after the coordinate box has been specified.

\GETTIME

Write the current time (format: hh:mm:ss) into the variable `TIME`.

\FITSVAR *fitsfilename* *parameter=value* [*parameter=value*] ...

reads header-entries from the fits file *fitsfilename*² into WRplot variables with the same name.

The "parameter-value" pairs are evaluated in the sequence as given. The following pairs parameters=value are allowed:

NHDU=n

chooses the "unit" n (n is integer number) within the fits file. Default is n=1. Non-simple fits files can contain NEXTEND units, each of them carrying an own HEADER. Note that NEXTEND is given in the HEADER of the first unit. One might use fitsview (fv) to inspect the structure of a fits file and the FEADER keywords.

VAR=keyword

reads the header entry with "keyword"; a WRplot variable with the same name "keyword" is created and filled with the value given in the fits header.

Example:

```
\FITSVAR fitsfile.fits VAR=NEXTEND NHDU=2 VAR=EXPTIME
```

returns the number of extensions into the WRplot-variable NEXTEND, and the exposure time as given in the header of unit 2 into the WRplot-variable EXPTIME.

\CALC *A = expression*

Calculate arithmetic expressions. The syntax is quite flexible and robust. It mainly corresponds to the syntax rules of FORTRAN. The following arithmetic operators are allowed:

+ - * /, power (a^b) with ** or ^ and functions as well as nested parentheses. Constants may be in the following formats: I, F, E (Integer, Float/Real, Exponential). Variables need the preceding \$, of course. Blanks are not significant.

The following *functions* are intrinsic (the names of these functions may be typed in small or in capital letters, but not mixed):

SIN, COS: sinus, cosinus; in contrast to FORTRAN, the argument is in degrees;

ATAN: arcus tangens; in contrast to FORTRAN, the result is in degrees;

DEX: dex(\$x) is equivalent to 10^x ;

LOG: decadic logarithm

EXP exponential function;

LN: natural logarithm;

SQRT: square root;

ABS: absolute value

RAND: random number in the range (0.0, 1.0); the argument of this function decides: RAND(0) → each run yields the same sequence of random numbers; RAND(1) (or any argument ≠ 0) → each run yields different random numbers. Actually, the alternative executables behave slightly different in this respect: in the *gnu* version, the *first* call of the RAN

²for allowed pathnames and zipped files see description of \INCLUDE on 14

function decides the seed behaviour, while in the *intel* versions *all* calls must have the zero argument in order to reproduce the random sequence;

INT: cuts off the decimals (like in FORTRAN);

NINT: rounding to the nearest integer (like in FORTRAN).

\CALC_DEBUG

or \CALC_DEBUG ON enables debug output in the console window showing how \CALC evaluates the arithmetic expression step by step.

\CALC_DEBUG OFF Switch off the debug output (default).

\EXPR A = \$var1 // \$var2

concatenates the strings in the two variables;

for historical reasons, the operator between the variables can also be an arithmetic operator (+, -, * or /), but one should now use \CALC instead.

\VAR-LIST

prints the current list of all variables and their content to the terminal.

\FORMAT *formatstring* \$var

Formatting of a real number according to the *formatstring* which has to be specified in FORTRAN syntax, including the parentheses, e.g. (F5.2), (PG8.2) or (E8.2). Use doublequotes if the string contains delimiters.

\FORMATI *formatstring* \$var

formatting of an integer number according to the *formatstring* which has to be specified in FORTRAN syntax, including the parentheses. Use doublequotes if the string contains delimiters. Example: a variable *i* = 1, after

```
\FORMATI "(SP,I4.3)" $i
```

has the content +001

\FORMATA *formatstring* \$var

formatting of a string variable according to the *formatstring* which has to be specified in FORTRAN syntax, including the parentheses. Enclose in doublequotes if the *formatstring* contains delimiters. Note that the format specifier "A" may not be used without specified length, and that the length of the re-formatted string may not exceed 132 characters. **FORMATA** can be conveniently used to augment a string with leading blanks, as in the following example:

```
\FORMATA "(3X,A10)" $text
```

\CUTVAR *var n : m*

Cutting a string that is content of a variable. The new value of *var* will be the substring (*n : m*) where *n* and *m* are the character positions within the string (starting with 1). The colon between *n* and *m* stands for any delimiter or blank. The first argument may be left blank (i.e. *var : m*) or the second may be blank, i.e. *var n :* (with obvious defaults). If *n* and/or *m* are negative, the position is counted from the end.

Example: \CUTVAR \$var 1:-5 cuts off the last five characters of \$var. Here the dollar sign of \$var may be omitted.

\PARSEVAR *varname n*

The string in variable *varname* is parsed into arguments (with the delimiter rules as described in the handbook at beginning of Sect. 2, and the content of the variable *varname* is replaced by its *n*-th argument.

varname can be written with or without leading '\$'. If *n* is larger than the number of arguments in the input string, the output string will be empty.

3.3.3 IF (conditional) constructions

Like in FORTRAN, one can build (nested) constructions of IF, ELSEIF, ELSE and ENDIF. In contrast to FORTRAN syntax, the logical expression may only be a simple comparison (no **.AND.** and **.OR.**). Neither parentheses nor the keyword THEN are allowed.

The operators **.EQ.** and **.NE.** compare the operands as strings (i.e. 0 is different from 0.0 here!); the other operators **.LT.**, **.LE.**, **.GT.** and **.GE.** assume the operands as numbers and compare them accordingly.

The command ELSEIF must be written as one word.

Example:

```
\IF $k .LT. $l
  \ECHO branch 0
\ELSE
  \IF a .EQ. b
    \ECHO branch1
  \ELSEIF a .EQ. $c
    \ECHO branch 2
  \ELSE
    \ECHO branch 3
  \ENDIF
\ENDIF
```

3.3.4 DO-Loops and GOTO

One can construct (nested) DO loops with KASDEFs, too:

```
\DO labelname l=start end [increment]
  \... kasdef statements)
\LABEL labelname
```

```
\LABEL labelname
```

(with the same *labelname* as specified in the \DO statement) marks the end of the loop. The control parameter must not be integer. Nested loops may end at the same LABEL.

Note: WRplot does not check for repeated allocation of the same *labelname*; a DO loop ends where the first fitting LABEL is found.

```
\GOTO labelname
```

Jump to the line with the *first* occurrence of \LABEL *labelname* in the plot file, starting the search from the beginning of the file.

3.3.5 Write into/read from a file

Only one file can be open at a time for reading or writing with KASDEF commands. A unix filename is assigned by:

```
\OPEN filename
```

```
...
```

```
\WRITE ...
```

```
\CLOSE <filename>
```

If the OPEN command is absent, the default filename is `fort.50`. If the CLOSE command is absent, the file will not be closed and may either be incomplete or grow with each new execution of the WRplot file.

Note: If a file is opened and written with an INSTRUCTION EXPORT block, the file written can already be read as a dataset in the dataset section. This provides very flexible possibilities.

```
\READ varname
```

reads from the currently open file. The current line will be stored in the variable `$varname`. If reading reaches the end of the file, `$varname` gets the content E-O-F.

Example:

```
\VAR n=0
\OPEN inutfile
\LABEL begin
  \READ line
  \IF $line .EQ. E-O-F
    \GOTO continue
  \ENDIF
  \CALC n = $n + 1
  \FORMATI "(I2.2)" $n
  \ECHO $n $line
\GOTO begin
\LABEL continue
\CLOSE
```

This example reads from *inputfile* and echoes line-by-line on the console with a preceding two-digit line number, till the end of the file is reached.

3.4 Plotcommands

Plot commands allow the output of text or graphics or relate to such output.

```
\PAUSE
```

the execution of the subsequent KASDEF commands is postponed until the datasets have been plotted.

3.4.1 Changing line thickness, color and font

```
\PEN <pen: integer>
```

sets the line thickness hereafter to *pen* typographical points (default: PEN=1).

\FONT <font:string>

sets the current font for text output in the plot. The following *fonts* are defined: WRPLOT (default), TIMES, HELVETICA (in short: HELVET), COURIER or ZAPF (see page 51).

\COLOR <color *i*:integer>

sets the color for graphical and text output by the subsequent KASDEF commands. Colors are encoded as one-digit integers ($i = 0 \dots 9$), using the current color definitions. See Fig. 3 for the default definitions. The default is COLOR=1.

\DEFINECOLOR <color *i*:integer> <*r*:real> <*g*:real> <*b*:real>

It is possible to change colors in WRplot, i.e. to plot in more than ten colors or just to alter them. The new color definition is valid from the point of definition onward (either to the end of the plot – if using a MULTIPLOT it needs to be defined for each PLOT – or until the color is reset/redefined).

The values r , g and b are the relative intensities of red, green and blue, respectively; they can each range between 0 (no intensity, i.e. '00' in hex) and 1 (full intensity, i.e. 'FF' in hex). Colors are mixed in an additive way.

The colors 0 to 9 are predefined according to the following table (also see Fig. 3). For usage with projectors one can use a set of predefined colors with darker background, see `wrplot.dir/screen.kasdefs` and the right part of Fig. 3.

number	<i>i</i>	0	1	2	3	4	5	6	7	8	9
name		white	black	red	green	dark blue	yellow	pink	light blue	light green	blue
red	<i>r</i>	1.	0.	1.	0.	0.	1.	1.	0.	0.5	0.0
green	<i>g</i>	1.	0.	0.	1.	0.	1.	0.	1.	1.0	0.5
blue	<i>b</i>	1.	0.	0.	0.	1.	0.	1.	1.	0.0	1.0

\RESETCOLOR

Restore all predefined colors to their original values (i.e. see the above table). Luxury!

\LIMITCOLOR = *nr ng nb*

Limit the number of colors usable with DEFINECOLOR. It coarsens the steps of the RGB-color cube to nr , ng and nb steps. This only applies to the X11-window (if the graphic controller of the computer needs some adjustment), but not the generated PS-file(s). Should be obsolete with modern computers.

\NCOLORSTEP = *nstep*

This command is only used in combination with the plot SYMBOL=40 (false-color plot) and limits the number of color steps *per color table*. This only applies to the X11-window (if the graphic controller of the computer needs some adjustment), but not the generated PS-file(s). Should be obsolete with modern computers.

3.4.2 Graphics: drawing actions

Nearly all plot options in the following description allow to specify coordinates by four parameters, x, y in the units of the plot, plus offsets in x and y , named x_{offset} and y_{offset} in centimeter.

In the following, these parameters are not repeatedly described. Parameters in [square brackets] are optional, while mandatory ones are placed between < and >.

\LIN <x1:cm> <y1:cm> <x2:cm> <y2:cm>

draws a straight line from (x1,y1) to (x2,y2) (parameters here only in cm!).

\LINREL <x:units> <y:units> < Δx :cm> < Δy :cm> [<x_{offset}:cm> <y_{offset}:cm> <SYMBOL=*i*> <SIZE=*x*>]

Plot a line from a starting point (x, y) with the length (Δx , Δy) in cm.

The following parameters are optional:

- Offsets in *x* and *y* (either none or both need to be specified);
- A different SYMBOL can be chosen for the line, i.e. dashed lines, SYMBOL=*i*, where *i* is the number of the symbol, see Fig. 5 (note: discrete symbols only mark the start and end point, not in between).
- A different SIZE can be chosen as well, the default is SIZE=0.5

(if SYMBOL and/or SIZE are used, one needs to specify the offset before these keywords).

\LINUN <x1:units> <y1:units> <x2:units> <y2:units> <x_{offset}:cm> <y_{offset}:cm> [<SIZE=*x*> <SYMBOL=*i*>]

Plot a line between the points (x1,y1) and (x2,y2). The keywords SIZE (default: 0.5) and SYMBOL (default: 5) can change the size and style of the line, i.e. to plot a dashed line.

\ARC <x:units> <y:units> <x_{offset}:cm> <y_{offset}:cm> <radius:cm> < α :degree> < $\Delta\alpha$:degree> [<FILLED>]

Plot an arc of a circle at the given coordinates with given radius, starting at the angle α and with an arc of length $\Delta\alpha$ ($\alpha = 0$ is to the right, $\Delta\alpha$ counting in the mathematically positive sense). With the keyword FILLED the sector ("piece of cake") is filled.

\ELLI <x:units> <y:units> <x_{offset}:cm> <y_{offset}:cm> <a:units> <b:units> < ϕ :degree> < α :degree> < $\Delta\alpha$:degree> [<FILLED> <SYMBOL=*i*> <SIZE=*x*>]

Plot an arc of an ellipse with half axes *a* and *b*, where the longer axis is tilted by the angle ϕ . The arc starts at the angle α (the angles are taken for a circle of radius $r = \max(a, b)$ before the axes' ratio is applied) and has the length $\Delta\alpha$. If the keyword FILLED is set, the (arc of an) ellipse is filled. By setting SIZE and SYMBOL, one can choose the style of the arc (see Fig. 5).

\ARR <x:units> <y:units> <3rd parameter> <4th parameter> <size *r* of the arrow's tip:cm> <x_{offset}:cm> <y_{offset}:cm> <Mode:0,1,2,3,4,5> [<FILLED> <BAR> <SHRINK=*f*:real>]

Plot an arrow. If the keyword BAR is set, the arrow's foot gets a perpendicular bar, if the keyword FILLED is set, the arrow's head is filled, with SHRINK=*real* one can multiply the arrow's length by a given factor *f*, i.e. to leave some elegant space between the connected points.

Example: \ARR 0 0 5 45 0.25 0 0 0 FILLED

They have the starting point (x,y) and the fifth parameter as the size of the arrow's tip (if the keyword BAR is set, the bar will have the same size) in common.

There are different modes (specified as seventh parameter). The meaning of the third and forth parameter depending on that mode:

Mode	Interpretation of the third and fourth parameter
0	3 rd par.: Length of the arrow in cm or units ⁽¹⁾ ; 4 th par.: Set the angle to which the arrow points (0: right, 90: top etc.);
1	as in mode 0, but x_{offset} is the offset in direction of the arrow; y_{offset} is ignored
2	3 rd par.: Length of the arrow in x direction in cm or units ⁽¹⁾ ; 4 th par.: Length of the arrow in y direction in cm or units ⁽¹⁾ ;
3	as in mode 2, but y_{offset} is ignored here
4	3 rd par.: x coordinate of the tip in units; 4 th par.: y coordinate of the tip in units;
5	as in mode 4?

Note ⁽¹⁾: Lengths (in modes 0 and 2) can be specified in units if an U is directly attached to the number, i.e. 5U.

\RECT *<x1:units> <y1:units> <x2:units> <y2:units> <x_{offset}:cm> <y_{offset}:cm> [<FILLED>]*
Plot a rectangular between the given corners (x_1, y_1) and (x_2, y_2).

\RECTLUN *<x:units> <y:units> <x_{offset}:cm> <y_{offset}:cm> <width:cm> <height:cm> [<FILLED> <ROTATE= α :degree>]*
Plot a rectangular. Depending on the prefix of x_{offset} and y_{offset} (L, R, M in x for *left, right, middle* and U, D, M in y for *upper, lower, middle*) the coordinates (x, y) refer to the left, right, upper, lower or middle of the rectangle. If no prefixes are given, the coordinates refers to the lower left corner.

The rectangular has dimensions *width, height* in cm, with attached U in units (c.f. \ARR). If the keyword FILLED is set, it will be filled. If ROTATE= α is set, the figure will be rotated (the reference point of rotation is set by the offsets) by angle α .

Note: It's useful, e.g. when plotting a box around some text, to use SAME instead of (x, y) and x/y_{offset} . SAME takes the prefixes of the offsets used in the LUN command, i.e. one has the position NEXTLUN would have. This makes plotting boxes around text rather easy:

```
KASDEF LUN 5 2 M0 M0 0.22 Text text text
KASDEF RECTLUN SAME SAME SAME SAME 2.5 0.7
```

\SYM *<x:units> <y:units> <x_{offset}:cm> <y_{offset}:cm> <size r :real> <symbol j :integer> [<CFILL= i :integer>]*
Plot the symbol j (numbers see Fig. 5 – only discrete symbols are allowed here!) in size r at the position (x, y). If CFILL= i is set, fill the inner part of the symbol (if it's a circle, square etc.) with color i and have its border in the current plot color.

\BARLEN *< $\Delta_{y,\text{up}}$:units> < $\Delta_{y,\text{down}}$:units> < $\Delta_{x,\text{up}}$:units> < $\Delta_{x,\text{down}}$:units>*
Define the lengths of the error bar in $\pm y$ and $\pm x$ direction in units of x, y . Default: 0.,0.,0.,0. (i.e. no bars).

\BARPAR *<space:cm> <edge:cm>*
Specify the space left out at the center (at the bars' intersection) and define the length of the edges (i.e. small perpendicular foot bars at the end of the error bars). Default: 0.,0.

\BAR <*x: units*> <*y: units*> <*x_{offset}: cm*> <*y_{offset}: cm*>

Plot an error bar at (*x*, *y*) with the previously specified properties.

Note: One *first* needs to specify the BARLEN (and BARPAR) parameters at least once *before* setting the BAR, else it will plot error bars of zero (=default) length.

3.4.3 Hatched Areas

Various areal objects can be filled with a color, e.g. by the optional keyword FILLED (see, e.g., \ARC or \ELLI). Discrete symbols (cf. \SYM) can be filled by specifying a negative SIZE, or with the parameter CFILL=*n*, where *n* specifies the color of the filling.

Alternatively to a full color, the area can be filled with a hatched pattern. Note that hatched patterns *only* appear in the PostScript files, while in the X11-window the area is just filled smoothly.

For hatching, one uses the keyword HATCHED instead of FILLED. In \SYM, the value of SIZE must be positive now, while CFILL can still be used to define the color of the hatched pattern.

The hatched pattern can be specified in detail by further keywords (actually, the keyword HATCHED is redundant as soon as such hatching specifications appear):

HTYPE = *t*

with *t* = [D, V, H, K, C] defines the pattern of the hatching; **D**: (default) diagonal lines; **V**: vertical lines; **H**: horizontal lines; **K**: checked hatching (vertically and horizontally); **C**: cross-hatched (diagonal).

HW=*x.x*

sets the thickness of the hatching lines. *x.x* is in typographical points, i.e. 1/72 inch as in the PEN commands. For hatching even decimal fractions are allowed.

HSEP=*x.x*

sets the spacing between the hatching lines. The value is in typographical points as for HW (see above).

3.4.4 Text output

\LAB <*x: cm*> <*y: cm*> <*size r: real*> <*text: string*>

Write the text *string* of size *r* at the location *x*,*y*.

\LUN <*x: units*> <*y: units*> <*x_{offset}: cm*> <*y_{offset}: cm*> <*size r: real*> <*text: string*>

Plotting a text string at a position given in the coordinate units (LUN = Label at UNits). An offset in cm can be given additionally. *x_{offset}* and *y_{offset}* can have prefixes. In order to understand their meaning, imagine the text string represented by its bounding-box. By default, the string is placed such that the coordinates refer to the lower-left corner of that box. For *x_{offset}* precede by the letter **L** (default), **M** or **R**, the string is placed with its *middle* (in *x* direction) or with its *right* boundary at the specified coordinates. Similarly, one can use the prefixes **D** (default), **M** or **U** for *y_{offset}*. The coordinates then refer to the *baseline* (D), the *middle*, or the *upper* boundary of the string's bounding-box. Note that these features make it very convenient to place labels close to discrete plotsymbols.

\LUNA *<x: units> <y: units> <x_{offset}: cm> <y_{offset}: cm> <size r: real> <angle: degree> <text: string>*

issues a text string like LUN, but the string is rotated counterclockwise by the given angle; the reference point for rotation is set by the coordinates including the offsets. Hence the prefixes R, M etc. allow to rotate around the respective corner or midpoint of the string's bounding-box.

\LINUNLAB *<x1: units> <y1: units> <x2: units> <y2: units> <x_{offset}: cm> <y_{offset}: cm> <S_{off-center}: cm> <size r: real> <text: string>*

Plot a line from (x1, y1) to (x2, y2) and write the text *string* at the center of that line, shifted by *S_{off-center}*.

\LUNINC *<x: units> <y: units> <x_{offset}: cm> <y_{offset}: cm> <size r: real> <filename: string> <prefix: string> <search string: string>*

Grep the first line in *filename* that starts with the *search string*, and issues that line as text string like LUN would do, but with preceeded by the string given as parameter *prefix* (e.g. &F&2 as prefix string writes the line in boldface and red color). The writing position is given by the coordinates.

\NEXTLUN *<text>*

writes another line of text. The position (coordinates, offsets, prefixes) are taken from the previous LUN command, but shifted by a LINESKIP to write into the next line. An error will occur if no previous writing position has been defined by a corresponding LUN command.

\SAMELUN *<text>*

Like NEXTLUN, but without a LINESKIP in order to write into the same line.

\TEXT [*<x: units> <y: units> <x_{offset}: cm> <y_{offset}: cm> <size r: real>*]
until

\ENDTEXT

All lines between these two commands are considered as a paragraph of running text. Line breaks are created automatically according to the currently set RIGHTMARGIN. *Hyphenation* is semi-automatic: hyphenation breaks can only happen where indicated by hand by “\-” (like in \LaTeX). Example: hyphen\ -ation

Line spacing is the same as LINESKIP by default, but can be chosen differently by the command TEXTLINESKIP.

The parameters (position, offsets and size) are optional; if they are not set, the first line of the floating text appears in the same place as NEXTLUN could write.

3.4.5 Text output options

\NEWSIZELUN *<size r: real>*

Change the font size of LUN to *r*.

\LINESKIP *<spacing l: cm>*

Set the line spacing (for NEXTLUN) to the given value. The default unit of *r* is cm. If an S or U is attached to the number (i.e. 0.5S), it is interpreted in units of symbol size (S) or

y-units (U), respectively. The default corresponds to `LINESKIP=2.0S`, which means $2.0 \times$ the current font size.

\TEXTLINESKIP *<spacing l:cm>*

Set the line spacing for continuous text (for `TEXT`) to the given value (default: same as `LINESKIP`). Again, an attached S or U is interpreted as units of symbol size or y-units, respectively.

\MEDSKIP

Insert half a `LINESKIP` as line spacing at that position, i.e. between two lines.

\TABLUN *<tab l:cm>*

adds a horizontal offset to the current writing position; the parameter *tab* may carry a prefix (L, R or M) which then replaces a prefix that has been set before.

\RIGHTMARGIN *<x:cm>*

Set the right margin for floating text and for horizontal lines (see `\HLINE`). Default is 18 cm.

\BLOCK or **BLOCK ON**

sets a parameter that floating text appears in block mode, i.e. each line is stretched till the `RIGHTMARGIN`. Default is ragged-right margin.

\BLOCK OFF

returns to ragged-right margin for floating text.

\BGRLUN [*<L=x.x, R=x.x, U=x.x, D=x.x > <OFF> <ON> <RESET> <COLOR=i>*]

fills the text box with color *i* before printing the text. This works for all subsequent `LUN`, `NEXTLUN` and `LUNA` commands (but not for `TEXT`), as well as included eps files (see Sect. 3.4.12) or LATEX blocks. Default for the underlying color is *i=0* (white, if not re-defined).

The filled box is larger than the text bounding box a margin; its widths can be with the parameters L, R, U and D (left, right, upper, lower; in units of the current font size). By default, all margins are 0.5.

If `COLOR=i` is negative, the box is not filled but only surrounded by a line with color `abs(i)`. One can also be filled with hatched patterns (see Sect. 3.4.3).

Note: The underlying box does *not* belong to the object as such, i.e. so its position and `BoundingBox` are not affected by `BGRLUN`.

BGRLUN OFF

disables the `BGRLUN` feature. With a later `BGRLUN` one enters it again with the parameters kept from the previous use. All defaults (color, L, R, U, D) are restored by the option `RESET`.

\HLINE

draws a horizontal line between the current writing position (where `NEXTLUN` would write) and the `RIGHTMARGIN` position. If `HLINE` is drawn within the `TABON` environment, it covers the width of the table.

3.4.6 Itemizing

\ITEMIZE *<size r:real> <string>*

starts an itemizing environment; subsequently, all text lines or paragraphs written by LUN, NEXTLUN or TEXT are indented and marked by *string* in the font size *r* (or SAME). Examples: \ITEMIZE SAME &2\o

marks the items with a red bullet;

\ITEMIZE SAME &4\>

marks each item with a blue arrow

\ITEMIZE OFF

closes the itemize environment.

\ITEMSEP *<spacing l:cm>*

specifies an additional space between items (default: 0cm) *l* is in cm, or in units of the current font size if appended with letter S.

3.4.7 Tables

\TABON *<columns> [<prefix>]*

enters the tabular environment. The string *columns* consists of the letters L, R or C, each letter representing one column. The meaning of the letters is analog to L^AT_EX syntax and describes the positioning of the entries: (**L**: left-justified, **R**: right-justified, **M**: centered).

The optional parameter *prefix* will precede each entry in the table.

Example:

\TABON LLL &F&2

opens a table with three columns, where all entries are flushed left, and all entries are in boldface style and color 2.

The table starts at the current position (i.e. where NEXTLUN would write). In *x* direction the current prefix is taken, i.e. M centers the table. In that sense TABON acts like a NEXTLUN.

\TAB *<strings>*

writes a row in the table. The string is parsed according to the usual rules, i.e. the arguments are separated by blanks and/or delimiters, while quotation marks keep an argument together. Each argument then is a column entry. The number of arguments *must* agree with the number of columns declared in TABON.

\TABOFF

quits the tabular environment.

\DATE [*<x:units> <y:units> <x_{offset}:cm> <y_{offset}:cm> <size r:real> <angle:real>*]

Write the date string ("Plot created on dd-Mon-yy hh:mm:ss" where *Mon* is the three character abbreviation of the month) at the given position or (default) at the right side of the plot in vertical alignment.

\FILE [*<x:units> <y:units> <x_{offset}:cm> <y_{offset}:cm> <size r:real> <angel:real>*]

Write the file name of the current file at the given position or (default) at the right side of the plot in vertical alignment (a bit right of the position DATE would have).

3.4.8 Special characters, Umlauts etc.

See Appendix A (page 51) to have an overview of the special characters WRplot can print in different fonts. Special characters are encoded by a prefixed `\` or `&` (see Table 1). For other special characters (i.e. \mp , \acute{o} , \hat{s} etc., use \LaTeX instead).

Table 1: Encoding of special characters with `\` and `&`

Code	Symb.	Code	Symb.	Code	Symb.	Code	Symb.	Code	Symb.
<code>\S</code>	\circ	<code>\A</code>	\AA	<code>\M</code>	\dot{M}	<code>*</code>	$*$	<code>\8</code>	∞
<code>\></code>	\rightarrow	<code>\<</code>	\leftarrow	<code>\+</code>	\pm	<code>\o</code>	\bullet	<code>\\</code>	\backslash
<code>\#</code>	$\#$	<code>\'e</code>	\acute{e}	<code>\'e</code>	\grave{e}	<code>\~n</code>	\tilde{n}	<code>&i</code>	\acute{i}
<code>&'</code>	“	<code>&'</code>	”	<code>&&</code>	$\&$	<code>&a</code>	\AA	<code>&o</code>	\ddot{o}
<code>&u</code>	\ddot{u}	<code>&A</code>	\AA	<code>&O</code>	\ddot{O}	<code>&U</code>	\ddot{U}	<code>&s</code>	β

`\GLATEX` or `\GLATEX ON`

Enter the “German-LaTeX” mode, i.e. umlauts and german β can be encoded by a prefixed `"` (double quote) like in \LaTeX , i.e. `"a` for \AA etc.

To print double quotes, use two single quotes (apostrophes, `'`) like in \LaTeX and two backquotes (```): (``quoted text``), or (the german variant with lower and upper quotation marks): `"`Text"'`, double quote plus backquote and double quote plus single quote.

`\GLATEX OFF`

leaves the German- \LaTeX environment again.

Greek letters

The hashmark (`#`) toggles between latin and greek letters. Example: `#abg#` results in plotting $\alpha\beta\gamma$.

Fractions

simple fractions can be written in text strings in the form `\{a\|b\}`, which produces $\frac{a}{b}$.

Special spaces/orientation can be inserted into strings by:

- `\,` : “thin space” (0.3 SIZE)
- `\!` : “negative thin space” (-0.3 SIZE)
- `\^` : small upward displacement (superscript; only in PS fonts)
- `\v` : small downward displacement (subscript; only in PS fonts)

3.4.9 \LaTeX formulae

Any text strings (e.g. in LUN commands or axis descriptions) can contain \LaTeX formulae. This environment is started by `\(` and ended by `\)`.

The formula is inserted into the string in the current font size (the default scaling factor, $3.6 \times \text{SIZE}_{\text{LUN}}$, can be changed by setting `SCALE.LATEX`; see there) and font color. `WRplot` variables, recognized by the preceding `$` sign, can also appear in this environment and are expanded as usual.

As \LaTeX output is imported as postscript code, it cannot be displayed in the X11-window where only its bounding box is shown instead.

Example:

```
\NEXTLUN Every child knows \((c = \sqrt{a^2+b^2})\), the essence
\> of the Pythagorean theorem.
```

`\SCALE.LATEX = <factor f :real>`

The given factor (default $f = 1.0$) is multiplied to the default factor (3.6) and acts on all \LaTeX elements. Together with the Helvetica font (`\FONT=HELVET`) a factor $f = 1.25$ looks more harmonic than the default.

3.4.10 Text attributes

The text style can be changed at (mostly) any point of the string by prefixes, see Table 2 below.

Table 2: Prefixes in text strings

Prefix	Effect
<code>&T</code>	subscript (t iefgestellt)
<code>&H</code>	superscript (h ochgestellt)
<code>&M</code>	Return to normal, m iddle vertical alignment
<code>&E</code>	compact letter spacing (compressed by 30%; e ng gestellte Schrift)
<code>&B</code>	broad letter spacing (broadened by 30%; weit/ b reit gestellte Schrift)
<code>&I</code>	<i>Italics</i> font
<code>&N</code>	return to n ormal font style (inclination a nd letter spacing, switch off <code>&W</code>)
<code>&R</code>	inclination to the r ight
<code>&L</code>	inclination to the l eft
<code>&G</code>	return to uninclined font (g erade)
<code>&F</code>	“boldface” Postscript-Fonts (f ett; has no effect in the X11-window)
<code>&f</code>	quit boldface font
<code>&W</code>	quasi-bold face by double plotting (<code>WRplot</code> fonts only, use in X11 window)
<code>&i</code>	change to color $i = 0 \dots 9$
<code>&PW</code>	locally change the font to <code>WRplot</code> (only at the beginning of the string)
<code>&PT</code>	locally change the font to Times (only at the beginning of the string)
<code>&PH</code>	locally change the font to Helvetica (only at the beginning of the string)

3.4.11 Spectral line identifications

A whole set of `KASDEF` commands are especially designed to label spectral lines in a pretty and convenient way.

\IDENT *<x: units> <text: string>*

Plot an identification mark at the given position in x (i.e. wavelength) and write the *text* at that mark. The height and font size can be changed by the following commands; the default size is 0.4 and the default height is 8 cm. Note that (by default) identifications and their texts do not run into each other, i.e. get an automatic spacing.

\IDMULT *< λ_1 : units> < λ_2 : units> ... < λ_n : units> <text: string>*

This command combines n components of a blended line or multiplet transition with their respective wavelengths λ_i and a single identification *text* (the last parameter; if it contains blanks, use quotation marks, e.g. "C IV").

Settings for subsequent line identifications

\IDLENG *<length l: cm>*

Set the length of the identification marks (default 2 cm). In case the value is negative, the identification mark points down and the text is underneath. If the suffix U (i.e. 0.5U) is set, the length is taken in y -units.

\IDY *<height y: cm>*

Set the height of the identification marks over the x axis (default: 8 cm) in cm or units if suffixed with U (i.e. 1.5U).

\IDXOFF *<offset a: units>*

Shift the x position of the identifications (**\IDENT** and **\IDMULT**) by a units in x direction.

\IDXFAC *<factor f: real>*

Multiply the x position of the identification (**\IDENT** and **\IDMULT**) by f , i.e. for radial velocity or redshift. If both **IDXOFF** and **IDXFAC** are set, the total shift is calculated as follows:
 $x_{\text{ident}} = f \times \lambda + a$, where λ is the unshifted x value.

\IDSIZE *<size r: real>*

Set the font size for identifications (default: 0.4).

\IDSPACE or **ID_SPACE** *<space l: real>*

Set the minimum spacing between the identifications (basically in units of **IDSIZE**).

\IDRESET or **ID_RESET**

Disable the spacing once, i.e. identifications can run into each other.

\IDHOR or **ID_HOR**

The following identifications are written horizontally.

\IDVER or **ID_VER**

The following identifications are written vertically (default).

\IDBACKWARD or **ID_BACKWARD**

The following identifications are arranged from right to left (or downward).

\IDFORWARD or **ID_FORWARD**

The following identifications are arranged from left to right (or upward) (default).

\ID_START or **IDENT_START** *<x: units>*

The identifications' text starts at the given x value, i.e. wavelength.

\ID_CONNECT

The identifications are kept together.

\ID_NOCONNECT

The identifications are not kept together (default), i.e. remain at their given x value (i.e. wavelength).

\IDMLENG *<RESET>* or *<a:real> <b:real>*

Set the division of the identification line or *RESET* to default values of $a = 0.43$ and $b = 0.43$ (43%). The first value, a , sets the proportion of the lower part of the line, the second (b) is the middle part. The upper part gets the rest of the length, i.e. $1 - a - b$. Thus, both a and b can only range between (0, 1) and their sum must be less than 1.

\IDMCOMB *<TRUE/FALSE>*

If set *TRUE*, IDMULT does not write the identifications and just plots the comb, i.e. connects the components of IDMULT.

\IDLOG

take the logarithm of all x positions specified in subsequent identification commands; useful if spectra are plotted over $\log \lambda$

\IDNOLOG

Change to non-logarithmic x scale (default).

\ID_INBOX

Set to suppress identifications outside the x range of the plot box. It acts on `\IDENT` and `\IDMULT` (note: at IDMULT the whole set is suppressed if at least one of its components is out of the box's range).

There are some related, obsolete commands:

\LAM *<x:units> <y:cm> < Δy :cm>*

Plot a vertical line (i.e. line identification) of length Δy at position x and height y .

\TRA *<x:units> <y:cm> <size r:real> <text:string>*

Write the string *text* in vertical alignment and given size at position x and height y .

\CON *<x1:units> <x2:units> <y:cm>*

Plot a horizontal line in height y between $x1$ and $x2$ (i.e. connect the components of a multiplet).

3.4.12 Include PostScript files/figures

WRplot can include PostScript (PS) files in its PS output (in the X11-window only the BoundingBox is shown as a filled rectangular in the current color). The position and scaling of the included object is oriented – like in \LaTeX – by its BoundingBox. If neither NEXTLUN nor COORDs are given, the lower left corner of the BoundingBox will be placed at $(x, y) = (0, 0)$.

\EPSF *<filename.ps> [options]*

If no options are specified, the size, position and scaling is default (this also works if no BoundingBox is present or if it's broken).

- **COORD** *<x: units> <y: units> <x_{offset}: cm> <y_{offset}: cm>*

Place at coordinates (x, y) with offsets (see `\LUN`, Sect. 3.4.4, for details of the offset options, i.e. prefixes and their meaning).

- **NEXTLUN**

Place at the position NEXTLUN would write (set either COORD or NEXTLUN).

- **EPSFXSIZE** = *<xsize: cm>* or **EPSFYSIZE** = *<ysize: cm>*

Scale to the given width *xsize* or height *ysize* in cm or in x/y units if suffixed by U.

- **SCALE** = *<factor f: real>*

Scale by the given factor

- **ROTATE** = *<angle: degree>*

Rotate by the given angle, the center of rotation is set by the reference point of the coordinates/offsets (cf. `\LUNA`, page 23).

- **LATEX**

Try to remove the *showpage* of epsf files, i.e. files that were created by latex and dvips
-o -E *filename*.

- **NOSHOW**

Do not *include* the actual EPSF file but simulate a placeholder of same size; the positions for NEXTLUN is updated, and the variables EPS_X1 etc. are updated as if the eps file was shown. BGRLUN is shown if active.

- **New variables for coordinates:**

The box's coordinates are saved in new variables named EPS_X1, EPS_XM and EPS_X2 for the x coordinates (left, middle, right) and the same with Y1 etc. for the y coordinates of the box (all in units). An additional set of variables with suffix CM (EPS_X1CM etc.) is also created (all in cm).

3.4.13 Objects from \LaTeX code

```
\LATEX [options]
...latex-script ...
...latex-script ...
\ENDLATEX
```

The lines between LATEX and ENDLATEX are inserted into a \LaTeX document (documentstyle, default font size 12pt) between `\begin{document}` and `\end{document}`. In addition, the leading lines may contain `\usepackage{...}` instructions which are properly taken into account.

The font color is taken from the current `\COLOR`; if the color is changed within the \LaTeX code, this will affect the current color outside the LATEX-Block.

The \LaTeX code is compiled and the created eps file is included into the WRplot. By default, the position correspond to the next line of text, but can be specified alternatively by the option `COORD=xunits yunits xoffset yoffset` (same syntax as described for `\LUN`).

Note: As the X11-window cannot display the eps file, it just shows a filled rectangle of the BoundingBox's size.

The *options* following `\LATEX` are those of `\EPSF` (position, scaling, rotation). In contrast to `\EPSF` the defaults are different:

- The default position is like in `NEXTLUN` (i.e. next line) unless `COORD` is set.
- The font size is scaled to the current size (`SIZELUN`) unless another scaling is set (the created EPS file has a font size of 12 pt that is scaled by `SCALE`, `EPSFXSIZE` or `EPSFY-SIZE`).

Notes: In most cases the default options for `\LATEX` will be sufficient. Use `BGRLUN` to create a background or box around.

Hint: For inserting smaller formulae in text strings, see Sect. 3.4.9.

4 The coordinate box

The plot header describes the axes and captions/labels of a diagram. If no such coordinate box is needed, use \DEFAULTS instead.

The coordinate box is specified by a block of six lines, starting with the keyword HEADER. Their sequence is mandatory, but comment lines starting with * are allowed in between).

Example:

```
HEADER : \CENTER\Centered header caption
X-AXIS : X-Axis description
Y-AXIS : Y-Axis description
      SCALE   MIN    MAX     TICKS     LABELED   INCLUDING
X :      1.    15.    16.        1.         5.         0. [options (x)]
Y :      0.     0.    20.        1.         5.         0. [options (y)]
```

Important:

- The interpretation of lines 2 and 3 starts from character 9
- The fourth line (SCALE etc.) is only comment, but may not be omitted!
- The interpretation of lines 4 and 5 starts from character 4

Explanation of parameters:

HEADER : Headline, printed on top of the box (may be left blank)

X-AXIS : Description of the X axis (may be left blank)

Y-AXIS : Description of the Y axis (may be left blank)

By default, these three text strings appear left-justified. Optionally, if the strings starts with \CENTER\ they will be centered. Header and X-axis description are flushed right if starting with \RIGHT\, the Y-axis descriptor is flushed to the upper corner with \UP\.

Example:

```
\HEADER : \CENTER\Plot of interesting data
```

Axis specifications

The lines starting with X : and Y : (lines 5 and 6) specify details of the X-axis and the Y-axis, respectively.

AUTO / AUTOX / AUTOY

Automatic scaling allows a quick inspection of data. The scale(s) are chosen by the program such that the range of data that are to be plotted (see below) is covered. Automatic scaling can be chosen only for the x-axis, only for the y-axis, or for both. In the automatic mode, the user has no influence on the length of corresponding axis (X axis: 20cm; Y axis: 15cm), nor on its ticks and labels.

Automatic scaling is invoked by the corresponding keyword in the lines 5 and 6 as first parameter:

```
X : AUTO ...
```

both axes are scaled automatically; the Y : line is dummy;

```
X : AUTOX ...
```

only the X-axis is scaled automatically; the Y : line is valid;

Y : AUTO (or AUTOY) ...

the Y-axis is scaled automatically.

SCALE

if not AUTO (see above), the first parameter specifies the scale of this axis in cm per unit.

Note: a reversed axis requires a negative scale.

Alternatively, if this parameter ends on the letters CM (for example, 10.5CM), it gives the length of this axis in cm. If the scale parameter is 0 (zero), the length of the axis is default (X-axis: 20 cm, Y-axis: 15 cm).

MIN

value at the left / lower box boundary

MAX

value at the right / upper box boundary; note that MAX is smaller than MIN in case of a reversed axis.

TICKS

Difference between the values marked with ticks; if the axis is reversed, this number must also be negative)

LABELED

Difference between values marked with longer ticks and labels if the axis is reversed, this number must also be negative)

INCLUDING

one of the values which should appear among the labels;

[options]

optional parameters, see Table 3; especially useful in multi-plots where plots are directly backed to each other

The box is plotted with the current (i.e. of the last KASDEF commands) color, pen (line width) and font. To change the font size, see LETTERSIZE; to change the size of the ticks see TICKSIZE (Sect. 3.2).

Table 3: Options for the plot box axes

Option	Effect
ALT	place the labels and numbers of that axis at the alternate side (top or right, respectively) of the box;
MAXIND	slightly indent the last number to the left (downwards);
MAXNOIND	suppresses the default indentation of the last number if it has overhang to the right (top) of the box;
MININD	slightly indent the first number to the right (upwards);
NOLAB	suppresses the labels on this axis;
NOMAX	suppresses the last (i.e. rightmost / upmost) label;
NOMIN	suppresses the first (i.e. leftmost of the X axis) number for this axis;
NOTICK	suppresses the tick marks at the un-labeled side of the box (by default the right or upper side);
NOTICK-BOTH	suppresses the ticks at both sides of the box;

5 The plot data

5.1 Datasets

A dataset is a table with (pairs of) values. Each dataset gets a number, according to its appearance, and can be given an additional name (see COMMAND SETNAME).

Each dataset is defined by a block started by a line

N=...

and ended by any of these lines

FINISH

COMMAND INCLUDE

COMMAND INFITS

(see Sect. 5.3).

The block consists of the data and COMMANDs. Most of the COMMANDs can address another dataset (if it already exists at that point), if the COMMAND ends with DATASET=*dataset* where *dataset* is the number or name of that set.

5.2 Headline and plot attributes

N=*m* SYMBOL=*n* SIZE=*x.x* PEN=*j* XYTABLE COLOR=*i*

This is a typical headline of a dataset block. The first two characters “N=” are mandatory. All other parameters can be in arbitrary order. If absent, default values are applied.

N=*m*

Set the number *m* of following data points (i.e. pairs of *x, y* values) or set *N* = ? to read all data until FINISH is encountered. Note: In case an explicit *m* is given, there is no need for a FINISH line.

SYMBOL=*n*

(also: PLOTSYMBOL=*n*) Set the symbol for this dataset (default is 5, see Fig. 5). If

SYMBOL=0, the set will not be plotted but it can be addressed (DATASET=*dataset*, calculations, error bars etc.).

SIZE = $x.x$

(also SYMBOLSIZE) Set the symbol size (default: 0.3). If the size is negative, the area enclosed by the curve is filled (for symbols which draw a line), or the symbols are filled (in case of fillable symbols, see Appendix C).

HATCHED

acts like negative size, but applies a hatched filling of areas (cf. Sect. 3.4.3 for details).

PEN = j

Set the line width (default: 1 or the value set in PENDEF).

COLOR = i

Set the color ($i = 0 \dots 9$; default: 1 or the value set in DEFAULTCOLOR). If SYMBOL=40 (false color) is set, COLOR refers to the color table (1 to 5):

1= greyscale, 2= blackbody, 3= blue-black-red, 4= blue-white-red, 5= rainbow (see Fig. 4). Also see \NCOLORSTEP if needed (Sect. 3.4.1).

- Data formats: WRplot can read different formats of datasets – tables, sorted pairs and line-wise x and y values:

XYTABLE

Set this parameter to expect a normal table with data in columns. If XYTABLE is not set, WRplot expects a different (default) format of the data: First a line of x values, followed by a line of the same number of y values (the number of values per line is irrelevant, it just needs to be the same to allocate the x, y pairs), then a line of x values etc.

FLEX-XYTABLE

Set (alternatively to XYTABLE and SELECT) to expect the data in one line as ordered pairs, i.e. $x_1 y_1 x_2 y_2 \dots$

SELECT $\langle n_1 \rangle : \langle n_2 \rangle$

Set the ID of the columns of the data table (only if XYTABLE is set) that shall be used as x values (n_1) and y values (n_2); default: 1 : 2. Bonus: If one of the n_i is set to -1 (i.e. SELECT= -1 : 1), the values are sequentially numbered. Attention: There is no check if the given column ID is present in the data table! If it's not, WRplot will probably plot nonsense.

5.3 Data tables

Data tables can either be written inside the block (between N= and FINISH), be included from external files, or mixed. In the mixed case the data inside the block are read first, followed by the external ones.

FINISH

Finish a dataset. Alternatively to FINISH it can also be closed by including external files: COMMAND INCLUDE or COMMAND INFITS. The data table must be between the head-line (N=? ...) and the FINISH command.

COMMAND APPEND *dataset*

Append the data of *dataset* to the current dataset.

COMMAND INCLUDE *<filename> [options]*

Read the data of this dataset from the external file *filename*³. Note: The chosen data format (see above) applies here. Recursive including is allowed (i.e. the included file may include another file).

There are three basic ways the data in the included file may look:

Raw dataset: The file just contains the data points, i.e. pairs of wavelength and flux.

Normal WRplot file: The file is another WRplot file with a single dataset (i.e. starting with N=...) which is included.

Normal plot file: Like in the second case but with multiple datasets. Use the keyword INCKEY=*string* (cf. \INCLUDE in Sect. 3.2) to extract the dataset following the given *string* or/and use the keyword DATASET=*n* to include the *n*th dataset of that file. If both INCKEY and DATASET=*n* are set, take the *n*th dataset following the *string*, instead. This allows to extract datasets from plots with several datasets in one plot.

In any case an additional FINISH is not needed. Either the included file has a FINISH/END or the E-O-F of that file is allowed as end of the dataset.

COMMAND INFITS: *<fitsfilename> [options]*

Include data from the fits file *fitsfilename*⁴.

Fits files can have various structures and formats, and WRplot does not cover all possibilities. However, so far we managed to read any FITS formats provided by the observatories and data archives, except of three-dimensional data-cubes from integral-field spectrographs like MUSE.

INFITS reads a vector of data into the Y-values of the dataset. The X-values are by default just the index numbers. If other X-values, for instance a vector of wavelengths, is stored in another row or column in the FITS file, these data must be read subsequently in a second dataset as its Y-values, and then transferred to the X-values of the first dataset using COMMAND WAVECAL, see page 39.

However, a frequent exception is made by the following conversion. If the HEADER of this data/header unit (HDU, see below) gives the parameters CRVAL1 and CDEL1 (and, optionally, CRPIX1), it is assumed that the wavelength scale is equidistant with increment CDEL1 and starting point CRVAL1 at pixel number CRPIX1, with the latter having the default value of 1 if not given. In the described case, the X-vector is filled with

$$X(I) = CRVAL1 + (I - CRPIX1) * CDEL1$$

with I being the pixel index. In some fits files, CD1_1 is used instead of CDEL1, which is also recognized by COMMAND INFITS. Note that header parameters can also be read in WRplot with the backslash-instruction \FITSVAR, see page 15.

Options:

³for allowed pathnames and zipped files see description of \INCLUDE on 14

⁴for allowed pathnames and zipped files see description of \INCLUDE on 14

- **INFO**
displays in the terminal some information extracted from the header of the first HDU ("header/data unit");
- **SHORT-INFO**
displays in the terminal some short information extracted from the header of the first HDU ("header/data unit");
- **NHDU= n**
refers the reading actions to the HDU number n ;
- **ROW= n**
select row n from a 2-D table (e.g. a CCD image);
- **ROWS= $m\ n$**
select rows m to n from a 2-D table (e.g. a CCD image) and add their values up into the Y-vector;
- **BINTABLE <TFIELD for X> <TFIELD for Y>**
specifies that the current HDU contains a binary table; this option must be followed by exactly two names as assigned to columns in the BINTABLE that shall be read into the X and the Y values of the WRplot dataset, respectively.
One of the two field names might be the placeholder @N, meaning that the corresponding values (X or Y, respectively) are just the running index of that entry.
A typical INFITS command would be:
`COMMAND INFITS <fitsfile> NHDU=2 BINTABLE WAVELENGTH FLUX`
Note that the field names can be different in different FITS files. If a requested field name is not defined in the respective HDU, an error stop will occur, after a list of all available field names that are defined in this HDU is displayed on the terminal.
If the X-values are defined by the @N placeholder, the running index can be transformed into an equidistant (wavelength-) scale by setting the two options NSTART and DN (both!):
- **NSTART= x**
sets the first X-value (starting wavelength); similar to CRVAL1, but now for a binary table;
- **DN= Δx**
sets the increment between subsequent X-values; similar to CRVAL1, but for a binary table;
Example generating an equidistant λ scale:
`COMMAND INFITS <fitsfile> NHDU=2 BINTABLE @N FLUX NSTART= λ_1 DN= $\Delta\lambda$`
- **NROW= n**
select row n in a binary table;

Further examples without BINTABLE):

`COMMAND INFITS filename.fits NHDU=2 ROW=7`
extracts from a 2-D image in the second HDU the 7th row;

COMMAND INFITS filename.fits NHDU=2 ROWS= 7 15

extracts from a 2-D image in the second HDU the rows 7 to 15 and adds the values into the Y-vector; these features can be used for simple extractions, background correction etc. from spectrographic CCD images;

COMMAND INFITS ~wrh/obsdat.dir/CPN-WR23.UVES.blue10.fits NHDU=2 ROW=1
reads a ESO-VLT UVES spectrum including wavelength scale which is implied by the header parameters CRVAL1, CDEL1, and CRPIX1).

5.4 COMMAND lines

Some COMMANDs are executed (or set parameters) when they are read: APPEND, X-RANGE, Y-RANGE, SKIP, SETNAME, INCLUDE, INFITS. They cannot be skipped by IF constructions (see below). All other COMMANDs are first read and put on a stack before they are applied on the complete dataset in their order of appearance.

COMMAND SETNAME <name:string>

Set a *name* for the current dataset to address it (i.e. for APPEND, ARI) either by this name or its sequential number.

COMMAND X-RANGE: < x_1 :units> < x_2 :units>

Ignore all points during the read-in phase, if their x coordinate is not in the interval $[x_1, x_2]$.

COMMAND Y-RANGE: < y_1 :units> < y_2 :units>

Ignore all points during the read-in phase, if their y coordinate is not in the interval $[y_1, y_2]$.

COMMAND X-CUT: < x_1 :units> < x_2 :units>

Delete all points of the dataset, if their x coordinate is not in the interval $[x_1, x_2]$.

COMMAND Y-CUT: < y_1 :units> < y_2 :units>

Delete all points of the dataset, if their y coordinate is not in the interval $[y_1, y_2]$.

COMMAND X-OMIT: < x_1 :units> < x_2 :units>

Delete all points of the dataset, if their x coordinate is in the interval $[x_1, x_2]$ (negation of X-CUT).

COMMAND Y-OMIT: < y_1 :units> < y_2 :units>

Delete all points of the dataset, if their y coordinate is in the interval $[y_1, y_2]$ (negation of Y-CUT).

COMMAND XY-SWAP

Swap the x and y values of the current dataset.

COMMAND WAVECAL <dataset i >

The y values of the current dataset are taken as x values for dataset i (i.e. it must already exist at that point). This requires both sets to have identical x values before.

Especially for wavelength calibration the current dataset may have a different set of x values, i.e. just a few support points of calibration lines. They need to be in monotonic order, at least. COMMAND WAVECAL then interpolates between them (splines) and/or extrapolates outside (linear with the first/last two values of the table) to obtain the set of x values as needed for dataset i .

COMMAND SKIP *<i>*

Skip the current (or the last *i*) dataset(s), i.e. do not plot it/them and delete it/them after all COMMANDs were performed. Note: The sequential ID for following datasets is reduced by 1 (or *i*), too.

COMMAND EXPAND *<N>* [*< λ_1 :units>* *< λ_2 :units>*]

or

COMMAND EXPAND-SPLINE *<N>* [*< λ_1 :units>* *< λ_2 :units>*]

Expand the dataset by *N* points (between λ_1 and λ_2 if they are set) by linear interpolation of cubic spline interpolation, respectively.

COMMAND INSERT *< λ_1 :units>*

Insert a point at $x = \lambda_1$ (always by linear interpolation between two existing points).

COMMAND HLYMAN *parameter* (also HLYMANA)

Apply corrections for interstellar absorption by hydrogen Lyman lines (from Ly α to Ly $_{20}$). The format of *x, y* data is mandatory: *x* values are expected to be wavelengths in Å and *y* is expected to be the linear flux (i.e. in $\text{erg s}^{-1} \text{cm}^{-2} \text{Å}^{-1}$).

This command *corrects* the observation by absorption (removes it); to add the absorption to a model flux, one needs COLDENS or EBV as *negative* value. In case EBV is specified, this is converted into the column density as $N_{\text{H}} = 3.8E21 * E_{B-V}$ (Groenewegen & Lamers 1989).

By default, HLYMAN uses the formalism from Groenewegen & Lamers (1989) which accounts only for the damping (i.e. Lorentzian) profiles but no Doppler cores. However, if one of the parameters T and/or VTURB is specified, HLYMAN applies Voigt profiles with the according Doppler velocity.

The parameters consist of pairs “keyword = value”; allowed keywords are:

Parameter	Description
EBV= <i>x.x</i>	Color excess E_{B-V} , in magnitudes
COLDENS= <i>x.x</i>	Column density of H-atoms per cm^2 (N_{H})
VRAD= <i>x.x</i>	Radial velocity, in km/s
T= <i>x.x</i>	Temperature in Kelvin; to additionally account for Doppler broadening with the thermal velocity (Maxwellian?)
VTURB= <i>x.x</i>	Turbulence velocity in km/s causing additional Doppler broadening

One of the parameters (EBV or COLDENS) is mandatory.

Example:

COMMAND HLYMANA EBV=0.2 VRAD=255.

For compatibility with older syntax, one can use numerical parameters without keyword (but this is not recommended):

- only one parameter is interpreted as the value of EBV
- two numerical parameters are interpreted as EBV and VRAD

COMMAND ISMLINE *parameter*

Correction or simulation of an interstellar line; the command is similar to HLYMAN, but now

for an arbitrary line which is fully specified by the command. This command *corrects* the observation by absorption (removes it); to add the absorption to a model flux, one has to specify COLDENS as *negative* value.

The following parameters are mandatory:

Parameter	Description
COLDENS = $x.x$	Column density of the absorbing atoms per cm^{-2}
L0 = $x.x$	Central wavelength of the ISM line in \AA
GAMM = $x.x$	Damping constant for this transition in s^{-1}
F = $x.x$	Oscillator strength

If the following three parameters are also specified, the synthetic profile will also include a Doppler core and produce a Voigt profile:

Parameter	Description
T = $x.x$	Temperature in Kelvin; additionally accounts for Doppler broadening with the thermal velocity
A = $x.x$	Atomic weight in Atomic Mass Units
VTURB = $x.x$	Turbulence velocity in km/s causing additional Doppler broadening
Further optional parameter:	
VRAD = $x.x$	Radial velocity in km/s

COMMAND REDDENING $\langle E_{(B-V)}:real \rangle [law] [R_V]$

Apply a correction for interstellar reddening. In contrast to HLYMAN, both the x and y values are now expected to be logarithmic ($\log[\lambda/\text{\AA}]$ and $\log f$, respectively). The x values must be in monotonic order and are automatically expanded to 1000 points if there are less. If no *law* is given, the default is used (Seaton 1979).

Table 4: Reddening laws as for the different Keyword

Keyword	Ref. ^a	default R_V	Comment
SEATON (or none)	S, N	3.1	default in the range $911 \text{\AA} - 1 \mu\text{m}$; established by Seaton for $\lambda < 3700 \text{\AA}$, and augmented by Nandy et al. for $3500 < \lambda < 10000 \text{\AA}$
CARDELLI	C	3.1	established for $1000 \text{\AA} - 3.33 \mu\text{m}$, used here for $911 \text{\AA} - 2.44 \mu\text{m}$
FITZPATRICK	F	3.1	established from $1150 \text{\AA} - 5 \mu\text{m}$, used here for $911 \text{\AA} - 4 \mu\text{m}$
LMC	H	3.2 (fixed)	special for the LMC
SMC	H	2.7	R_V from Bouchet et al. (1985) with $R_V = 3.2$ it becomes the LMC law
SMC_BAR	G	2.74	for $\lambda < 3000 \text{\AA}$; to the red: FITZPATRICK with $R_V = 2.74$ for $\lambda > 5 \mu\text{m}$ it follows Nandy et al. (1975)

References: C: Cardelli et al. (1989); F: Fitzpatrick (1999); G: Gordon et al. (2003); H: Howarth (1983); N: Nandy et al. (1975) S: Seaton (1979)

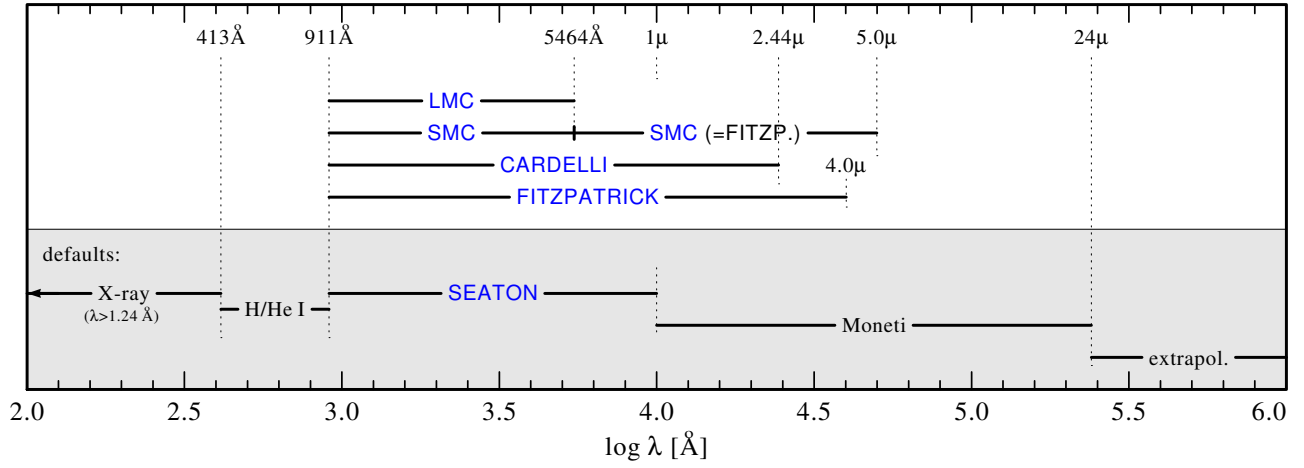


Figure 1: Ranges of different reddening laws in dependence from the keyword

Note: $R_V = A_V/E_{B-V}$, where A_V is the extinction in the V band. All reddening laws are extended towards the red end (outside their “range”) according to a table from Moneti et al. (2001), above $\lambda > 24\mu\text{m}$ an extrapolation $\propto \lambda^{-2}$ is extended.

In the X-ray regime ($\lambda = 1.24 - 413\text{Å}$) the absorption is calculated according to Table 2 of Morrison & McCammon (1983). The edges in this wavelength range arise due to the K-shell absorption of different elements (a solar composition is assumed). In the range $\lambda = 413 - 504\text{Å}$ (the He I edge) these tabulated polynoms are extrapolated.

In the range of $\lambda = 504 - 911\text{Å}$, i.e. between the Lyman edge of hydrogen and the He I edge, only the opacity of H I is considered in the same way as in the PoWR code (including the Gaunt factor).

COMMAND CONVOL: <Keyword> <C>

Convolve the dataset by the function given as *keyword* and parameter *C*:

Keyword	Kernel function	Parameter
GAUSS	Gaussian	FWHM
BOX (or KASTEN)	box profile	half box width
ROT[ATION]	half ellipse ¹	$\Delta\lambda_{\text{rot}} = \lambda v_{\text{rot}}/c$
MACRO-RT	radial-tangential macroturbulence profile ²	$\Delta\lambda_{\text{macro}} = \lambda v_{\text{macro}}/c$

¹ additional parameter (optional): limb-darkening coefficient $\text{BETA}=\beta$ (Unsöld 1968). Definition: $I(\mu)/I(0) = (1+\beta)^{-1}(1+\mu\beta)$. Limb-darkening coefficients β can be calculated with PoWR. A typical value is 1.5

² Gray (1975)

Example (see also the example plots):

COMMAND CONVOL ROT=0.2 BETA=1.5

COMMAND BINNING *w*

This command requires the dataset to be monotonic in x . Starting with the first value, all

points with x values differing $\leq w$ ($\forall j$ with $x_i + w > x_{i+j}$) are averaged. Each new point $(x_{i'}, y_{i'})$ is the arithmetic mean of all x_{i+j}, y_{i+j} that were averaged.

COMMAND X+ <A: units> (also COMMAND XADD or COMMAND XOFFSET)

Add the constant A to the x values of the dataset, i.e. shift the x coordinate by A . Note: Between X+ and A there must be a blank (this applies to the rest, too!).

COMMAND X- <A: units> (also COMMAND XSUB)

Subtract the constant A from the x values of the dataset.

COMMAND Y+ <A: units> (also COMMAND YADD or COMMAND YOFFSET)

Add the constant A to the y values of the dataset.

COMMAND Y- <A: units> (also COMMAND YSUB)

Subtract the constant A from the y values of the dataset.

COMMAND X* <A: units> (also COMMAND XMULT)

Multiply the constant A to the x values of the dataset.

COMMAND X/ <A: units> (also COMMAND XDIV)

Divide the x values of the dataset by the constant A .

COMMAND Y* <A: units> (also COMMAND YMULT)

Multiply the constant A to the y values of the dataset.

COMMAND Y/ <A: units> (also COMMAND YDIV)

Divide the y values of the dataset by the constant A .

Note: Instead of a constant A one can also write **X** or **Y** in the above COMMANDs, i.e. to multiply the flux by the wavelength.

COMMAND XINV or COMMAND 1/X

Invert the x coordinates of the dataset.

COMMAND YINV or COMMAND 1/Y

Invert the y coordinates of the dataset.

COMMAND XDEX or COMMAND 10**X

Delogarithmize (un-log) the x values of the dataset.

COMMAND YDEX or COMMAND 10**Y

Delogarithmize (un-log) the y values of the dataset.

COMMAND XLOG or COMMAND LOG(X)

Logarithmize the x values of the dataset.

COMMAND YLOG or COMMAND LOG(Y)

Logarithmize the y values of the dataset.

COMMAND XMIN = x.x

Set all x values of the dataset with $x \leq x.x$ to that value. In an analogous way the following three COMMANDs exist:

COMMAND XMAX = x.x

COMMAND YMIN = x.x

COMMAND YMAX = x.x

COMMAND Y-NORM < x_1 :units> < y_1 :units>

or

COMMAND Y-NORM* < x_1 :units> < y_1 :units>

Normalize the curve (y values) by the given point (x_1, y_1). This command finds the interval around x and adjusts the y values of the dataset by addition (or multiplication, respectively) such, that the given point (x_1, y_1) becomes part of the curve, i.e. normalizes the curve.

COMMAND SIGN(X)

Replace the x values by $-1., 0., +1.$, according to the signum function.

COMMAND SIGN(Y)

Replace the y values by $-1., 0., +1.$, according to the signum function.

COMMAND X-INC

Delete all points of the dataset if their x values are not monotonically increasing. This can help to work with imported data, i.e. IUE data, that do not necessarily follow the “strict german purity laws for nice spectra”. Certain COMMANDs (like CONVOL) need monotonically sorted x values.

COMMAND ARI <dataset 1> <operator> <dataset 2> [KEEP_X1]

Execute the operation given as *operator* (+, -, *, /) on the y values of the given datasets. The result is saved in *dataset 1*. If the x values are identical for both datasets, the y values are assigned pair-wise, else they are interpolated in the overlapping region. If either x set is nonmonotonic the command returns an Error message.

The optional parameter KEEP_X1 disables the interpolation of x data, i.e. the y values are only calculated at these supporting points. The saved values for *dataset 1* are only at these points, then.

COMMAND STRIP < y_1 :units> < y_2 :units>

Set the strip (between y_1 and y_2) in which the false-color representation is plotted. The x values are taken from the dataset, i.e.

```
N=2 XYTABLE SYMBOL=40 SELECT=1:1 COLOR=5
COMMAND STRIP= 0.5 1
COMMAND EXPAND 1000
0
2
FINISH
```

plots a rainbow colored strip (COLOR 5) between $x = [0, 2]$ and $y = [0.5, 1]$.

COMMAND XERROR <dataset n > <type:PLUSMINUS/MINMAX>

Assign the dataset n as error data in x to the current dataset, i.e. the dataset n must be defined before the current one. The mandatory parameter *type* sets the type of error data:

relative to the data point (PLUSMINUS); or absolute (MINMAX).

Note: The dataset n must have SYMBOL=0, i.e. not be plotted directly, and must either consist of a *single* pair of data (to have that error bar for the whole dataset); or have the same number of data points as the current set (individual error bars for each x value). The style (COLOR, PEN, SIZE) of the error bars is taken from the headerline of the dataset n . SIZE characterizes the size of the error bar's footer (the small perpendicular bars). Note: To get error bars in both directions, i.e. $x_{-0.7}^{+0.5}$, the dataset n must contain both numbers as two columns, e.g. 0.5 as x value and -0.7 as y value.

COMMAND YERROR <n> <type>

Like XERROR, but for errors in y (to have independent error bars for either/both uncertainties).

COMMAND MERGE <dataset1> <dataset2>

Merge two (ordered) datasets into *dataset1*. In overlapping intervals the arithmetic mean is taken (by interpolation where needed, i.e. if the x values are not identical). This command is especially useful to merge overlapping spectra, like the different Echelle orders.

COMMAND SORT [INV] [JOIN]

Sort the dataset such, that the x values will be in monotonic order. With the optional parameter INV given, the dataset is sorted in a falling sequence. With the optional parameter JOIN given, the dataset becomes *strictly* monotonic. Entries with identical x values are joined into one entry, with the y value set to their arithmetic mean. This command is useful in preparing datasets for operations that require a monotonic order.

COMMAND WRITE [FILE=<filename:string>]

Write the current dataset (in its state at the line COMMAND WRITE is given) as an XYTABLE into the given file (overwrites an existing file). If no FILE=*filename* is given, the output file will be named WRPLOT_DATASET n .DAT, where n is the sequential number of that dataset.

COMMAND kasdef-kommando, i.e. IF, ELSE, ENDIF

Furthermore, many commands described in the KASDEF sections can be placed into the COMMAND block, excluding those which plot/write text (from ARR to LUN and the TEXT options). But commands that handle variables, i.e. COMMAND CALC, conditional constructions (IF, ELSE etc.), DO loops, ECHO, GOTO and LABEL are allowed, i.e. most of the *instructions* described in Sect. 3.3.

5.5 COMMAND-Functions

COMMAND CF-... ..

WRplot has some *Command-Functions*. These functions calculate different things and save the result in variables that can be used in following COMMANDs or KASDEFs. Currently there are:

COMMAND CF-YEXT < λ :units> <VAR>

Interpolate the y value at $x = \lambda$ and save it in the variable VAR. If SYMBOL=6 or 7, the interpolation will be splines, else linear.

COMMAND CF-XMIN <XVAR> <YVAR>

Find the point with $x = \min(x_{\text{dataset}})$ and save that point's coordinates in variables *XVAR*, *YVAR*.

COMMAND CF-XMAX <XVAR> <YVAR>

Find the point with $x = \max(x_{\text{dataset}})$ and save that point's coordinates in variables *XVAR*, *YVAR*.

COMMAND CF-YMIN <XVAR> <YVAR>

Find the point with $y = \min(y_{\text{dataset}})$ and save that point's coordinates in variables *XVAR*, *YVAR*.

COMMAND CF-YMAX <XVAR> <YVAR>

Find the point with $y = \max(y_{\text{dataset}})$ and save that point's coordinates in variables *XVAR*, *YVAR*.

COMMAND CF-YSUM <VAR>

Sum up all *y* values of the dataset and save the result in the variable *VAR*.

COMMAND CF-INT <VAR> [FROM <*x*₁>:units> TO <*x*₂>:units>]

Integrate the tabulated function and save the result in the variable *VAR*. The function just needs to be monotonic. By default the integration is done over all *x* values, however, one can set the integration limits *x*₁ and *x*₂ to integrate $VAR = \int_{x_1}^{x_2} y(x)dx$ (both must be within the range of *x* values, i.e. no extrapolation).

COMMAND CF-LINREG [*x*₁ *y*₁ *x*₂ *y*₂ *a* *b* *a*_{err} *b*_{err}]

Fit the dataset by a linear regression curve $y_r = a + bx$. Note: One can also fit power laws if the dataset is logarithmic. There are up to eight variables for CF-LINREG, which can get their values by this COMMAND function. Their meaning is:

*x*₁ = smallest *x* value of the dataset

*y*₁ = value of *y*_r at that point

*x*₂ = largest *x* value of the dataset

*y*₂ = value of *y*_r at that point

These four variables are most helpful to plot a regression curve using

`\LINUN $x1 $y1 $x2 $y2 0 0`

though the data points.

Further optional variables contain the values of:

a = coefficient *a* of the above regression function

b = coefficient *b* of the above regression function

*a*_{err} = formal uncertainty (1σ) of *a*

*b*_{err} = formal uncertainty (1σ) of *b*

If the dataset has error bars in *y* (i.e. YERROR has been used), the points are weighted accordingly. Note that the coefficients *a*, *b* are calculated by different means if error bars are present (cf. Numerical Recipes), especially the formal errors *a*_{err}, *b*_{err} can be *smaller* if the data have error bars.

All these parameters are also printed in the command window/Terminal.

COMMAND CF-NDATA *<VAR>*

Write the number of data points (i.e. number of (x, y) pairs) in the variable *VAR*.

6 Creating WRplot files with `FORTRAN` programs

The famous `FORTRAN` subroutines `PLOTANF` and `PLOTCON` can create ordinary WRplot files. To have some more specialized attributes, i.e. in the headline (`PEN=n COLOR=i SIZE=k`), use the routines `PLOTANFS` and `PLOTCONS`. These routines can e.g. be found in the WRplot installation directory in the library/archive *lib_plotcalls*. Find the parameters in the respective *.f* files (i.e. `~wrh/wrplot.dir/fortran.dir/`).

KASDEF options can also be called from within `FORTRAN` programs:

- a) Start the plot by hand, i.e. write the initial PLOT line:
`WRITE (KANAL,*) 'PLOT: Text ...'`
(this line just starts and names the plot, i.e. to get orientation in interactive usage).
- b) Write the KASDEF options as string, e.g.
`WRITE (KANAL,*) '\LUN'`
- c) Use `PLOTANF` to create the plot box (HEADER: etc.). It writes another PLOT line which is ignored by WRplot (*dummy* line).

7 Include plots into TeX documents

The PostScript files created by WRplot can easily be included into \LaTeX files, i.e. write:

- a) At the beginning:
`\usepackage{epsf}`
- b) Inside the figure environment (after `\begin{figure}`):
`\epsffile{filename.ps}`
to include the given PostScript file in original size (1:1), left-justified at the (current) position.
- c) To have centered figures, write:
`\centering`
`\mbox{\epsffile{filename.ps}}`
- d) To scale the included PostScript file use:
 - (a) Full line measure:
`\epsfxsize=\textwidth`
`\epsffile{filename.ps}`
 - (b) 70% of the line measure, centered:
`\centering`
`\epsfxsize=0.7\textwidth`
`\mbox{\epsffile{filename.ps}}`
 - (c) Alternatively one can set the width/height in cm:
`\centering`
`\epsfysize=8.0cm`
`\mbox{\epsffile{filename.ps}}`
(or `\epsfxsize=10.0cm` for the width)

Attention: In two column style (A&A etc.) `\textwidth` is the *full* width. To have a one-column figure, set `\columnwidth` instead.

References

- Bouchet, P., Lequeux, J., Maurice, E., Prevot, L., & Prevot-Burnichon, M. L. 1985, A&A, 149, 330
Cardelli, J. A., Clayton, G. C., & Mathis, J. S. 1989, ApJ, 345, 245
Fitzpatrick, E. L. 1999, PASP, 111, 63
Gordon, K. D., Clayton, G. C., Misselt, K. A., Landolt, A. U., & Wolff, M. J. 2003, ApJ, 594, 279
Gray, D. F. 1975, ApJ, 202, 148
Groenewegen, M. A. T. & Lamers, H. J. G. L. M. 1989, A&AS, 79, 359
Howarth, I. D. 1983, MNRAS, 203, 301
Moneti, A., Stolovy, S., Blommaert, J. A. D. L., Figer, D. F., & Najarro, F. 2001, A&A, 366, 106
Morrison, R. & McCammon, D. 1983, ApJ, 270, 119
Nandy, K., Thompson, G. I., Jamar, C., Monfils, A., & Wilson, R. 1975, A&A, 44, 195
Seaton, M. J. 1979, MNRAS, 187, 73P
Unsöld, A. 1968, Physik der Sternatmosphären

Acknowledgements

Many people have contributed over the years to the development of WRplot: Gerhard Dünnebeil, Ulf Wessolowski, Uwe Leuenhagen, Lars Koesterke, Helge Todt, Martin Steinke (list incomplete)

The english translation of this manual has been initiated by Martin Steinke.

A Appendix: Examples for fonts

<p>WRPLOT Font</p> <p>!"\$%\'()*+,-./0123456789:;<=>?@ ABCDEFGHIJKLMNOPQRSTUVWXYZ []^_`abcdefghijklmnopqrstuvwxyz äöüíÄÖÜß&#;" ∅ Å * Ñ ∞ → ← ± \ •</p>	<p>TIMES Fonts</p> <p>!"\$%\'()*+,-./0123456789:;<=>?@ ABCDEFGHIJKLMNOPQRSTUVWXYZ []^_`abcdefghijklmnopqrstuvwxyz äöüíÄÖÜß&#;" ∅ Å * Ñ ∞ → ← ± \ •</p>
<p>HELVETICA Fonts</p> <p>!"\$%\'()*+,-./0123456789:;<=>?@ ABCDEFGHIJKLMNOPQRSTUVWXYZ []^_`abcdefghijklmnopqrstuvwxyz äöüíÄÖÜß&#;" ∅ Å * Ñ ∞ → ← ± \ •</p>	<p><i>ZAPF Fonts (Zapf-Chancery medium italic)</i></p> <p>!"\$%\'()*+,-./0123456789:;<=>?@ <i>ABCDEFGHIJKLMN O P Q R S T U V W X Y Z</i> []^_`abcdefghijklmnopqrstuvwxyz äöüíÄÖÜß&#;" ∅ Å * Ñ ∞ → ← ± \ •</p>
<p>COURIER Fonts</p> <p>!"\$%\'()*+,-./0123456789:;<=>?@ ABCDEFGHIJKLMNOPQRSTUVWXYZ []^_`abcdefghijklmnopqrstuvwxyz äöüíÄÖÜß&#;" ∅ Å * Ñ ∞ → ← ± \ •</p>	<p>WRPLOT Symbol Font (# switch)</p> <p>" !"\$%\'()*+,-./0123456789:;<=>?@" ABXΔΕΦΓΗΨΚΛΜΝΟΠΘΡΣΤ~≈ΩΞΤΖ []Δ αβχδεφγηηψκλμνοπθρστ≈ωξυζ</p>
<p>PS Symbol Font (# switch)</p> <p>∇ !∇%\'()*+,-./0123456789:;<=>?≡∇ ABXΔΕΦΓΗΘΚΛΜΝΟΠΘΡΣΤΥζΩΞΨΖ []⊥_ ᾱβχδεφγηηφκλμνοπθρστυωξψζ</p>	<p>PS Italic Font (&I switch)</p> <p>" !"\$%\'()*+,-./0123456789:;<=>?@" <i>ABCDEFGHIJKLMN O P Q R S T U V W X Y Z</i> []^_`abcdefghijklmnopqrstuvwxyz</p>
<p>Text attributes (&-Switches)</p> <p>&H: ^{superscript} &M: normal &T: _{subscript} &H...&T (directly following): ^{upper} _{lower} &E: compact &B: broad &W: quasi-boldface (X11 window only) &N: regular &F: boldface &f: normal font weight (both only PS output) &R: <i>inclination to the right</i> &L: <i>inclination to the left</i> &G: uninclined Fractions: \{ numerator \} \{ denominator \} → $\frac{\text{numerator}}{\text{denominator}}$</p>	

Figure 2: The fonts available in PostScript

B Appendix: Examples for colors



Figure 3: The predefined colors: 0 ... 9; 0 is white

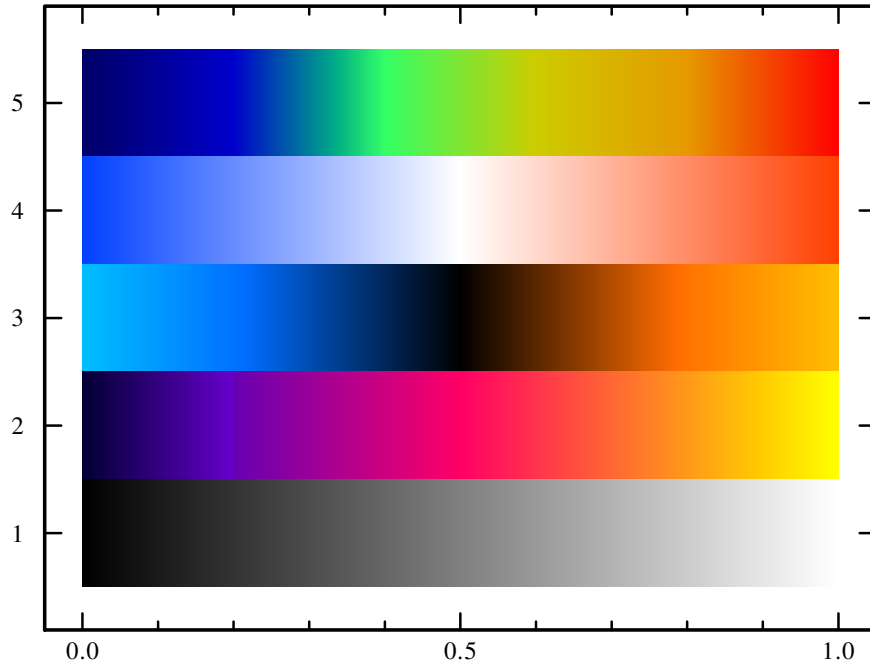


Figure 4: The defined color tables (COLOR=1 – 5) for the false-color plotting (SYMBOL=40): 1= greyscale, 2= blackbody, 3= blue-black-red, 4= blue-white-red, 5= rainbow

C Appendix: The Plot symbols

See Fig. 4!

- 0:** Nothing plotted
- 1:** Cross
- 2:** Asterisk
- 3:** Triangle, upright (fillable)
- 4:** Square, axis-parallel (fillable)
- 5:** Polygon (SIZE is redundant; if negative, the encircled area is filled)
- 6:** Cubic Spline Interpolation, solid line (also works for non-injective mappings). SIZE gives the spacing of the interpolated points in cm; negative SIZE: fill the encircled area).
- 7:** Cubic Spline Interpolation, dashed (also works for non-injective mapping). SIZE specifies both, the spacing of the interpolated points and the length of the dashes (in mm).
- 8:** Circle, negative SIZE fills the symbol. The diameter is $2/3 \times \text{SIZE}$ and fits harmonically to the size of the other discrete symbols.
- 9:** Polygon, dashed (SIZE specifies the length of the dashes)
- 10:** Polygons, dash-dotted (SIZE specifies the length of the dashes)
- 11:** Triangle, upside-down (fillable)
- 12:** Square, standing on corner (fillable)
- 13:** Bomb (fillable) - do not abuse !
- 14:** Star with three spikes (similar to the Daimler-Benz star) (fillable)
- 15:** Rhomb (diamond, lozenge) (fillable)
- 16:** Vertical dash, centered, SIZE specifies the length
- 17:** Rhomb (like SYMBOL=15), but lying (fillable)
- 18:** Star with five spikes (fillable)
- 19:** Cubic Spline with sinusoidal modulation; SIZE regulates the amplitude and wavelength of the sine (i.e., they cannot be chosen independently)
- 20:** Polygons, dashed, gaps of double length (SIZE is the length of the dashes in cm)
- 21:** String of beads with crosses (SYMBOL 1). SIZE sets both their distance and size
- 22:** String of beads with asterisks (SYMBOL 2). SIZE sets both their distance and size
- 23:** String of beads with triangles (SYMBOL 3). SIZE sets both their distance and size, negative SIZE fills the symbols

- 24:** String of beads with squares (SYMBOL 4). SIZE sets both their distance and size, negative SIZE fills the symbols
- 25:** String of beads with triangles of their tip (SYMBOL 11). SIZE sets both their distance and size, negative SIZE fills the symbols
- 26:** Cross (fillable)
- 27:** St Andrew's cross (fillable)
- 28:** String of beads with circles (SYMBOL 8). SIZE sets both their distance and size, negative SIZE fills the symbols
- 31:** String of beads with triangles of their tip (SYMBOL 11). SIZE sets both their distance and size, negative SIZE fills the symbols – (redundant to SYMBOL 25).
- 32:** String of beads with squares of their tip (SYMBOL 12). SIZE sets both their distance and size, negative SIZE fills the symbols
- 35:** Histogram, solid lines; negative SIZE fills the area underneath
- 36:** Histogram, dashed, negative SIZE fills the area underneath
- 40:** False color representation. The y values of the dataset are encoded in false colors (COLOR 1–5, see Fig. 4) and plotted in an horizontal strip (cf. COMMAND STRIP for details).

By default, the color table is scaled to the minimum and maximum of the dataset, unless specified otherwise: WRplot has *global* variables *MIN_COL* and *MAX_COL* that influence the part taken from the color table (so they need to be specified in the INSTRUCTION-EXPORT block, see page 54).

The color table is scaled to the minimum and maximum of the dataset by default unless specified otherwise: However, the user can define the variables *MIN_COL* and *MAX_COL* (within an INSTRUCTION EXPORT block) in order to fix these values globally, as shown in the following example:

```
\INSTRUCTION EXPORT
\VAR MIN_COL = 123.4
\VAR MAX_COL = 5000
\INSTRUCTION NO-EXPORT
```

Effect: the color table starts at $y = 123.4$ and ends at $y = 5000$. y -values smaller than *MIN_COL* are coded with the color of *MIN_COL*, while y -values larger than *MAX_COL* are coded with the color of *MAX_COL*,

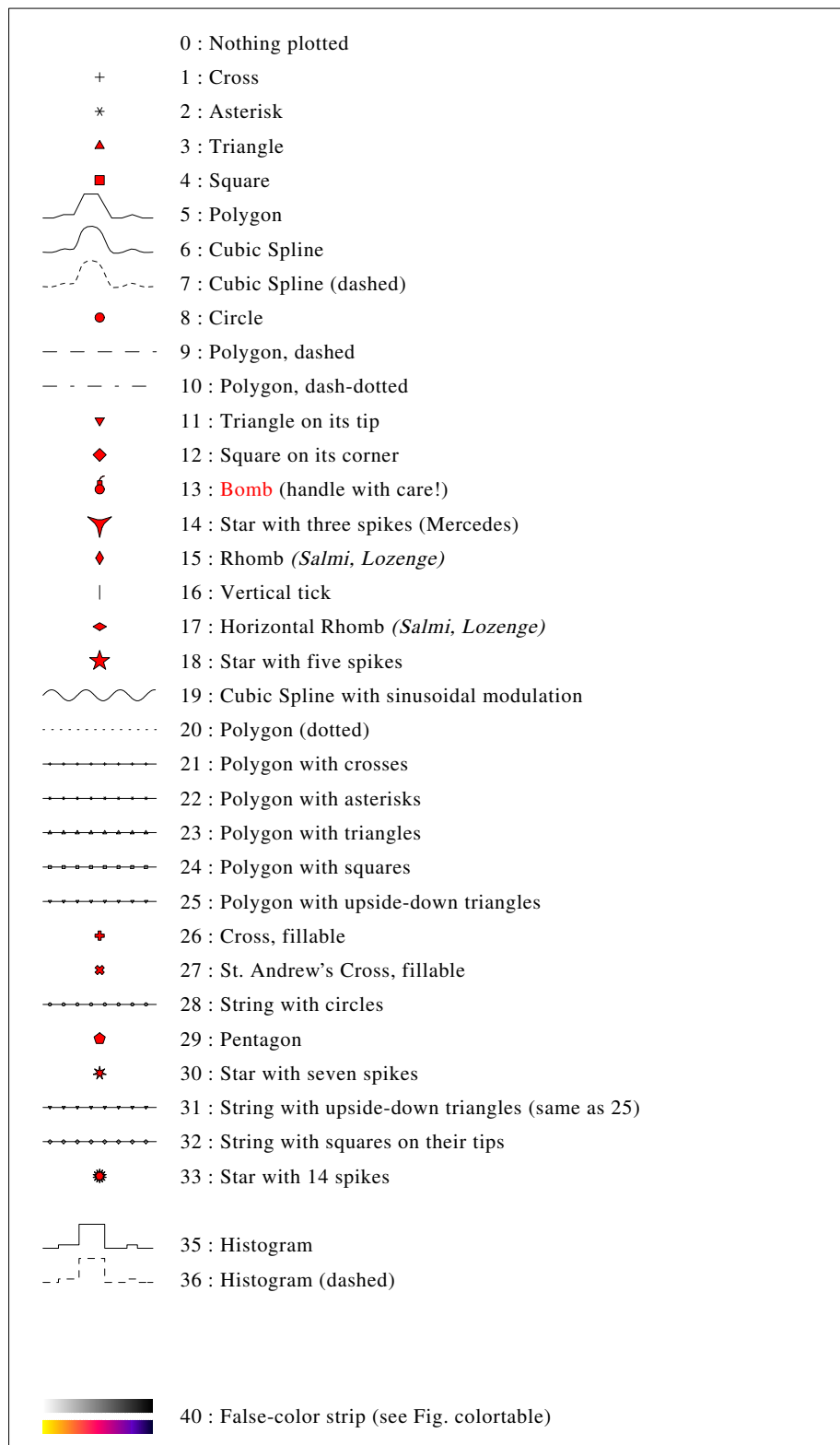


Figure 5: The available symbols for plotting, fillable discrete symbols are painted red (fill with a negative SIZE value). Note: Symbols 5 to 7, and 35+36 show the same dataset in this example.

Index

- LaTeX, 27, 31
- Arc of a circle, 21
- Arithmetic (Variables), 16
- Arithmetics (Datasets), 44
- ARR, arrow, 21
- ASCII-to-PostScript, 7
- AUTO box scaling, 33
- BBROTATE, 10
- BGRLUN (text background), 25
- BINNING, 42
- Bounding-Box, 10
- CALC (Arithmetics), 16
- CF/COMMAND functions, 45
- Circles (arc of), 21
- Colors, 20
- Continuation lines, 11
- Convolution, 42
- Coordinate Box, 33
- CUTVAR (string), 17
- Datasets, 35
- Default pen, 13
- DEFAULTCOLOR, 13
- DEFAULTS, 13
- Define colors, 20
- DO-Loops, 18
- ECHO, 15
- Ellipse plotting, 21
- EPSF-Files (include), 30
- Error bars (data), 44
- Error bars (KASDEF), 22
- EXPAND dataset, 40
- EXPR (connect strings), 17
- FILLING – datasets, 36
- FILLING – KASDEF, 23
- FITS files, 37
- FITSVAR, 15
- Fonts, 19
- FORMAT for real numbers, 17
- FORMATA for strings, 17
- FORMATFACTOR, 10
- FORMATI for integers, 17
- Fractions, 27
- German umlauts, 27
- GETTIME, 15
- GOTO, 18
- Greek letters, 27
- HATCHED – datasets, 36
- HATCHED areas, 23
- HLINE, 25
- HYLMAN, 40
- Identification marks, 28
- IF/ELSE (COMMAND), 45
- IF/ELSE (KASDEF), 18
- INBOX, 13
- INCLUDE (COMMAND), 37
- INCLUDE (KASDEF), 14
- INFITS, 37
- Installation, 4
- INSTRUCTION EXPORT, 14
- Instructions, 14
- Interactive mode, 12
- ISMLINE, 40
- ITEMIZE, 26
- LATEBOX, 13
- LaTeX, 27, 31
- LETTERSIZ, 13
- Line drawing, 21
- Line thickness, 19
- Max. no. of datapairs, 14
- Max. no. of datasets, 14
- MULTIPLY, 9
- NOBOX, 13
- OFS, box offset, 13
- PAPERFORMAT, 9
- PARSEVAR (string), 17
- PAUSE, 19
- PEN, 19
- PENDEF, 13
- Plot arrows, 21
- Plot circles/arcs of, 21
- Plot ellipses/arcs of, 21
- Plot error bars, 22
- Plot lines, 21
- Plot rectangle, 22
- Plot symbols (individual), 22
- Plotsymbols (Datasets), 53
- PREDEFINE-VARIABLES, 15
- READ (KASDEF), 19
- Reddening/dereddening, 41
- Running text, 24
- SKL (scaling), 13
- Special characters, 27
- Symbols plotting, 22
- SYSTEM, 15
- Tables, 26
- TABLUN, 25
- Tabulator, 25
- TEXT (running), 24
- Text strings, 23
- TICKSIZE, 14
- ttw, 7
- VAR-LIST, 17
- Variable, 15
- WAVECAL, 39
- WRITE (KASDEF), 19
- wrmult, 7
- wrpdf, 6
- wrps, 6