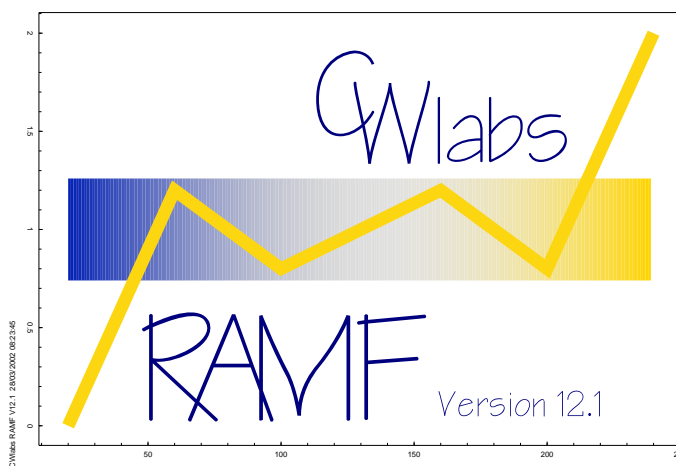

RAMF V12.1 - Reference Manual

by Wolfgang Lief & Cécilia Ewenz

Last updated: 30 March 2002



Abstract

This manual describes version 12.1 of the **RAMF** data processing system assuming a Linux platform with a shell interpreter as the front-end and GNU GhostScript as the graphical back-end (the reference platform for the purpose of this manual is a modified RedHat Linux 7.1 system (kernel 2.4.17/glibc 2.2.4) with Lahey FORTRAN 95 Ver6.0c and GNU GhostScript 5.50).

As great care has been taken to keep the code as portable as possible, most of this manual should also be applicable to R12.1 on any other platform where a FORTRAN 90 compiler and GhostScript (or a proprietary PostScript interpreter) are available. A command line/shell interpreter which is able to digest standard Unix 'sh' scripts is convenient on all platforms, but not totally essential.

Copyright

Copyright © 2002 CWlabs
Cécilia Ewenz & Wolfgang Lief, Bridgewater, South Australia

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

Contents

1	Overview	8
2	Implicit Commands	9
2.1	Command Line Processor	9
2.2	Command Interpreter	10
2.3	Conditional Construct Processor	11
2.4	Macro Processor	12
2.5	Equation processor	13
2.6	Format Expansion	16
2.7	Format Processor	17
3	Block Commands	18
3.1	Filter	18
3.1.1	Keywords	18
3.2	Gridder	20
3.2.1	Keywords	20
3.3	Plot	21
3.3.1	Group 1 Keywords	23
3.3.2	Group 2 Keywords	25
3.3.3	Group 3 Keywords	31
3.3.4	Group 4 Keywords	31
3.4	Pick	32
3.4.1	Keywords	32
3.5	Print	34
3.5.1	Keywords	34
3.6	Sort	36
3.6.1	Keywords	36
3.7	Spectra	37
3.7.1	Keywords	37
3.8	Trans	39
3.8.1	Keywords	39
4	Simple Conversion Functions	40
4.1	Conversion Functions with one result series	40
4.2	Conversion Functions with two result series	42

5	Data Import - ReadFile	43
5.1	ASCII	43
5.2	B1/D1	43
5.3	B2/D2	43
5.4	FC	44
6	Data Export - SaveFile	45
6.1	B1/D1	45
6.2	B2/D2	45
6.3	ASCII	45
7	General Single Line Commands	46
7.1	ACtoLoc3D	46
7.2	Almanac	46
7.3	alpha(acz,TAS)	46
7.4	alpha(acz,TAS,T,p)	47
7.5	AMG2LL	47
7.6	AppendSym	47
7.7	AppendVal	47
7.8	Area	48
7.9	Atan2	48
7.10	Bolton(P,T,Q)	48
7.11	Breakpoint	49
7.12	Bulkfluxes	49
7.13	CalculateTimeStep	51
7.14	Clear	51
7.15	ClipSer	51
7.16	CombineCSI	51
7.17	Combine	52
7.18	Covariance	52
7.19	CovShift	52
7.20	CreateLd	52
7.21	CreateLn	53
7.22	CreateLs	53
7.23	CreatePn	53
7.24	CreatePs	53
7.25	DateSince	53
7.26	DecDaysSince	54

7.27 DEMevaluate	54
7.28 Detrend	54
7.29 DetrendSpec	54
7.30 Diff	55
7.31 FwdDiff	55
7.32 DiffTag	55
7.33 DirnHelix	55
7.34 Dist(lat)	55
7.35 Dist(lon)	55
7.36 Div0	56
7.37 EleAve	56
7.38 EleSD	56
7.39 EleNum	56
7.40 Faster	56
7.41 FFT	57
7.42 FiltInt	57
7.43 GetTimeStep	58
7.44 GKX,GKY(LAT,LON)	58
7.45 GPStoBATS	58
7.46 hhmssh(sec)	59
7.47 Histogram	59
7.48 HSI2RGB	59
7.49 Integrate3	60
7.50 Int3Tag	60
7.51 InvFFT	60
7.52 Interleave	61
7.53 Join	61
7.54 Lat(dist)	61
7.55 Lon(dist)	61
7.56 LengthsCSI	61
7.57 LL2AMG	62
7.58 MatchEle	62
7.59 Merge	62
7.60 mHDG(tHDG,lat,lon,ht)	62
7.61 MinMax0X	63
7.62 ModifyCSI	63
7.63 Monotony	63

7.64	NormEle	64
7.65	Polynomial	64
7.66	QFF(z,p,Tv)	64
7.67	QNH(p,Tv,z)	64
7.68	RandomSeries	65
7.69	Response	65
7.70	Reverse	65
7.71	RevMeteolab	65
7.72	Rot2D	66
7.73	RandomSeed	66
7.74	RunInt	66
7.75	RunningCovar	66
7.76	RunningMean	67
7.77	RunningStDev	67
7.78	RunningMin	67
7.79	RunningMax	67
7.80	RunSum	67
7.81	sec(hhmmssh)	68
7.82	Select	68
7.83	SelS1(S2)	68
7.84	SetS1(S2)	68
7.85	Shift	69
7.86	Slower	69
7.87	SSTcorTC	69
7.88	StripInvalid	70
7.89	TASinAC	70
7.90	TAS(TS,TT)	70
7.91	TC(T,W,P,AH,RhoC)	70
7.92	TE(T,w,P,aH)	71
7.93	tHDG(mHDG)	71
7.94	tHDG(mHDG,lat,lon,ht)	71
7.95	ThreshCross	72
7.96	SymDist	72
7.97	Dist(Orig)	72
7.98	Dist(Orig,Trk)	72
7.99	Truncate	72
7.100	Ts(Tt,sp,dp)	73

7.101	<code>tas(t,qc,p,td,r)</code>	73
7.102	<code>Tt(Ttm,sp,dp,r)</code>	73
7.103	<code>uvw(ECEFuvw)</code>	74
7.104	<code>WarnInvalid</code>	74
7.105	<code>XYZ2LLA</code>	74
7.106	<code>zm(p,Tv)</code>	74
7.107	<code>z(p',Tv')</code>	75
7.108	<code>z(p,Tv)</code>	75
8	Grid Manipulation Commands	76
8.1	<code>2Daxes</code>	76
8.2	<code>2Dblank</code>	76
8.3	<code>2Dcoord</code>	76
8.4	<code>2DdiffX</code>	76
8.5	<code>2DdiffY</code>	76
8.6	<code>2DflipTB</code>	77
8.7	<code>2Drotate</code>	77
8.8	<code>2Dselect</code>	77
8.9	<code>2DSmooth</code>	78
8.10	<code>InterpolateColumn</code>	78
8.11	<code>InterpolatePath</code>	78
8.12	<code>ThinGrid</code>	78
9	Examples	80
10	Appendices	82
10.1	Fonts	82
10.1.1	Recommended Text Fonts	82
10.1.2	RAMFsymbols Font	83
10.1.3	Predefined Colours	84
10.1.4	Predefined Line Types	84
11	GNU Free Documentation License	85
11.1	Applicability and Definitions	85
11.2	Verbatim Copying	86
11.3	Copying in Quantity	86
11.4	Modifications	87
11.5	Combining Documents	88
11.6	Collections of Documents	89

11.7 Aggregation With Independent Works	89
11.8 Translation	89
11.9 Termination	90
11.10Future Revisions of This License	90

1 Overview

2 Implicit Commands

Implicit Command is the designation for any **RAMF** subroutines which are not explicitly called (e.g. by using their name in a command file), but which are called 'as necessary' by the program itself during the various processing stages.

Whenever the execution of **RAMF** is started, any command line parameters are passed on to the command line processor which then initiates any further processing steps as necessary (or enters interactive mode and prompts for commands to be entered from the console if the command line was empty).

2.1 Command Line Processor

Syntax: {command1}; {command2}; \
... \
{commandn-1}; {commandn} ! {comment}

Description: The *command line processor* is the initial processing agent for any commands passed to **RAMF**. Commands are read either from the console or command files. Each line is processed as follows before it is passed on to the command interpreter:

- if the last character of a line is a backslash (\), the next line is appended to it
- any characters following an exclamation mark (!) which is not part of a string enclosed in quotation marks (') marks are treated as a comment and discarded
- if the first character of the line is not a tilde (~), and the verbosity level is set to 3 or higher, it is echoed to the console
- the line is cut into sub-commands at every semicolon (;) outside a string enclosed in quotation marks

2.2 Command Interpreter

Syntax: {command}

Description: The *command line processor* is the main hub of **RAMF** which analyzes the general structure of any lines it receives and, depending on the result passes them on to the

- *Conditional Construct Processor*
- *Macro Processor*
- *Equation Processor*

or the individual *Command Subroutines*

2.3 Conditional Construct Processor

Syntax: IF {expression} {one-line statement}
 or
 IF {expression} BACKTO {target name}
 or
 IF {expression} ADVANCETO {target name}
 or
 IF {expression} THEN
 ...
 ELSEIF {expression} THEN (optional)
 ...
 ELSE (optional)
 ...
 ENDIF

Description: The *conditional construct processor* evaluates any conditional constructs occurring in a **RAMF** command sequence and depending on the outcome advises the command interpreter where to continue with its work.

Parameters:

{expression} a term which can be evaluated to be true or false. Currently implemented are:

exampleformula1 op formula2 Arithmetic comparison with the operator {op} being one of (EQ, NE, GT, LT,GE ,LE)

Exist({label}) True if the series {label} is in memory, false otherwise

NotExist({label}) True if the series {label} is not in memory, false otherwise

Conditional constructs can be nested up to 64 levels deep.

2.4 Macro Processor

Syntax: `{macro name} {parameter1} ... {parametern}`

Description: If a command does not match any of the internally assigned names, it is passed to the *macro processor* which then searches the directories on the *macro path* for a matching filename. If such a file is found, its contents is subjected to parameter expansion (each occurrence of `$1` is replaced by `{parameter1}`, `$2` by `{parameter2}` and so on) and then interpreted as **RAMF** commands.

Macros can be nested up to 16 levels deep and have up to 16 parameters. Conditional constructs should not traverse macro boundaries as this will yield unpredictable results.

2.5 Equation processor

Syntax: {...} {formula expression} {...}

or

{result} = {formula expression}

Description: The `equation processor` command is never called explicitly, but invoked every time when **RAMF** has to evaluate a formula type expression. Using a syntax as close as possible to standard mathematical notation, the equation processor can handle the following elements of formulas:

- references to existing series, including element subranges
- references to implicit series
- references to numerical constants
- function macros
- mathematical and hierarchical operators

The equation processor is programmed recursively, i.e. any sub-part of a formula can be another formula.

Series References: Any combination of alphanumeric characters delimited by non - alphanumeric characters is treated as a series reference, unless it is the name of a valid macro function followed by a '('. A sub-range of elements of a series can be referenced by using element indices in square brackets.

Example(s):

<code>ExampleSeries</code>	references all elements of series 'ExampleSeries'
<code>ExampleSeries[1]</code>	references the first element
<code>ExampleSeries[]</code>	references the last element
<code>ExampleSeries[7:11]</code>	references elements 7 to 11
<code>ExampleSeries[7:]</code>	references elements 7 to the end of the series
<code>ExampleSeries[:5]</code>	references the first 5 elements

Implicit Series: Two types of implicit series are available in formulas:

- *enumerated series* are defined by listing all elements separated by commas, enclosed in square brackets
- *parametric series* are defined by stating the series size, separated by a colon from the value or lue range, all enclosed in square brackets

Example(s):

<code>[0,1,2,3,4,5]</code>	is an enumerated series of five elements
<code>[10:1]</code>	is a parametric series of ten elements, all have the value 1
<code>[10:1,10]</code>	is a parametric series of ten elements, linearly increasing from 1 to 10

Numerical Constants: Numerical constants can be stated in any notation valid for FORTRAN REAL numbers:

- as integers without a decimal point
- as floating point numbers with a decimal point and optionally trailing digits

- optionally using the 'E' code for implicit multiplication with powers of ten
- '+' and '-' are allowed in front of both, mantissa and exponent, if the sign is omitted, the respective value is treated as positive

Example(s):

- 1 an integer number
- 1. an integer number written as a floating point number
- 1.23 a floating point number
- 1.23E5 a floating point number with decadic multiplier
- 1.23E-5 a negative floating point number with a negative decadic multiplier

Function Macros: The inbuilt function macros can be referenced by using their respective name and an operand enclosed in brackets. The available macros are:

- ABS Absolute value
- ACOS Inverse cosine (in degrees)
- ADEV Average absolute deviation
- ASIN Inverse sine (in degrees)
- ATAN Inverse tangent (in degrees)
- COS Cosine (argument in degrees)
- EXP e function
- INDEX Indices of the elements of a series
- INT Integer part
- KURT Kurtosis
- LN Logarithm to base e
- LOG10 Logarithm to base 10
- MAX Maximum value
- MEAN Mean value
- MEANN Contribution of negative values to mean
- MEANP Contribution of positive values to mean
- MIN Minimum value
- NELE Number of elements
- NINT Nearest integer value
- SD Standard deviation
- SDN Contribution of negative values to standard deviation
- SDP Contribution of positive values to standard deviation
- SIN Sine (argument in degrees)
- SKEW Skewness
- SQRT Square root
- SUM Sum of elements
- TAN Tangent (argument in degrees)

VAR Variance

Example(s):

NELE(SeriesA) returns the number of elements in SeriesA

INDEX(SeriesB) returns a series with the indices of the elements of SeriesB

SIN(SeriesC) returns the sines of all elements of SeriesC

MIN(SeriesD) returns the smallest value in SeriesD

Mathematical and hierarchical operators:

The equation processor interprets the four classical mathematical operators '+', '-', '*', and '/', as well as the exponentiation operator '^' and the concatenation operator '|'. Unless forced differently using brackets as hierarchical operators, they are evaluated in the usual sequence (i.e. '|', '^', '/', '*', and '+' and '-').

Example(s):

(2+1)*(SeriesA|[1,1,1]) performs a calculation

2.6 Format Expansion

Syntax: `~{formula}~`

or

`~{formula};{format code}~`

Description: With few exceptions, strings used in **RAMF** commands can contain a format expansion sequence enclosed in '~' (tilde) characters, which will be replaced by its evaluation. While this feature is mainly useful for inclusion of variable data in output products (i.e. plots, printouts, or data files), it can also be used to manipulate file names, etc..

Parameters: `{formula}`

An expression which can be evaluated by the equation processor to yield a scalar numerical value.

`{format code}`

A format expression which can be evaluated by the format processor. If no format code is present, the default format '*' will be used.

2.7 Format Processor**Syntax:** {format}

or

{parameter equation}:{format}

Description: The format code is used to convert numerical values to a text representation wherever such conversions are necessary for inclusion of values into output products.**Parameters:** {parameter equation}

This can be used to modify the value of the number to be represented. It can be any numerical expression that can be evaluated by the equation processor, with the '@' symbol representing the value to be modified. If no parameter equation is present, the unmodified value will be used.

{format code}

The format code can be any FORTRAN 90 edit code valid for REAL numbers, or one of the following special codes:

- * the default format; normally 'R' unless overridden by SET DEFAULTFORMAT
 - R try to represent the number as 'nice' as possible
 - RS{x} same as R, but the output is suppressed if the value is equal to {x}
 - RZ same as R, but values close to zero are rounded to zero
 - I integer part of the value as a default integer
 - IS{i} same as I, but the output is suppressed if the value is equal to {i}
 - D interpret value as DDMMYY date code and write as DD/MM/YY
 - H interpret value as HHMMSSHH time code and write as HH
 - HM interpret value as HHMMSSHH time code and write as HH:MM
 - HMS interpret value as HHMMSSHH time code and write as HH:MM:SS
 - SH interpret value as decimal seconds and write as HH
 - SHM interpret value as decimal seconds and write as HH:MM
 - SHMS interpret value as decimal seconds and write as HH:MM:SS
 - A interpret (value modulo 256) as an ASCII character code
 - [. . . , . . . , . . .] ignore value, use elements of a comma delimited list of strings for consecutive conversions
 - (. . . , . . . , . . .) use integer part of value as an offset into a comma delimited list of strings
- Additionally, one or more format modifiers can be used to further customize all predefined formats:
- + add an additional blank character before and after the string (can be repeated to add multiple blanks)

3 Block Commands

3.1 Filter

Syntax: Filter
 {keywords}
 End

Description: Filtering of series in time/space domain

3.1.1 Keywords

FILTERTYPE {type} {parameters}

Select the type of filter to apply.

{type} – one of the options RUNMEAN, FFT, KELLY, NOAA, COSINE, LANCZOS, SQLANCZOS

{parameters} – additional parameters for the KELLY filter

Available filter types are:

RUNMEAN – Running mean filter

FFT – Frequency domain filter using a Fast Fourier Transform

KELLY – Kelly's recursive high pass filter

NOAA – NOAA cubic mirror-image replacement filter

Linear symmetric filters with weights:

COSINE – $0.5 * (1 + \cos(\pi * (i - 1)/nWeights))$

LANCZOS – $\sin(\pi * (i - 1)/nWeights) / (\pi * (j - 1)/nWeights)$

SQLANCZOS – square of Lanczos

Additional parameters for the KELLY filter:

{rip} – maximum amount of ripple (dB) desired in the filter response function

{airspd} – average TAS

{wvlcut} – wavelength (m) of filter cut-off

PASS {highlow} {cutoff1} {cutoff2}

Define the pass characteristic of the filter.

{highlow} – one of HIGH, LOW, BAND, INVBAND

{cutoff1} – cut-off or lower corner frequency

{cutoff2} – cut-off or upper corner frequency

DATAINT {dt}

{dt} – data interval

WEIGHTS {n}

Override the automatically generated number of weights for linear symmetric filters (Cosine, Lanczos, or SqLanczos); the resulting filter consists of a central weight and nWeights-1 weights on each side of the central point

{n} – number of weights

APPEND {type}

Preserve length of input series by appending an appropriate number of elements to both ends of the series before filtering; if not specified, the output series will be shorter (linear symmetric filters and RunMean only)

{type} – one of MEAN, POINT, PLANE

The available append modes are:

MEAN – append mean of series

POINT – append elements using point symmetry

PLANE – append elements using plane symmetry

ALL – no parameters –

SELECT Filter all currently defined series. {s1} {s2} ... {sn}

Transform specified series.

{si} – series to filter

SAVEWEIGHTS {weights}

Store filter weights are saved in specified series. If Type = 'FFT', weights will be in the frequency domain, and stored in "wrap-around" order (see Numerical Recipes, p394).

{weights} – series to store weights in

3.2 Gridder

Syntax: Gridder
 {keywords}
 End

Description: generate grid point data on a regular grid from irregularly spaced data

3.2.1 Keywords

TRANSTYPE {type} {parameters}

Select the type of transformation to apply.

{type} – one of the options *EQUI*, *NONEQUI*, *TAS*

{parameters} – additional parameters depending on {type} (see below)

EQUI – transforms $f(x)$ to $f(y)$, where both coordinates are equidistant

NONEQUI – transforms $f(x)$ to $f(y)$, where both coordinates may be non-equidistant

TAS – transform time series $f(t)$ to space series $f(y)$ using *TAS* only (y -coordinates are space coordinates moving with the mean wind)

The type dependent parameters are:

3.3 Plot

```

Syntax: Plot
           End

           or

           Plot
           {primary keywords}
           End

           or

           Plot
           {primary keywords}
           Flush
           {secondary keywords}
           End

```

Description: The `plot` command is used to convert the numerical data generated by **RAMF** into a graphic representation in a standardized manner, while still leaving plenty of room for customization of the output for specific projects/tasks. A set of 'primary' commands can be used to generate a number of 'graphs', each containing an arbitrary number of 'curves', 'labels', and 'axes'. In a second (optional) processing step, 'secondary' commands can be used to directly manipulate the drawing surface (although this should remain the exception rather than the normal way of generating plots).

The output of `plot` is generated in the form of a PostScript file which can easily be viewed, printed, and converted into various other formats using tools freely available tools for most platforms (e.g. GhostScript, GhostView, gv, ps2pdf, etc.).

A **RAMF** plot is made up of a hierarchic group of elements - items of a higher class can act as 'containers' for one or more items of a lower class. Each item has certain properties whose default values can be overridden by the user.

The key classes are:

- PLOT** a plot is a single page of graphics output; the plot size equals the page size less the plot margins. A plot has properties like 'frame', 'logo', or 'margins' and can contain elements of the classes **GRAPH** and **HEADER**.
- HEADER** a header is a line of text plotted above the rest of the graphics area. A header has properties regarding its appearance and positioning, but cannot contain additional elements.
- GRAPH** a graph is a space within the plot area, bounded by four axes (which are not necessarily visible), the primary and secondary X and Y axes, respectively. Additional axes may be present, but do not play any role in scaling or bounding of the graph. A graph has properties like 'title' and 'innermargins' and can contains elements of the classes **AXIS**, **CURVE**, and **LABEL**.
- AXIS** an axis is a plot element which has a lot of properties related to scaling information and 'decoration' of the plot. The two 'primary' axes of each graph define the extent and scaling of that graph.
- LABEL** a label is an annotation which can be placed anywhere in the graph area.
- CURVE** a curve is any of the data-based graphics elements which can be added

to a graph. The currently available types are CURVE, MARKS, CONTOURS, BITMAP, SHADING, DOMAIN, and PROFILE.

Units of Measure and Declaration:

- the unit of measure used by plot for all dimensions (e.g. length, line thickness, font size etc.) is millimetres.
- angles are measured in degrees, with 0 degrees pointing straight upwards, and counting clockwise from there.
- colours are described by an index into a colour palette of 256 entries. Indices 0 to 15 are predefined with standard colours (refer to table 3 on page 84 for details), the rest can be assigned by the user with the PALETTE command.
- line types are defined as indices into a table containing style descriptors. Filled shapes are implemented as a special line style (i.e.. in this case the path formed by the line is closed and filled with the line colour). The predefined default values (refer to table 4 on page 84 for details) can be modified with the DASHLIST command.
- character fonts are defined by their PostScript name. It is recommended to use only 'standard' fonts, e.g. those distributed with the GhostScript package (Table 1 on page 82 shows list of these). Alternatively **RAMF** can embed any PostScript Type 1 ASCII fonts (pfa) in its plot files, which are either part of the **RAMF** package, (located in *R12/lib/fonts*) or supplied by the user.
- line style definitions appear in triplets of the form

```
{thickness} {colour} {type}
```

where

```
{thickness}  the line width
{colour}     the colour index
{type}      the type index
```

Values which are omitted or specified as '*' will default to 'reasonable' standard values.

- text style definitions appear in the form

```
{size} {colour} {font}
```

where

```
{size}  the font cell height in mm
{colour} the colour index
{font}  the font name
```

Values which are omitted or specified as '*' will default to 'reasonable' standard values.

Default behaviour: If nothing is specified, plot will generate an overview plot of all series currently in memory.

Keywords: The keywords used by the plot command can be grouped into four categories:

- *Group 1*: primary keywords which can be used anywhere in the primary section (i.e. between a `plot` command and a `flush` command or an `end` statement)
- *Group 2*: primary keywords which can be used only after at least one `graph` has been defined
- *Group 3*: primary keywords which can be used only after at least one `curve` has been defined (explicitely or implicitely)
- *Group 4*: secondary keywords which can be used in the secondary section (i.e. between a `flush` command and the `end` statement)

3.3.1 Group 1 Keywords

- END** - no parameters -
 Terminate the current plot command.
- FLUSH** - no parameters -
 Finish evaluation of primary commands, process all graphs, and begin with processing of secondary commands.
- FRAME** {mode} {space} {innerstyle} {outerstyle}
 Plot a frame around the whole page and/or the area occupied by graphs
- {mode} – one of the options INNER, OUTER, BOTH, or OFF
 - {space} – the width of the blank space both sides of the frame line
 - {innerstyle} – the line style of the inner frame
 - {outerlinestyle} – the line style of the outer frame (if not present, it defaults to the value of the inner frame)
- GRAPH** {xseries} {yseries} {xlabel} {ylabel} {textstyle}
 Add another graph to the plot, and if at least one of {xseries} or {yseries} is present and valid, add an initial curve to this graph.
- {xseries} – series to plot as X coordinate (defaults to element index of {yseries})
 - {yseries} – series to plot as X coordinate (defaults to element index of {xseries})
 - {xlabel} – axis label to use for primary X axis (defaults to {xseries})
 - {ylabel} – axis label to use for primary Y axis (defaults to {yseries})
 - {textstyle} – text style to use for axis labels
- GRAPHRESETDEFAULT** - no parameters -
 Reset the default graph to its original state.
- HEADER** {position} {text} {textstyle}
 Add another header line to the whole plot.
- {position} – one of LEFT,RIGHT,CENTER
 - {text} – the header text

- `{textstyle}` – the text style for the header
- LOGO** `{option} {text} {textstyle}`
 Override the logo text default setting (LOGODATE).
- `{option}` – one of LOGO,LOGODATE,USER,USERDATE,OFF
`{text}` – the text for the user logo
`{textstyle}` – the text style for the logo
- MARGINS** `{option} {left} {right} {bottom} {top}`
 Set or modify the margins for the whole plot. If a value is given as '*', the previous setting of that value will not be modified.
- `{option}` – one of SET,ADD
`{left}` – left margin
`{right}` – right margin
`{bottom}` – bottom margin
`{top}` – top margin
- NEWROW** `{n}`
 Force a row advance and set the number of graphs per row to a new value.
- `{n}` – the new value for 'graphs per row'
- NOGRAPHS** – no parameters –
 Switch off the default plot generated when no graph is defined in the plot, i.e. plot only frame, headers, and logo.
- SELECT** `{series list}`
 Select the listed series for a a 'Y over element index' plot.
- `{series list}` – a list of expressions the formula processor can digest
- SELECTTAG** `{tag} {tagname} {series list}`
 Select the listed series for a a 'Y over tag series' plot.
- `{tag}` – the series to use as tag series
`{tag name}` – the text to use as label for the bottommost x-axis
`{series list}` – a list of expressions the formula processor can digest
- SET** `{what} {value}`
 The normal **RAMF** set command is available from within plot, i.e.. all parameters definable by set are accessible.
- `{what}` – parameter name
`{value}` – value to set it to

The following parameters are directly connected with the plot command:

FONTPATH is the search path for font files

PALETTEPATH is the search path for palette files

DEFAULTFONT is the name of the default font

SIZE {formatcode}

or

SIZE {width} {height}

Set the plot size (which will also be written into the 'Bounding Box:' field of the PostScript file).

{formatcode} – one of A4,A4V,A4P,A4H,A4L

{width} – plot area width [mm]

{height} – plot area width [mm]

3.3.2 Group 2 Keywords

ASPECTRATIO {xratio} {yratio}

Set the aspect ratio of a graph

{xratio} – relative size factor of the horizontal axis (default =1)

{yratio} – relative size factor of the vertical axis (default =1)

AXIS {axis} {mode} {position} {linestyle}

Modify the properties of an axis

{axis} – index of the axis to operate on (1=primary x, 2=primary y, 3=secondary x, 4=secondary y, ...). This value will default to the most recently created axis if it is omitted.

{mode} – one of LINE,NOLINE,BOX,OFF

{position} – intersection value of the corresponding primary axis where to position the axis

{linestyle} – line style for the axis line

AXISADD {orientation} {mode} {position} {linestyle}

Add a new axis to a graph. As indices 1-4 are allocated to the pre-defined axes, the first added axis will get index 5, the next 6, and so on.

{orientation} – one of X,Y

{mode} – one of LINE,NOLINE,BOX,OFF

{position} – intersection value of the corresponding primary axis where to position the axis

{linestyle} – line style for the axis line

AXISDELTA {axis} {delta} {offset} {itiny} {ismall} {ibig}

Override the automatic tick spacing for an axis.

{axis} – index of the axis being worked on (refer to **AXIS** command for details)

{delta} – size of tick base interval

{offset} – start point for tick mark generation

`{itiny}` – interval count for tiny ticks

`{ismall}` – interval count for small ticks

`{ibig}` – interval count for big ticks

AXISLABEL `{axis} {direction} {orientation} {text} {textstyle}`

Setup of the axis label (unlike the implicit labeling in the `graph` command, this allows to independently set the parameters for all axes).

`{axis}` – index of the axis being worked on (refer to `AXIS` command for details)

`{direction}` – one of LEFT,RIGHT,BOTH,OFF

`{orientation}` – one of NORMAL,ROT90,ROT180,ROT270

`{text}` – the label text

`{textstyle}` – the text style for the label

AXISLENGTH `{axis} {length}`

Override automatic plot space distribution and force the length of an axis to a set value (this does currently not work together with auto-spaced graphs in the same plot).

`{axis}` – index of the axis being worked on

`{length}` – the length the axis is supposed to have

AXISNUMBERS `{axis} {type} {direction} {orientation} {textstyle} {format} {interval}`

Setup the numbers to appear next to tick marks (or at least hypothetical tick mark positions)

`{axis}` – index of the axis being worked on

`{type}` – location of the numbers, one of BIG, SMALL, TINY, OFF

`{direction}` – one of LEFT, RIGHT, BOTH, OFF

`{orientation}` – one of NORMAL, ROT90, ROT180, ROT270

`{textstyle}` – the text style for the numbers

`{format}` – any valid format descriptor

`{interval}` – spacing between numbers (default=1 -j plot each number)

AXISRANGE `{axis} {min} {max} {start} {end}`

Override the automatic min/max calculation for axis scaling.

`{axis}` – index of the axis being worked on

`{min}` – minimum value to appear on axis

`{max}` – maximum value to appear on axis

`{start}` – start point of axis in units of the respective primary axis (default: start of primary)

`{end}` – end point of axis in units of the respective primary axis (default: end of primary)

- AXISSAME** {axis} {index}
- Force an axis to have the same scaling as the same axis in another graph
- {axis} – index of the axis being worked on
 - {index} – index of the reference graph
- AXISTICKS** {axis} {type} {direction} {size} {linestyle}
- Set up tick marks.
- {axis} – index of the axis where the tick marks are located
 - {type} – one of BIG,SMALL,TINY,OFF
 - {direction} – one of LEFT,RIGHT,BOTH
 - {size} – length of the tick marks
 - {linestyle} – line style for the ticks
- CONTOURS** {xaxis} {yaxis} {zarray} {levels} {divisions} {linestyle} {numbers} {format} {textstyle}
- Draw contour lines for data contained in a two-dimensional grid.
- {xaxis} – a series containing the x-axis values of the grid
 - {yaxis} – a series containing the y-axis values of the grid
 - {zarray} – a series containing the folded array
 - {levels} – a series containing the level values to be contoured
 - {divisions} – determines the number of subdivisions to use for contouring, higher values will yield 'better' contours, but at the expense of a lot of memory usage.
 - {linestyle} – line style for contour lines
 - {numbers} – one of (ON,OFF) - switches contour line numbering on and off
 - {format} – format code for contour numbers
 - {textstyle} – text style for numbers
- CURVE** {xseries} {yseries} {index} {mode} {mode dependent parameters}
- Add a new curve to the current graph
- {xseries,yseries} – series to plot, if non-existing, defaults to the element index of the respective other series
 - {index} – if {index} is a single value, each index'th element of ({x,y}) will be used; if {index} is a series, it will be used as a list of element indices of ({x,y}) to use (Default=1, i.e. all values are plotted).
 - {mode} – one of LINES,DOTS,SYMBOLS,POLYGON,XAREA,YAREA
- The currently available CURVE modes and their parameters are:
- LINES** {linestyle}
- plots all requested points jointed by lines
- {linestyle} – line style

- DOTS** `{diameter} {colour}`
does not join the points
`{diameter}` – diameter of dots
`{colour}` – colour of dots
- POLYGON** `{linestyle}`
Plot a filled polygon with the requested points as corners (max. 2000 points)
`{linestyle}` – line style
- XAREA** `{linestyle} {yvalue}`
Fill the area between the graph and the parallel to the X axis at yvalue
`{linestyle}` – line style
`{yvalue}` – base value
- YAREA** `{linestyle} {xvalue}`
Fill the area between the graph and the parallel to the Y axis at xvalue
`{linestyle}` – line style
`{xvalue}` – base value
- DASHLIST** `{which} {dashlist}`
Define or override a line style definition
`{which}` – index of the line style to define/override
`{dashlist}` – series containing a postscript dash descriptor (circular list of on/off segments) in multiples of the line width; e.g. the dash descriptor of the default dashed line is [32,32].
- DOMAIN** `{xser} {yser} {index}`
Add coordinates to the auto-scaling data pool without actually plotting anythings
`{xseries}` –
`{yseries}` –
`{index}` – same as in CURVE
- GRAPHMARGINS** `{option} {left} {right} {bottom} {top}`
Set or adjust the margins of the current graph within its allocated plot space.
`{option}` – one of ADD,SET
`{left}` –
`{right}` –
`{Bottom}` –
`{top}` – the margin values

- GRAPHSETDEFAULT** - no parameters -
Define the current settings of the current graph as the default for all further graphs.
- GRID** {axis} {type} {linestyle}
Set up grid lines.
 {axis} - index of the axis the grid will be based on
 {type} - the ticks on this axis the grid will be based on (BIG,SMALL,TINY)
 {linestyle} - the line style of the grid lines
- INNERMARGINS** {xpercent} {ypercent}
Set the overscale factor of the graph autoscaling for the current graph (default 0.05, ie. 5overscaling). An overscale factor of 0.0 means that all curve points fit exactly within the graph area.
 {xpercent} -
 {ypercent} - the factors for x and y axis, respectively
- LABEL** {mode} {ctype} {x} {y} {text} {textstyle} {line length} {line width} {linestyle}
Add an annotation/label/legend to the current graph. In addition to the label text, a symbol can be plotted and a horizontal line drawn.
 {mode} - one of LEFT,RIGHT,CENTER
 {ctype} - type of coordinate system, one of RELATIVE, ABSOLUTE,REALWORLD
 {x} -
 {y} - position of the label
 {text} - label text. If the text is beginning with @xxxx (separated from the rest of the text by a blank), the symbol with the name xxxx is plotted centered on the label line (see below)
 {textstyle} - text style
 {line length} - if set to $\neq 0.0$, a line will be plotted next to the label text
 {linestyle} - line style
- MARKS** {type} {x} {y} {index} {type-specific parameters}
Plot a mark at each point designated by index(x,y).
 {type} - the type of the marks to plot. Currently available are: ARROWS, SYMBOLS, TICKS
 {x} - x-coordinate
 {y} - y-coordinate
 {index} - if {index} is a single value, each index'th element of ({x},{y}) will be used; if {index} is a series, it will be used as a list of element indices of ({x},{y}) to use.
- All further parameters depend on the type of the requested marks:
- ARROWS** {type} {angle} {size} {linestyle}

`{type}` – NORMAL,CENTERED

`{angle}` – the angle of the arrow in degrees with 0 being up and counting clockwise positive

`{size}` – length of the arrow, including head

`{linestyle}` – the line style used for the arrow

SYMBOLS `{position} {value} {angle} {format} {textstyle}`

`{position}` – LEFT,RIGHT,CENTRE

`{value}` – the values to plot

`{angle}` – the angle of the numbers in degrees with 0 being up and counting clockwise positive

`{format}` – the format code to use for the numbers

`{textstyle}` – the text style to use for the numbers

TICKS `{type} {size} {linestyle}`

`{type}` – LEFT, RIGHT, CENTRE

`{size}` – size of the ticks

`{linestyle}` – the line style used for the ticks

PALETTE `{mode} {startindex} {mode depending parameters}`

Redefine the plotting colour palette either using pre-defined palettes which are loaded from an ASCII file, or series containing red/green/blue intensity values.

`{mode}` – SET, LOAD or COLOUR

`{startindex}` – index of first colour to modify (colours 0-15 are usually reserved as primary system colours and should not be redefined)

In SET mode the following parameters are:

`{red}` –

`{green}` –

`{blue}` – series containing the relative intensities [0...1] of the colour components

In LOAD mode, the next parameter is the name of the file containing the palette.

In COLOUR mode the next parameter is a list X11 colour names.

PROFILE `{dir} {xseries} {yseries} {index} {offset} {scale} {linestyle}`

`{dir}` – VERT or HORZ designates the orientation of the profile

`{xseries}` –

`{yseries}` – the series to use for x and y coordinates

`{index}` – see CURVE for description

`{offset}` – offset of the footpoint in primary axis coordinates

`{scale}` – scaling factor between primary axis and profile

{linestyle} – line style

SHADING {xax} {yax} {zgrd} {factor} {offset} {xleft} {ytop} {xright} {ybottom}

Converts a 2D-grid into a shaded bitmap

{xax} –

{yax} – the axes values of the grid

{zgrid} – the grid data

{factor} –

{offset} – the scaling values used to transform the grid values into colour palette offsets

{xleft} –

{ytop} –

{xright} –

{ybottom} – extremal values for the shaded area

TITLE {position} {text} {textstyle}

{position} – one of LEFT,RIGHT,CENTER,OFF

{text} – the title text

{textstyle} – the text style for the title

3.3.3 Group 3 Keywords

LINESTYLE {mode} {mode dependent parameters}

Set style of line (in particular those defined in a GRAPH or PROFILE statement)

{mode} – one of LINES, DOTS, SYMBOLS, POLYGON, XAREA, YAREA

{linestyle} – line style

all other mode dependent parameters are described under CURVE.

3.3.4 Group 4 Keywords

The secondary plotting commands allow direct access to the low-level plot routines. All coordinates used by secondary commands are in millimetres relative to the plot system (i.e. page) origin.

3.4 Pick

Syntax: Pick
 {keywords}
 End

Description: Pick values from series according to some specification using an indicator series

3.4.1 Keywords

INDICATOR {indicator series}
 Defines the indicator series

BETWEEN {lower} {upper}
 desired range of indicator series; elements for which {lower} \leq {indicator series} \leq {upper} will be accepted, other elements will be rejected and omitted or replaced according to 'cut' or 'inpol'

SEGMENT {left} {right}
 series element numbers of left and right extremities to be subjected to the omission criteria (default: whole series)

LEFTRIGHT {left} {right}
 number of elements to be included in rejection interval at both ends of the interval

POINTS {left} {right}
 to linearly interpolate between two designated points of a series.
 left/right : integer values of series to interpolate between

CUT - no parameters -
 cut out the parts of the series which do not fulfil the condition for the indicator series

SET {value}
 set values the series which do not fulfil the condition for the indicator series to value

INPOL {InpolType}
 replace the parts of the series which do not fulfil the condition for the indicator series with interpolated values. InpolType = 'LINEAR' - linear interpolation

MAXPERCENT {percent}
 If the total rejected period is greater than percent**RAMF** stops in error.
 Default: 50

APPTYPE {Type}
 Appending technique to be used when bad values occur at either end of the series and "Inpol" is being used (as opposed to "Cut"). Type="constant": Append with the last "good" value ="linear": Append via linear interpolation from the last 2 "good" points (WARNING: this can produce crazy results if there is a big jump between the last 2 "good" points).

ALL - no parameters -

Perform 'picking' for all currently defined series/labels (including indicator series)

SELECT {label1} {label2} ... {labeln}

Perform 'picking' for specified series/labels only

3.5 Print

Syntax: Print
 {keywords}
 End

Description: The `print` command allows to print series, or parts of series, to the screen, the protocol file, a separate ASCII file, or a spreadsheet file which can be imported into most commercial spreadsheet programs.

3.5.1 Keywords

DATAFORMAT {data format}
 Set the data format for all currently selected series
 {dataformat} – the format to use for printed data (see section 2.7 on page 17 for details)

FROMTO {from} {to} {seq}
 Define the range and spacing of the elements to be printed
 {from} – index of first element to be printed (default: beginning of series)
 {to} – index of last element to be printed (default: end of series)
 {seq} – spacing between elements (default: every element)

MAXLINES {max}
 Limits the number of printed lines to max. If the total number of elements to be printed is greater than max, seq is changed so that the total number falls just below max.
 {max} – maximum number of lines (default: 50)

NOHEADER - no parameters -
 Disable output of the column headers

NOPRT - no parameters -
 Disable output to screen/PRT file

SELECT {label1} {label2}
 Select series to be printed (wildcards possible)

SERIES {series} {dataformat} {headertext} {columnwidth}
 Select a series to be printed
 {series} – name of the series/formula, if set to '*', an element index will be printed
 {dataformat} – can either be '*', 'R' (default REAL), 'I' (default INTEGER) or any FORTRAN format code valid for REAL numbers
 {headertext} – text to print in the column header, '*' uses series label (the text will be 'format expanded')
 {columnwidth} – width of the column

TITLE {text} {option}

Print a title on top of the printout (default: 'default printout')

{option} – 'CENTRE', 'LEFT', or 'RIGHT' (default: CENTRE)

TOFILE {mode} {filename} {option} {delimiter}

Print to a file

{mode} – 'ASCII' or 'LOTUS' (default: ASCII)

{filename} – name of the file to write to

{option} – one of 'OVERWRITE', 'NOOVERWRITE', or 'APPEND' (default: APPEND)

{delimiter} – the column delimiter to use in ASCII files (default: |)

3.6 Sort

Syntax: Sort
 {keywords}
 End

Description: Sort series in increasing order OR Fix identical frames OR both.

The series will be sorted in increasing order, i.e. the first element will be the minimum and the last the maximum of the series. Selected series will be sorted synchronously to the first one.

Optionally: In a second step, frames with successive equal values of first series will be eliminated. Values of sets of identical frames will be averaged.

3.6.1 Keywords

SORTKEY {series}

Select the series to sort/use as the sort/frame-fixing key

{series} – the series to operate on

EQUI – transforms $f(x)$ to $f(y)$, where both coordinates are equidistant

NONEQUI – transforms $f(x)$ to $f(y)$, where both coordinates may be non-equidistant

TAS – transform time series $f(t)$ to space series $f(y)$ using TAS only (y-coordinates are space coordinates moving with the mean wind)

SELECT {s1} {s2} ... {sn}

Select series which will be sorted/fixed according to key series

{si} – series to operate on

FIXFRAMES {epsilon}

Fix identical frames

{epsilon} – "tolerance" range for series values to be regarded as "close enough" to be the same. (e.g.. 0.0 for exactly equal, 5.0 for $s(i)-s(i+1) \leq 5.0$ to be o.k.)

NOSORT -

USEFIRST No sorting, fix identical frames only -

For identical frames, use first one (makes only sense together with NOSORT, default is to use the mean of identical frames)

3.7 Spectra

Syntax: Spectra
 {keywords}
 End

Description: Computes power- and cross-spectra using fast Fourier transforms (FFT).

Fourier transformations to spectra are performed whenever the card "Transfm" is specified. Prior to this, one or two series labels must be specified using "SeriesX" and "SeriesY". If only one is given, only the power spectrum of the X series can be computed. The other cards appearing before "Transfm" in the list below are optional, and may be specified once before "Transfm", if required. Once the transformations have been performed, the user may "Smooth" or "Compact" the spectra (only once for each call to "Tranfm"), before calling either the evaluation card "Evaluate" any number of times. This card represents the output from this menu, and must be called at least once for each call to "Transfm".

3.7.1 Keywords

SERIESX {x} {dt}

Define the x data series

{x} – x data series

{dt} – data interval of x series

SERIESY {y}

Define the y data series; it is assumed that it has the same data interval as the x series

{y} – y data series

NOTES

- Power spectra are given in units which are the square of those of the data
- Cross-spectral magnitudes have units: x-units * y-units
- Phase is given in degrees in the range: -180 to 180
- It is generally a good idea to plot turbulence spectra on a "log-log" scale in order to see the full resolution at all wavelengths. This also causes power-law relationships to appear as straight lines. The phase of the cross-spectrum should, however, be plotted with a linear y-axis. Different ways of plotting spectra are discussed by Stull in [4] pp315-316. For turbulence applications, a plot of $\log(S(f))$ or $\log(f * S(f))$ versus $\log(f)$ should produce a line of slope -5/3 in the inertial subrange if the data is uncontaminated and well-conditioned.
- In order to avoid spurious "red noise" ([4], p308), the data should be high-pass filtered, or at least de-trended, before spectra are formed. Removal of the means is automatically performed by SPECTRA as the first step of the transformation.
- See [4], pp329-335, for a good discussion of the various cross-spectra. A few short notes:

-
- The co-spectrum is useful in turbulence work, as it is a measure of the contribution of the different frequencies to the covariance of two series. The sum over frequency of all the cospectral amplitudes is equal to the covariance.
 - A peak in the cross-spectral magnitude at frequency f , indicates a strong correlation between the two series at that frequency, **WITHOUT REGARD TO PHASE DIFFERENCES**. The phase spectrum is a measure of the converse.
 - The coherency is a normalised cross-spectrum (ranges from 0 to 1) which acts like a frequency dependant but phase-independent correlation coefficient.
 - The correlation spectrum is a normalised co-spectrum.

3.8 Trans

Syntax: Trans
 {keywords}
 End

Description: Transformation of one or multiple series

3.8.1 Keywords

TRANSTYPE {type} {parameters}

Select the type of transformation to apply.

{type} – one of the options *EQUI*, *NONEQUI*, *TAS*

{parameters} – additional parameters depending on {type} (see below)

EQUI – transforms $f(x)$ to $f(y)$, where both coordinates are equidistant

NONEQUI – transforms $f(x)$ to $f(y)$, where both coordinates may be non-equidistant

TAS – transform time series $f(t)$ to space series $f(y)$ using *TAS* only (y -coordinates are space coordinates moving with the mean wind)

The type dependent parameters are:

EQUI {ny} {dx} {dy} {type}

{ny} – Number of data points for y -coordinate

{dx} – x -coordinate spacing

{dy} – y -coordinate spacing

{type} – *LINEAR* or *SPLINE* interpolation

NONEQUI {x} {y} {type}

{x} – series with x -coordinates

{y} – series with y -coordinates

{type} – *LINEAR* or *SPLINE* interpolation

TAS {dt} {dy} {TAS}

{dt} – time-coordinate spacing

{dy} – required distance-coordinate spacing

{TAS} – series containing *TAS*

ALL – no parameters –

SELECT Transform all currently defined series. {s1} {s2} ... {sn}

Transform specified series.

{si} – series to transform

4 Simple Conversion Functions

This section describes various 'simple' conversions functions, 'simple' in this context meaning that each datapoint of a series is treated totally separate from all others, i.e. the result only depends on the datapoint itself, and not its neighbours or any other datapoint in the series.

The sources used for consistent formulation of the thermodynamic equations where the notes for a lecture series given by P. Schwerdtfeger in the late 1990s [3], the atmospheric physics textbook by H. Kraus [1], and information posted on the www by B. Kümmel [2].

4.1 Conversion Functions with one result series

Syntax:	<code>{function} {result} {input1} {input2}...{inputn}</code>
<code>a(q,T,p)</code>	<code>{a} {q} {T} {p}</code> Absolute humidity [g/m^3] from specific humidity [g/kg], temperature [C], and pressure [hPa]
<code>a(rh,T)</code>	<code>{a} {rh} {T}</code> Absolute humidity [g/m^3] from relative humidity [%] and temperature [C]
<code>a(Td,T)</code>	<code>{a} {Td} {T}</code> Absolute humidity [g/m^3] from dewpoint [C] and temperature [C]
<code>d.dec(deg)</code>	<code>{d.dec} {deg}</code> Convert angle from ddmssh format to decimal degrees
<code>deg(d.dec)</code>	<code>{deg} {d.dec}</code> Convert angle from decimal degrees to ddmssh format
<code>e(a,T)</code>	<code>{e} {a} {T}</code> Vapour pressure [hPa] from absolute humidity [g/m^3] and temperature [C]
<code>e(T,Tw,P)</code>	<code>{e} {T} {Tw} {p}</code> Vapour pressure [hPa] from dry bulb temperature [C], wet bulb temperature [C], and pressure [hPa]
<code>es(T)</code>	<code>{es} {T}</code> Saturation vapour pressure [hPa] at given temperature [C]
<code>g(lat,ht)</code>	<code>{g} {lat} {ht}</code> Gravitational acceleration [m/s^2] from latitude [deg] and altitude above MSL [m]
<code>IAS(qc)</code>	<code>{IAS} {qc}</code> Indicated airspeed IAS [m/s] from dynamic pressure [hPa]
<code>InvertCSI</code>	<code>{out} {in}</code> Invert conditional sampling indicator
<code>omega(wair,p,T)</code>	<code>{omega} {wair} {p} {T}</code> Omega [hPa/s] from w [m/s], pressure [hPa], and temperature [C]

<code>q(a,T,p)</code>	<code>{q} {a} {T} {p}</code>	Specific humidity [g/kg] from absolute humidity [g/m^3], temperature [C], and pressure [hPa]
<code>meteolab</code>	<code>{Td} {V} {Tref}</code>	Dewpoint temperature [C] from Meteolab TP4S reference temperature [C] and sensor voltage [V]
<code>p(PA)</code>	<code>{p} {PA}</code>	Pressure [hPa] from altitude [m] assuming ISA conditions
<code>PA(ps)</code>	<code>{PA} {ps}</code>	Pressure altitude [m] from static pressure [hPa] assuming ISA conditions
<code>q(rH,T,p)</code>	<code>{q} {rh} {T} {p}</code>	Specific humidity [g/kg] from relative humidity [%], temperature [C], and pressure [hPa]
<code>q(Td,p)</code>	<code>{q} {Td} {p}</code>	Specific humidity [g/kg] from dewpoint [C] and pressure [hPa]
<code>qc(IAS)</code>	<code>{qc} {IAS}</code>	Dynamic pressure [hPa] from IAS [m/s]
<code>rh(a,T)</code>	<code>{rh} {a} {T}</code>	Relative humidity [%] from absolute humidity [g/m^3] and temperature [C]
<code>rh(Td,T)</code>	<code>{rh} {Td} {T}</code>	Relative humidity [%] from dewpoint [C] and temperature [C]
<code>rho(p,T)</code>	<code>{rho} {p} {T}</code>	Dry air density [kg/m^3] from pressure [hPa] and temperature [C]
<code>T(T, IAS)</code>	<code>{Trec} {T} {IAS} {rec}</code>	Temperature [C] corrected for IAS [m/s] and recovery factor
<code>TAS(IAS,rho)</code>	<code>{TAS} {IAS} {rho}</code>	True airspeed [m/s] from indicated airspeed [m/s] and density [kg/m^3]
<code>TAS(qc,p,T)</code>	<code>{TAS} {qc} {p} {T}</code>	True airspeed [m/s] from dynamic pressure [hPa], pressure [hPa], and temperature [C]
<code>Td(a,T)</code>	<code>{Td} {a} {T}</code>	Dewpoint [C] from absolute humidity [g/m^3] and temperature [C]
<code>th(T,p)</code>	<code>{theta} {T} {p}</code>	Potential temperature [C] from temperature [C] and pressure [hPa]
<code>Tv(T,q)</code>	<code>{Tv} {T} {q}</code>	Virtual temperature [C] from temperature [C] and specific humidity [g/kg]
<code>wair(omega,p,T)</code>	<code>{wair} {omega} {p} {T}</code>	

`wair(w,alp,pch,TAS)` `{wair}` `{w}` `{alpha}` `{pitch}` `{TAS}`
 w [m/s] from Omega [hPa/s], pressure [hPa], and temperature [C]
 w [m/s] from w in aircraft coordinates [m/s], alpha [deg], pitch [deg] and TAS [m/s]

4.2 Conversion Functions with two result series

Syntax: `{function}` `{result1}` `{result2}` `{input1}` `{input2}`...`{inputn}`

`ff,dd(uair,vair)` `{ff}` `{dd}` `{u}` `{v}`
 Wind speed [m/s] and direction [deg] from wind components [m/s]

`gs,hdg(u,v)` `{gs}` `{hdg}` `{u}` `{v}`
 Ground speed [m/s] and heading [deg] from components [m/s]

`r,phi(x,y)` `{r}` `{phi}` `{x}` `{y}`
 Range and angle from coordinates

`u,v(gs,hdg)` `{u}` `{v}` `{gs}` `{hdg}`
 Speed components [m/s] from ground speed [m/s] and heading [deg]

`uair,vair(ff,dd)` `{u}` `{v}` `{ff}` `{dd}`
 Wind components [m/s] from wind speed [m/s] and direction [deg]

`x,y(r,phi)` `{x}` `{y}` `{r}` `{phi}`
 Coordinates from range and angle

5 Data Import - ReadFile

5.1 ASCII

Syntax: ReadFile ASCII {file} {out} {iele} {nskip}

Description: Read ASCII file containing a single series using FORTRAN free format until EOF or as specified.

Parameters:

{file} name of file to read
 {out} (out/series) – series to be read
 {iele} (in/scalar) – number of elements to read (default=0 – read whole file)
 {nskip} (in/scalar) – number of lines to skip before reading data (default=0)

5.2 B1/D1

Syntax: ReadFile {subformat} {file} {tag} {start} {end} {seq} {lab_len}
 All End

or

ReadFile {subformat} {file} {tag} {start} {end} {seq} {lab_len}
 Series {label1} {newlabel1} . . . Series {labeln} {newlabeln}
 End

Description: Read binary file containing multiple series in single (B1) or double (D1) precision. If the keyword "All" is used, all series contained in the file will be read using their original series names as stored in the file. The "Series" keyword allows to read selected series only, and to override their names with new ones.

Parameters:

{subformat} one of B1,D1
 {file} name of file to read
 {tag} (out/series) – time tag series
 {start} (in/scalar) – start to read from sample number "n" or time tag "t:n"
 {end} (in/scalar) – stop to read at sample number "n" or time tag "t:n"
 {seq} (in/scalar) – sample interval "n" or minimum time spacing "t:n"
 {lab_len} (in/scalar) – if present and set to '8', read old format B1 files containing 8-byte labels
 {labeli} label of series in file
 {newlabeli} label to use for series in memory (if omitted or set to '*' defaults to {labeli})

5.3 B2/D2

Syntax: ReadFile {subformat} {label} {file}

Description: Read binary file containing a single series in single (B2) or double (D2) precision.

Parameters:

{subformat} one of B2,D2

{file} name of file to read
{label} (out/series) – series to read

5.4 FC

Syntax: ReadFile FC {file} {n} {nHeaderlines} {nHeaderchars} Series {label1} {colno1} . . Series {labeln} {colnon} End

Description: Read an ASCII file containing a multiple series using FORTRAN free format. Lines beginning with a '!', 'c', or 'C' and empty lines will be ignored.

Parameters:

{file} name of file to read
{n} (in/scalar) – number of elements to read (Default=0 – read whole file)
{nHeaderLines} (in/scalar) – number of lines to skip at beginning of file (Default=0)
{nHeaderChars} (in/scalar) – number of characters to skip at beginning of each line (Default=0)
{labeli} name to use for series
{colnoi} (in/scalar) – column number to read from

6 Data Export - SaveFile

6.1 B1/D1

Syntax: SaveFile {subformat} {file} {tag} {header} Series {label1} {name1} {comment1} . . . Series {labeln} {namen} {commentn} End

Description: Save data as binary file containing a multiple series in single (B1) or double (D1) precision.

Parameters:

{subformat} one of B1,D1
 {file} name of file to write
 {tag} (out/series) – time tag series
 {header} file header text
 {labeli} series to write
 {namei} name to use for the series in the file (if omitted or set to '*' defaults to {labeli})
 {commenti} comment text

6.2 B2/D2

Syntax: SaveFile {subformat} {label} {file} {header1} {header2}

Description: Write binary file containing a single series in single (B2) or double (D2) precision.

Parameters:

{subformat} one of B2,D2
 {file} name of file to write
 {label} (out/series) – series to write
 {header1} header text
 {header2} header text

6.3 ASCII

The PRINT command described in section 3.5 on page 34 can be used for exporting data to ASCII files.

7 General Single Line Commands

7.1 ACtoLoc3D

Syntax: ACtoLoc3D {xout} {yout} {zout} {xin} {yin} {zin} {pch} {rll} {thdg} {option}

Description: Transform 3D vector from aircraft system into local Earth coordinates or the inverse.

Parameters:

{xout}, {yout}, {zout} (out/series) – output vectors

{xin}, {yin}, {zin} (in/series) – input vectors

{pch} (in/series) – pitch angle [*deg*]

{rll} (in/series) – roll angle [*deg*]

{thdg} (in/series) – true heading [*deg*]

{option} (keyword) – if nothing is specified, the transformation will be from aircraft coordinates to Earth coordinates, if INVERT is specified, it will be the other way round

7.2 Almanac

Syntax: Almanac {Az} {ZnOb} {ZnGe} {Rfr} {Lat} {Lon} {Time} {Day} {P} {T}

Description: Computes solar angles etc. from lat/lon/time. The values computed with this command agree to 0.2' with those in the published in the tables of the Nautical Almanac.

Parameters:

{Az} (out/series) – azimuth angle [*deg*]

{ZnOb} (out/series) – zenith distance (observed) [*deg*]

{ZnGe} (out/series) – zenith distance (geocentric) [*deg*]

{Rfr} (out/series) – refraction [*deg*]

{Lat} (in/series) – latitude [*deg*]

{Lon} (in/series) – longitude [*deg*]

{Time} (in/series) – time in [*HHMMSS*]

{Day} (in/series) – date in [*DDMMYY*]

{P} (in/series) – pressure [*hPa*]

{T} (in/series) – temperature in [*C*]

7.3 alpha(acz,TAS)

Syntax: alpha(acz,TAS) {alpha} {acz} {TAS} {T} {p} {w1} {c10} {dcla}

Description: Angle of attack from vertical acceleration and TAS using mean air density.

Parameters:

{alpha} (out/series) – angle of attack [*deg*]

{acz} (in/series) – vertical acceleration [*m/s²*]

- `{TAS}` (in/series) – true airspeed [m/s]
- `{T}` (in/scalar) – mean air temperature [C]
- `{p}` (in/scalar) – mean static pressure [hPa]
- `{w1}` (in/scalar) – wing loading [kg/m^2]
- `{c10}` (in/scalar) – lift coefficient at $\alpha = 0$
- `{dc1a}` (in/scalar) – change of lift coefficient with alpha [$1/rad$]

7.4 alpha(acz,TAS,T,p)

Syntax: `alpha(acz,TAS,T,p) {alpha} {acz} {TAS} {T} {p} {w1} {c10} {dc1a}`

Description: Angle of attack from vertical acceleration and TAS using air temperature and static pressure.

Parameters:

- `{alpha}` (out/series) – angle of attack [deg]
- `{acz}` (in/series) – vertical acceleration [m/s^2]
- `{TAS}` (in/series) – true airspeed [m/s]
- `{T}` (in/series) – air temperature [C]
- `{p}` (in/series) – static pressure [hPa]
- `{w1}` (in/scalar) – wing loading [kg/m^2]
- `{c10}` (in/scalar) – lift coefficient at $\alpha = 0$
- `{dc1a}` (in/scalar) – change of lift coefficient with alpha [$1/rad$]

7.5 AMG2LL

Syntax: `AMG2LL {lat} {lon} {easting} {northing} {zone}`

Description: Conversion of Australian Map Grid (AMG) Coordinates to Latitude/Longitude using Redfearn's Formula.

Parameters:

- `{lat}` (out/series) – latitude [deg]
- `{lon}` (out/series) – longitude [deg]
- `{easting}` (in/series) – AMG easting [m]
- `{northing}` (in/series) – AMG northing [m]
- `{zone}` (in/series) – AMG zone number

7.6 AppendSym

Syntax: `AppendSym {out} {in} {nleft} {nright} {type}`

7.7 AppendVal

Syntax: `AppendVal {out} {in} {nleft} {nright} {value}`

Description: Append symmetric or constant value to either or both ends of a series.

Parameters:

- `{out}` (out/series) – output series

{in} (in/series) – input series
{nLeft} (in/scalar) – number of elements to append to left end (beginning)
{nRigth} (in/scalar) – number of elements to append to left end (end)
{type} (keyword) – type of symmetry to apply (PLANE/POINT)
{value} (in/scalar) – value to set to

7.8 Area

Syntax: Area {area} {x} {y}

Description: Calculate area enclosed by polygon.

Parameters:

{area} (out/scalar) – Area in area units equivalent to length units of {x} and {y}
{x}, {y} (in/series) – series containing (x,y) coordinates of corner points

7.9 Atan2

Syntax: Atan2 {out} {x} {y}

Description: Calculate the arctangent of y/x (principal value of the argument of the complex number (x,y)).

Parameters:

{out} (out/series) – output series
{x} (in/series) – x series
{y} (in/series) – y series

7.10 Bolton(P,T,Q)

Syntax: Bolton(P,T,Q) {TH} {THE} {THes} {Xu} {Xs} {Tlc1} {Plc1} {p} {t} {q}

Description: Potential, Equivalent and Saturated Equivalent Potential Temperatures, Unsaturated and Saturated Water Vapour Mixing Ratios, Temperature and Pressure at the Lifting Condensation Level from Pressure, Temperature and Specific Humidity.

Parameters:

{TH} (out/series) – potential temperature [C]
{THE} (out/series) – equivalent potential temperature [C]
{THes} (out/series) – saturated equivalent potential temperature [C]
{Xu} (out/series) – unsaturated water vapour mixing ratio [kg/kg]
{Xs} (out/series) – saturated water vapour mixing ratio [kg/kg]
{Tlc1} (out/series) – temperature [C] at lifting condensation level
{Plc1} (out/series) – pressure [hPa] at lifting condensation level
{p} (in/series) – pressure [hPa]
{t} (in/series) – temperature [C]
{q} (in/series) – specific humidity [g/kg]

Notes: This command is based on the derivations found in David Bolton's 1980 paper "The Computation of Equivalent Potential Temperature", in vol 108 of the Monthly Weather Review. Similar equations may be found on pages 547 & 551 of Stull's "An Introduction to Boundary Layer Meteorology" 1988.

7.11 Breakpoint

Syntax: Breakpoint

Description: Interrupt execution of commands and wait for confirmation of a prompt from the console.

7.12 Bulkfluxes

Syntax: BulkFluxes {out} {Zu} {Zt} {Zsst} {ws} {sst} {atb} {qq} {pp} {Zi} {Sin} {Lin} {lon} {lat} {time} {warm} {cool} {option} {Zref}

Description: RAMF implementation of the COARE bulk fluxes routine.

Parameters:

{out} (out/series) – output series (depends on {option}; see below)
 {Zu} (in/series) – altitude of wind sensor [*m*]
 {Zt} (in/series) – altitude of temperature and humidity sensors [*m*]
 {Zsst} (in/series) – depth of SST sensor [*m*]
 {ws} (in/series) – wind speed [*m/s*]
 {sst} (in/series) – sea surface temperature [*C*]
 {atb} (in/series) – air temperature [*C*]
 {qq} (in/series) – specific humidity [*g/kg*] or relative humidity [%]
 {pp} (in/series) – pressure [*hPa*] (for surface data, assume 1008 hPa is data unavailable)
 {Zi} (in/series) – depth of boundary layer [*m*] (assume 600m, if data unavailable)
 {Sin} (in/series) – short-wave incoming radiation [*W/m²*]
 {Lin} (in/series) – long-wave incoming radiation [*W/m²*]
 {lon} (in/series) – longitude [*deg*]
 {lat} (in/series) – latitude [*deg*]
 {time} (in/series) – time (GMT in hhmmss.ssss)
 {warm} (keyword) – YES/NO – warm layer correction
 {cool} (keyword) – YES/NO – cool skin correction
 {option} (keyword) – Output type, one of:
 QH – Sensible heat flux [*W/m²*]
 QE – Latent heat flux [*W/m²*]
 TAU – Momentum flux [*N/m²*]
 RainF – Heat flux due to rain [*W/m²*]
 TauR – Momentum flux due to rain [*N/m²*]

Ustar – u_* [m/s]

Qstar – q_* [kg/kg]

Tstar – θ_* [C]

CD – Drag coefficient

CH – Transfer coefficient for heat

CE – Transfer coefficient for moisture

RR – Roughness Reynolds number

ZL – [ht/L] where L is the Obukhov length

ZO – Roughness length [m]

T0 – Skin sea surface temperature [C], for ship data

WBAR – Webb mean vertical velocity [m/s]

Uref – Wind speed at altitude Z_{ref} [m/s], using neutral profile

Tref – Air temperature at altitude Z_{ref} [C], using neutral profile

Qref – Specific humidity at altitude Z_{ref} [g/kg], using neutral profile

dtCool – Magnitude of cool skin correction [C], for ship data

dtWarm – Magnitude of warm layer correction [C], for ship data

tkPWP – Ocean mixed layer depth [C], for ship data

{Zref} (in/scalar) – altitude at which to calculate U_{ref} , T_{ref} , Q_{ref} [m] (default=10m)

Notes:

Zsst, **warm** and **cool** are included for working with ship data set to 0, NO and NO for aircraft data.

COARE bulk flux version 2.6a was made from 2.5b by Frank Bradley 3/12/99 Replace 20 iterations with Grachev and Fairall first guess z/L routine J. App. Meteor. 36, 406-414, 1997

Add Chris' mods as used in Jasmine of rr-zoq, charnock, cool skin

Untangle some complexity which had crept into 2.5b Use separate phiu, phit functions with Grachev et al (2000) constants Stable functions according to Beljaars and Holtslag J. Appl. Meteor. 30, 327-341, 1991 Return Webb Wbar because Webb flux correction already included in QE

RAMF implementation based on v2.6a fortran code (SM2000), modified from previous v2.5b code.

Some minor changes made to adapt to **RAMF** style and improve readability. All outputs listed above have been tested against the original fortran code. and Chris Fairall's Matlab version using the 'test2_5b.dat' file as input. Code and test files can be found at

<ftp://ftp.etl.noaa.gov/et7/users/cfairall/bulka/g/>

7.13 CalculateTimeStep

Syntax: CalculateTimeStep {dt} {in} {epsilon}

Description: Calculate average timestep from time tag series. Issue warnings for excessive deviation from the average value, constant time values and negative time steps.

Parameters:

{dt} (out/scalar) – average timestep

{in} (in/series) – input series

{epsilon} (in/scalar) – tolerable variations in % of average timestep

7.14 Clear

Syntax: Clear {name_1} {name_2} {name_3} ... {name_n}

or

Syntax: Clear All

or

Syntax: Clear AllBut {name_1} {name_2} {name_3} ... {name_n}

Description: Clear selected series and release their allocated memory.

Parameters:

{name_i} (in/series) – series to clear (wildcard specification using '*' and '?' allowed)

7.15 ClipSer

Syntax: ClipSer {out} {in} {lower} {upper}

Description: Clip series, i.e. limit values of series between {lower} and {upper} limit.

Parameters:

{out} (out/series) – output series

{in} (in/series) – input series

{lower} (in/scalar) – lower boundary value

{upper} (in/scalar) – upper boundary value

7.16 CombineCSI

Syntax: CombineCSI {out} {overlap} {in1} {in2} {operator}

Description: Combine two conditional sampling indicators using the rule {out}=1 IF {in1}=1 {operator} {in2}=1. Also outputs the fraction of overlapping elements (those that have value "1" in both indicators), into the constant /codeoverlap.

Parameters:

{out} (out/series) – output series

{overlap} (out/scalar) – overlap fraction

{in1} (in/series) – first input series

{in2} (in/series) – second input series

{operator} (keyword) – one of (AND,OR,EXOR)

7.17 Combine

Syntax: Combine {out} {in1} {in2} {crossover}

Description: Combine two ranges of the same sensor

Parameters:

{out} (out/series) – output series

{in1} (in/series) – first input series (smaller values)

{in2} (in/series) – second input series (larger values)

{crossover} (in/scalar) – crossover threshold

7.18 Covariance

Syntax: Covariance {cov} {series1} {series2}

Description: instantaneous covariances

Parameters:

{cov} (out/series) – covariances

{series1} (in/series) – first series

{series2} (in/series) – second series

7.19 CovShift

Syntax: CovShift {cov} {label1} {label2} {nlags}

Description: Covariance with shifting

Parameters:

{cov} (out/series) – covariance

{label1} (in/series) – first series

{label2} (in/series) – second series

{nlags} (in/scalar) – number of shifts

Notes: If $l1 = label1 - mean$ and $l2 = label2 - mean$, then $cov(j) = average[l1(i) * l2(i+j), i = 1...N-j]$ for $j = 0...nlags$. The new series has length $nlags+1$, with the 1st element containing the covariance with zero shifting. If $label1 = label2$, auto-covariances are computed, and the 1st element is then the variance. To get cross/auto-correlations, divide the output of CovShift by $sd(x) * sd(y)$. Zero lag should then give approximately 1.0.

7.20 CreateLd

Syntax: CreateLd {series} {gradient} {from} {to}

Description: Create series using gradient between start and end value

Parameters:

{series} (out/series) – output series

{gradient} (in/scalar) – change in function value per element

{from} (in/scalar) – first value

{to} (in/scalar) – last value

7.21 CreateLn

Syntax: CreateLn {series} {nele} {from} {to}

7.22 CreateLs

Syntax: CreateLs {series} {template} {from} {to}

Description: create series using number of elements (explicit or from template series) between start and end value

Parameters:

{series} (out/series) – output series

{nele} (in/scalar) – number of elements

{template} (in/series) – template series

{from} (in/scalar) – first value

{to} (in/scalar) – last value

7.23 CreatePn

Syntax: CreatePn {series} {nele} {a0} {a1} {a2} {a3} {a4}

7.24 CreatePs

Syntax: CreatePs {series} {template} {a0} {a1} {a2} {a3} {a4}

Description: Create series from polynomial (maximum order: 4) using explicit number of elements or template series

Parameters:

{series} (out/series) – output series

{nele} (in/scalar) – number of elements

{template} (in/series) – template series

{a0}..{a4} (in/scalar) – coefficients of polynomial

7.25 DateSince

Syntax: DateSince {DDMMYY} {HHMMSS} {days.dec} {DDMMYYref}

Description: Convert time in decimal days from beginning of reference year into date and time in 6-digit format

Parameters:

{DDMMYY} (out/series) – 6 digit date

{HHMMSS} (out/series) – 6 digit time

{days.dec} (in/series) – decimal days since midnight on {DDMMYYref}

{DDMMYYref} (in/scalar) – 6 digit reference date (default = 010100)

7.26 DecDaysSince

Syntax: DecDaysSince {days.dec} {DDMMYY} {HHMMSS} {DDMMYYref}

Description: Convert time in decimal days from beginning of reference year into date and time in 6-digit format

Parameters:

{days.dec} (out/series) – decimal days since midnight on {DDMMYYref}
 {DDMMYY} (in/series) – 6 digit date
 {HHMMSS} (in/series) – 6 digit time
 {DDMMYYref} (in/scalar) – 6 digit reference date (default = 010100)

7.27 DEMevaluate

Syntax: DEMevaluate {alt} {lat} {lon} {filename} {maxLat} {minLon} {sizex} {sizey} {dlat} {dlon} {byteorder}

Description: Generate altitude data for lat/lon coordinates using a DEM file in 16 bit BIL format

Parameters:

{alt} (out/series) – altitude [*m*]
 {lat} (in/series) – latitude [*deg*]
 {lon} (in/series) – longitude [*deg*]
 {filename} name of DEM file; if set to 'AUTO', the default DEM at /data/dem/gtopo30 is used and the following parameters do not need to be specified
 {maxLat} (in/scalar) – latitude of first datum in DEM file
 {minLon} (in/scalar) – longitude of first datum in DEM file
 {sizex} (in/scalar) – horizontal dimension of DEM file
 {sizey} (in/scalar) – vertical dimension of DEM file
 {dlat} (in/scalar) – resolution of DEM latitude [*deg/pixel*]
 {dlon} (in/scalar) – resolution of DEM longitude [*deg/pixel*]
 {byteorder} (keyword) – 'M' for Motorola, 'I' for Intel

7.28 Detrend

Syntax: Detrend {out} {in} {type}

7.29 DetrendSpec

Syntax: Detrend {out} {in} {a0} {a1} {a2} {a3} {a4}

Description: Remove trend from series assuming constant spacing of elements. Detrend automatically finds linear or quadratic trend. DetrendSpec uses a specified polynomial.

Parameters:

{out} (out/series) – output series
 {in} (in/series) – input series
 {type} one of LINEAR or QUADRATIC

{a0}...{a4} (in/scalar) – coefficients of polynomial

7.30 Diff

Syntax: Diff {diff} {series} {delta}

7.31 FwdDiff

Syntax: FwdDiff {diff} {series} {delta}

Description: Differentiate series using centered or forward differences

Parameters:

{diff} (out/series) – differentiated series

{in} (in/series) – input series

{delta} (in/scalar) – spacing of elements

7.32 DiffTag

Syntax: DiffTag {out} {in} {tag}

Description: Differentiate series with irregular spacing using forward differences.

Parameters:

{out} (out/series) – output series

{in} (in/series) – input series

{tag} (in/series) – tag series

7.33 DirnHelix

Syntax: DirnHelix {out} {in} {jump}

Description: Helix removal of 360/0 degree jumps

Parameters:

{out} (out/series) – output series

{in} (in/series) – input series

{jump} (in/scalar) – permissible jump threshold (default=180)

7.34 Dist(lat)

Syntax: Dist(lat) {dist} {lat} {lat0}

7.35 Dist(lon)

Syntax: Dist(lon) {dist} {lon} {lat0} {lon0}

Description: Distance from reference point in metres from latitude/longitude in decimal degrees.

Parameters:

{dist} (out/series) – distance [*m*]

{lat} (in/series) – latitude [*deg*]

{lon} (in/series) – longitude [*deg*]

{lat0} (in/scalar) – latitude of reference point [*deg*]

{lon0} (in/scalar) – longitude of reference point [*deg*]

7.36 Div0

Syntax: Div0 {out} {series1} {series2} {value} {epsilon}

Description: Division of series1 by series2, setting the result to value where series2 is below epsilon threshold.

Parameters:

{out} (out/series) – output series
 {series1},{series2} (in/series) – input series
 {value} (in/scalar) – override value (default=C_invalid)
 {epsilon} (in/scalar) – threshold (default=C_eps)

7.37 EleAve

Syntax: EleAve {out} {series1} {series2} ... {seriesn}

7.38 EleSD

Syntax: EleSD {out} {series1} {series2} ... {seriesn}

Description: Element-wise statistics (average and standard deviation of multiple series (e.g. the first element of {out} is assigned the average of the first elements of the input series).

Parameters:

{out} (out/series) – output series
 {seriesi} (in/series) – the series to calculate statistics for

7.39 EleNum

Syntax: EleNum {out} {series} {value} {tolerance}

Description: Checks a series for the occurrence of a given value, and if present, will return the element numbers of the occurrences of that number within the series.

Parameters:

{out} (out/series) – indices of elements of series within {tolerance} of {value}
 {value} (in/scalar) – reference value
 {tolerance} (in/scalar) – tolerance

7.40 Faster

Syntax: Faster {out} {in} {factor}

Description: Increase sampling rate by linear interpolation.

Parameters:

{out} (out/series) – output series
 {in} (in/series) – input series
 {factor} (in/scalar) – integer increment factor

7.41 FFT

Syntax: FFT {freq} {real} {imag} {in} {dt}

Description: Computes a pure fourier transform of {in} using a Fast Fourier Transform (FFT) algorithm.

Parameters:

{freq} (out/series) – Frequency of the (complex) Fourier transform in units of $1/\{dt\}$. First frequency is 0.0, and the total power is stored in first element of the {real} series.

{real} (out/series) – Real part of the complex Fourier transform

{imag} (out/series) – Imaginary part of the complex Fourier transform

{in} (in/series) – input series

{dt} (in/scalar) – data interval of input series

Notes: No windowing, tapering or overlapping of segments is performed. Consequently, computation of stable power- or cross-spectra should therefore be done using the SPECTRA command instead.

With

$$F_{mag} = \sqrt{real^2 + imag^2} \text{ (Magnitude of Fourier coefficients)}$$

$$N_f = \text{number of elements in } F_{mag}$$

$$N_s = \text{number of elements on input series}$$

the relationship of Fourier Coefficients to the power spectrum and the variance of the input series is

$$Pwr_S = \frac{F_{mag}^2}{N_f(N_s-1)} \text{ (Power Spectrum of Input Series)}$$

$$Var_S = \sum Pwr_S$$

7.42 FiltInt

Syntax: FiltInt {output} {badstart} {badlength} {input} {del} {weight}

Description: Rejection of noise spikes in a series using a filtered interpolation scheme.

Parameters:

{output} (out/series) – despiked series

{badstart} (out/series) – starting element number of bad data blocks

{badlength} (out/series) – length of bad data blocks

{input} (in/series) – series to be despiked

{del} (in/series) – allowed change between points

{weight} (in/scalar) – weight for filter scheme, 0.0 to 1.0

Notes: The routine works by checking each point to see if it lies within allowable limits of the previous point. If the current point lies within the limits then it is assumed to be good data, passed for output and the filtered interpolation scheme updated using this value. If the current point lies outside the limits then it is assumed to be bad data, rejected and a predicted value from the filtered interpolation scheme is substituted.

The routine returns the starting position and length of any bad data blocks. If there are no bad data blocks, the starting position and length are set to 0. The maximum length should be checked after the routine has been used since long segments (say ≥ 3 points) may indicate that the filtered interpolation scheme has become unstable.

This routine was written to deal with noise spiking on the Honeywell LaserNav during APSS4. It uses the TREND function given in Appendix G of Leise and Masters. Relatively smooth data, eg horizontal velocity components respond well and the routine can be used with narrow limits (small $\{del\}$) and little filtering ($\{weight\} \geq 0.5$) without becoming unstable. More rapidly varying data eg roll, heading require wider limits and stronger filtering ($WT = 0.25$) to prevent the rejection/interpolation scheme from becoming unstable.

7.43 GetTimeStep

Syntax: GetTimeStep $\{dt\}$ $\{TimeTag\}$ $\{epsilon\}$ $\{option\}$

Description: Calculate average timestep from TimeTag channel, with checks for timestep changes within the series.

Parameters:

$\{dt\}$ (out/scalar) – average time step
 $\{TimeTag\}$ (in/series) – TimeTag series in seconds from start time. Must be monotonously increasing
 $\{epsilon\}$ (in/scalar) – tolerable variation in % of time step between 1. and 2. sample
 $\{option\}$ one of CONTINUE,IGNORE,CRASH

CONTINUE continue if the epsilon condition is violated; issue warning for each violation.

IGNORE continue if the epsilon condition is violated; issue a single warning if the epsilon condition has been violated.

CRASH terminate with an error exit if the epsilon condition is violated

7.44 GKX,GKY(LAT,LOX)

Syntax: GKX,GKY(LAT,LOX) $\{gkx\}$ $\{gky\}$ $\{lat\}$ $\{lon\}$ $\{reflon\}$

Description: Convert latitude/longitude coordinates to Gauss-Krüger coordinates (based on subroutine PHLA2G by Wenzel Brücher, IGM, Cologne University)

Parameters:

$\{gkx\}, \{gky\}$ (out/series) – Gauss-Krueger coordinates [m]
 $\{lat\}, \{lon\}$ (in/series) – Latitude/longitude coordinates [deg]
 $\{reflon\}$ (in/scalar) – Reference longitude [deg]

7.45 GPStoBATS

Syntax: GPStoBATS $\{Bu\}$ $\{Bv\}$ $\{Bw\}$ $\{Nu\}$ $\{Nv\}$ $\{Nw\}$ $\{Prt\}$ $\{Rrt\}$ $\{Yrt\}$ $\{x\}$ $\{y\}$ $\{z\}$

Description: Infer BAT probes' motion from motion of fuselage-mounted GPS antenna

Parameters:

- `{Bu}` (out/series) – longitudinal Probe’s motion in aircraft system (m/s)
- `{Bv}` (out/series) – lateral Probe’s motion in aircraft system (m/s)
- `{Bw}` (out/series) – vertical Probe’s motion in aircraft system (m/s)
- `{Nu}` (in/series) – longitudinal Antenna’s motion in aircraft system (m/s)
- `{Nv}` (in/series) – lateral Antenna’s motion in aircraft system (m/s)
- `{Nw}` (in/series) – vertical Antenna’s motion in aircraft system (m/s)
- `{Prt}` (in/series) – Pitch rate (+ve: nose rising) (deg/s)
- `{Rrt}` (in/series) – Roll rate (+ve: right wing dropping) (deg/s)
- `{Yrt}` (in/series) – Yaw rate (+ve: nose turning right) (deg/s)
- `{x}` (in/scalar) – x-Distances FROM antennea TO probes (m)
- `{y}` (in/scalar) – y-Distances FROM antennea TO probes (m)
- `{z}` (in/scalar) – z-Distances FROM antennea TO probes (m)

Note: Aircraft system is : x, y, z, +ve is forward, right, and upwards, respectively.

7.46 hhmssh(sec)

Syntax: `hhmssh(sec) {hhmssh} {sec} {StartTime}`

Description: Time in 8 digit format from decimal time in seconds

Parameters:

- `{hhmssh}` (out/series) – Time in hours minutes seconds and 100th of seconds
- `{sec}` (in/series) – time series in seconds
- `{StartTime}` (in/scalar) – start time in seconds

7.47 Histogram

Syntax: `Histogram {histgr} {midpts} {series} {start} {end} {step}`

Description: Form an amplitude histogram from `{series}`, giving the fraction of the whole series falling into a number of amplitude categories.

Parameters:

- `{histgr}` (out/series) – histogram (units: fraction of 1)
- `{midpts}` (out/series) – midpoints of intervals
- `{series}` (in/series) – input series
- `{start}, {end}` (in/scalar) – starting and ending values for histogram
- `{step}` (in/scalar) – width of amplitude intervals

7.48 HSI2RGB

Syntax: `HSI2RGB {r} {g} {b} {h} {s} {i}`

Description: (red/green/blue) colour descriptor from (hue/saturation/intensity) colour descriptor

Parameters:

{r} (out/series) – red component [0 – 1]
 {g} (out/series) – blue component [0 – 1]
 {b} (out/series) – green component [0 – 1]
 {h} (out/series) – hue [0 – 360]
 {s} (out/series) – saturation [0 – 1]
 {v} (out/series) – intensity [0 – 1]

7.49 Integrate3

Syntax: Integrate3 {ix} {iy} {iz} {x} {y} {z} {x0} {y0} {z0} {dt}

7.50 Int3Tag

Syntax: Int3Tag {ix} {iy} {iz} {x} {y} {z} {tag} {x0} {y0} {z0}

Description: Simultaneous integration of 3 series (for example 3D-accelerations to 3D-velocities) with a common tag series or an explicit data interval

Parameters:

{ix} (out/series) – first integrated series
 {iy} (out/series) – second integrated series
 {iz} (out/series) – third integrated series
 {x} (in/series) – first series to be integrated
 {y} (in/series) – second series to be integrated
 {z} (in/series) – third series to be integrated
 {tag} (in/series) – tag series
 {x0} (in/scalar) – starting value for first integration
 {y0} (in/scalar) – starting value for second integration
 {z0} (in/scalar) – starting value for third integration
 {dt} (in/scalar) – data interval

7.51 InvFFT

Syntax: InvFFT {out} {real} {imag}

Description: Computes an inverse Fourier transform.

Parameters:

{out} (out/series) – output series
 {real} (in/series) – real part of input series
 {imag} (in/series) – imaginary part of input series

Note: Because the FFT requires a series of length equal to a power of 2, the input series is padded with 0.0 to the next power of 2. Thus, when an Inverse Fourier Transform is performed, the series produced will be longer than the original series, and will need to be truncated by the user back to its original length. If some manipulation of the coefficients has been performed in between, the unwanted trailing section of the final series may not be uniformly zero.

7.52 Interleave

Syntax: Interleave {out} {series1} {series2} ... {seriesn}

7.53 Join

Syntax: Join {out} {series1} {series2} ... {seriesn}

Description: Concatenate or interleave elements of multiple series (max = 50), if the output series is stated as '*', the first input series will be used as output series.

Parameters:

{out} (out/series) – output series
 {seriesi} (in/series) – the series to join/interleave

7.54 Lat(dist)

Syntax: Lat(dist) {lat} {dist} {lat0}

7.55 Lon(dist)

Syntax: Lon(dist) {lon} {dist} {lat0} {lon0}

Description: Latitude or longitude in decimal degrees from distance from reference point in metres.

Parameters:

{lat} (out/series) – latitude [*deg*]
 {lon} (out/series) – longitude [*deg*]
 {dist} (in/series) – distance [*m*]
 {lat0} (in/scalar) – latitude of reference point [*deg*]
 {lon0} (in/scalar) – longitude of reference point [*deg*]

7.56 LengthsCSI

Syntax: LengthsCSI {lengths} {start} {frac} {indicator} {dt} {value}

Description: Determine the lengths, starting indices and total fraction of the segments of series {indicator} with value equal to {value}. The length of a segment is calculated as (number of elements * {dt}).

Parameters:

{lengths} (out/series) – lengths of segments
 {start} (out/series) – starting points (units of {DataInt})
 {frac} (out/scalar) – fraction of elements of {indicator} which have the value {value}
 {indicator} (in/series) – input series
 {dt} (in/scalar) – data interval of {indicator}
 {value} (in/scalar) – value to check for

7.57 LL2AMG

Syntax: LL2AMG {easting} {northing} {zone} {lat} {lon}

Description: Conversion of Latitude/Longitude (decimal degrees) to Australian Map Grid (AMG) coordinates using Redfearn's formula.

Parameters:

{easting} (out/series) – AMG easting [*m*]
 {northing} (out/series) – AMG northing [*m*]
 {zone} (out/series) – AMG zone number
 {lat} (in/series) – latitude [*deg*]
 {lon} (in/series) – longitude [*deg*]

7.58 MatchEle

Syntax: MatchEle {out} {s1} {s2}

Description: For each value in s2, the element index of the closest value contained in series s1 is returned. If s1 contains 2 or more values equally far from the value in s2, the index of its first occurrence is returned.

Parameters:

{out} (out/series) – indices of elements of s1
 {s1} (in/series) – series to create index for
 {s2} (in/series) – series of values to look up in s1

7.59 Merge

Syntax: Merge {merged} {fast} {slow} {dt} {mergetime}

Description: Merge fast/unstable series with slow/stable one. The running averages of the fast series over time mergetime are adjusted to the running averages of the slow series.

Parameters:

{merged} (out/series) – merged series
 {fast} (in/series) – fast input series
 {slow} (in/series) – slow input series
 {dt} (in/scalar) – sampling interval
 {mergetime} (in/scalar) – time interval for merging

7.60 mHDG(tHDG,lat,lon,ht)

Syntax: mHDG(tHDG,lat,lon,ht) {mHDG} {tHDG} {lat} {lon} {ht} {year} {CoSys} {Accuracy}

Description: Magnetic heading (deg) from true heading (deg), calculating the magnetic deviations from latitude (decimal degrees), longitude (decimal degrees), and height above MSL (km).

Parameters:

{mHDG} (out/series) – magnetic heading [*deg*]

- {tHDG} (in/series) – true heading [*deg*]
 {lat} (in/series) – latitude [*deg*]
 {lon} (in/series) – longitude [*deg*]
 {ht} (in/series) – height above MSL [*km*]
 {year} (in/scalar) – decimal year (eg. 1992.14 for February 20, 1992). Must be in the range 1990-1995
 {CoSys} (keyword) – geocentric or geodetic
 {Accuracy} (keyword) – high or low
- Note:** if "high", magnetic variation is computed for every element (this is SLOW!) if "low", magnetic variation is computed from the average of the first and last elements (accurate enough for most purposes).

7.61 MinMax0X

Syntax: MinMax0X {min} {max} {in}

Description: Creates series of local minimum and maximum values of a series, based on segments defined by zero crossings in label. Output series are "padded" to same length as input series by linear interpolation between the located extrema, i.e. the result series are first order envelopes of the input series.

Parameters:

- {min} (out/series) – maximum envelope series
 {max} (out/series) – minimum envelope series
 {in} (in/series) – input series

7.62 ModifyCSI

Syntax: ModifyCSI {NewInd} {OldInd} {Govern} {Lower} {Upper}

Description: Modify a 0/1 "conditional sampling" indicator

Parameters:

- {NewInd} (out/series) – new indicator series to be formed
 {OldInd} (in/series) – indicator series to be modified
 {Govern} (in/series) – governing series
 {Lower} (in/scalar) – lower threshold value
 {Upper} (in/scalar) – upper threshold value

Rules: The mean value of {govern} is calculated for each +1 segment of {OldInd}. If the mean value of a segment does NOT fall between {lower} and {upper} (inclusive of the limits), that segment is set to 0 in {NewInd}.

Notes: A single thresholding technique can be obtained by setting the appropriate threshold to some ridiculous value.

7.63 Monotony

Syntax: Monotony {series}

Description: Test series to be monotonously ascending – issue warning if constant, abort if descending

Parameters:

{series} (in/series) – series to check for monotony

7.64 NormEle

Syntax: NormEle {out} {in} {index}

Description: Normalize series with value of element number `index`.

Parameters:

{out} (out/series) – normalized series

{in} (in/series) – input series

{index} (in/scalar) – index of element to normalize with (default=1), if a negative index is given, the series will be normalized with its maximum value.

7.65 Polynomial

Syntax: Polynomial {out} {in} {a0} {a1} {a2} {a3} {a4}

Description: Calculate polynomial $out = a4 * in^4 + a3 * in^3 + a2 * in^2 + a1 * in + a0$

Parameters:

{out} (out/series) – output series

{in} (in/series) – input series

{a0..a4} (in/scalar) – coefficients of polynomial (default=0)

7.66 QFF(z,p,Tv)

Syntax: QFF(z,p,Tv) {QFF} {z} {p} {Tv} {dtdz}

Description: QFF from altitude, pressure and virtual temperature.

Parameters:

{QFF} (out/series) – QFF

{z} (in/series) – altitude [m]

{p} (in/series) – pressure [hPa]

{Tv} (in/series) – virtual temperature [degC]

{dtdz} (in/scalar) – vertical temperature gradient [K/m]

7.67 QNH(p,Tv,z)

Syntax: QNH(p,Tv,z) {QNH} {p} {Tv} {z}

Description: Static pressure corrected to MSL

Parameters:

{QNH} (out/series) – QNH [hPa]

{p} (in/series) – pressure [hPa]

{Tv} (in/series) – virtual temperature [degC]

{z} (in/series) – altitude [m]

7.68 RandomSeries

Syntax: RandomSeries {series} {nElements} {cmul} {cadd}

Description: Generate random series ($random[0,1] * cmul + cadd$)

Parameters:

{out} (out/series) – output series
 {nElements} (in/scalar) – number of elements
 {cadd} (in/scalar) – additive constant
 {cmul} (in/scalar) – multiplicative constant

7.69 Response

Syntax: Response {corr} {series} {dt} {rt1} {a} {rt2}

Description: McCarthy algorithm for correcting time series data exhibiting response functions with 2 exponential drop-offs (e.g. fast temperature sensor in a protective housing). Reference: McCarthy (1973) JAM vol 12.

Parameters:

{DDMMYY} (out/series) – 6 digit date
 {HHMMSS} (out/series) – 6 digit time
 {days.dec} (in/series) – decimal days since midnight on {DDMMYYref}
 {DDMMYYref} (in/scalar) – 6 digit reference date (default = 010100)
 {corr} (out/series) – corrected series
 {series} (in/series) – series to be corrected
 {dt} (in/scalar) – time step of input series [s]
 {rt1} (in/scalar) – response time 1 [s]
 {a} (in/scalar) – relative weighting coefficient to be attributed to response time rt1 ($0 < a < 1$)
 {rt2} (in/scalar) – response time 2 (set to 0 for single exponential correction)

7.70 Reverse

Syntax: Reverse {out} {in}

Description: Reverse series sequence (i.e. first element to last one, etc.).

Parameters:

{out} (out/series) – output series
 {in} (in/series) – input series

7.71 RevMeteolab

Syntax: RevMeteolab {Tref} {V} {T}

Description: Get Tref backwards through meteolab for calibration.

Parameters:

{Tref} (out/series) – derived reference temperature [C]
 {V} (in/series) – voltage at sensor [V]

{T} (in/series) – actual temperature [C] from other source

7.72 Rot2D

Syntax: Rot2D {xnew} {ynew} {xold} {yold} {angle}

Description: Rotate 2D coordinates anticlockwise about the origin.

Parameters:

{xnew} {ynew} (out/series) – new (x,y) coordinates

{xold} {yold} (in/series) – original (x,y) coordinates

{angle} (in/scalar) – rotation angle [*deg*]

7.73 RandomSeed

Syntax: RandomSeed {seed}

Description: Initialize the pseudo random sequence generator at a fixed point. If this command is not used, the random generator will be initialized using the system time at program startup.

Parameters:

{seed} (in/scalar) – seed value (real number $\neq 0$)

7.74 RunInt

Syntax: RunInt {int} {series} {dt} {c0}

Description: Running integration of series.

Parameters:

{out} (out/series) – output series

{seriesi} (in/series) – the series to join/interleave

{int} (out/series) – integrated series

{series} (in/series) – input series

{dt} (in/scalar) – data spacing

{c0} (in/scalar) – starting value

7.75 RunningCovar

Syntax: RunningCovar {runcov} {label1} {label2} {slen} {jump} {detrend_opt}

Description: Running covariance (flux) of label1 vs label2 over intervals of length slen (NOTE: must be odd). Each segment of length slen is first detrended, before computing the covariance.

Parameters:

{runcov} (out/series) – running covariance series of {label1} vs {label2}

{label1} (in/series) – first input series for covairance calculation

{label2} (in/series) – second input series for covariance calculation

{slen} (in/scalar) – length of intervals

{jump} (in/scalar) – parameter specifying how many elements to "jump" between estimates of the covariance

{detrend_opt} (keyword) – "quadratic" (default), "linear" or "none" (just remove mean)

Note: If the {slen} given is even, this menu will set it to slen-1. The "padding" scheme given above will then give a result which is too large by 1 element. If {jump} = 0, jump 1 element at a time, and pad the start and end of the covariance series with the value of the first/last actual covariance calculated. In this case, {runcov} will have the SAME LENGTH as the original series. If {jump} > 0, jump by {jump} elements after each covariance calculation. In this case, {runcov} will have a length of M+1 elements, where $M = INT((N - slen)/jump)$ and N is the length of the original series. To make {runcov} series have the same length as the original series, use the menu "faster" with a factor of {jump}, and remove the last OVERLP elements, where $OVERLP = jump - 1$. Then add NEDGE elements to the start and NEDGE+NWASTE elements to the end of {runcov}, where: $NEDGE = (slen - 1)/2$, $NWASTE = N - slen - M * jump$.

7.76 RunningMean

Syntax: RunningMean {runmean} {series} {slen} {jump}

7.77 RunningStDev

Syntax: RunningStDev {runstdev} {series} {slen} {jump}

7.78 RunningMin

Syntax: RunningMin {runmin} {series} {slen} {jump}

7.79 RunningMax

Syntax: RunningMax {runmax} {series} {slen} {jump}

Description: Running statistics over arbitrary interval len, jumping 1 or more elements each time.

Parameters:

{runmean} (out/series) – running mean

{runstdev} (out/series) – running standard deviation

{runmin} (out/series) – running minimum

{runmax} (out/series) – running maximum

{series} (in/series) – input series

{slen} (in/scalar) – length of sampling interval

{jump} (in/scalar) – spacing of interval application

Note: jump > 0 shift interval by jump from one calculation to the next, the resulting series will have the length $INT((InputLength-slen)/jump) + 1$ same as above, but pad the start and end of the result to have the same length as the input series

7.80 RunSum

Syntax: RunSum {sum} {series}

Description: Running sum of series elements

Parameters:

{sum} (out/series) – output series
{series} (in/series) – input series

7.81 sec(hhmmssh)

Syntax: sec(hhmmssh) {sec} {hhmmssh}

Description: Time in seconds after midnight from time in hhmmssh check if series starts pm or am, to take switch of time at midnight into account

Parameters:

{sec} (out/series) – time in decimal seconds
{hhmmssh} (in/series) – time in 8-digit format

7.82 Select

Syntax: Select {out} {in} {first} {last} {step}

Description: Select a subrange from a series.

Parameters:

{out} (out/series) – output series
{in} (in/series) – input series
{first} (in/scalar) – first element to select (default = beginning of series)
{last} (in/scalar) – last element to select (default = end of series)
{step} (in/scalar) – spacing of elements to select (default=1)

7.83 SelS1(S2)

Syntax: SelS1(S2) {out} {s1} {s2}

Description: Select elements of series 1 using indices of desired elements contained in series 2. Negative indices will be counted from the end of the input series.

Parameters:

{out} (out/series) – series of selected elements
{s1} (in/series) – series to select from
{s2} (in/series) – series of indices to select

7.84 SetS1(S2)

Syntax: SetS1(S2) {out} {s1} {s2} {value}

Description: Sets the elements of S1 indexed by the elements of S2 to a fixed value.

Parameters:

{out} (out/series) – output series
{s1} (in/series) – series containing the original values
{s2} (in/series) – series of indices of elements to set to **value**
{value} (in/scalar) – value to set indicated elements to

7.85 Shift

Syntax: `Shift {out} {in} {n}`

Description: Chop off $\text{abs}(n)$ elements from beginning or end and extend series to original length using plane symmetry at the other end.

Parameters:

`{out}` (out/series) – output series
`{in}` (in/series) – input series
`{factor}` (in/scalar) – distance to shift (n_i0 : shift towards start, n_e0 : shift towards end)

7.86 Slower

Syntax: `Slower {out} {in} {factor}`

Description: Reduce sampling rate by element-wise integration.

Parameters:

`{out}` (out/series) – output series
`{in}` (in/series) – input series
`{factor}` (in/scalar) – integer reduction factor

7.87 SSTcorTC

Syntax: `SSTcorTC {SST_corr} {SST_unc} {L_in} {Emax} {Emin}`

Description: Correction of radiometric sea surface temperature (SST) for the effects of reflected long wave radiation from the non- blackbody surface of the ocean.

Parameters:

`{Tt}` (out/series) – true temperature
`{Ttm}` (in/series) – measured temperature
`{sp}` (in/series) – static pressure
`{dp}` (in/series) – dynamic pressure
`{r}` (in/scalar) – recovery factor
`{SST_corr}` (out/series) – corrected SST
`{SST_unc}` (in/series) – uncorrected SST from a downward-looking thermoradiometer assuming blackbody spectrum
`{L_in}` (in/series) – long-wave incoming energy from upward-looking pyrgeometer
`{Emax}, {Emin}` (in/scalar) – maximum and minimum values of incoming long wave radiation expected for whole experiment (corresponding to completely clear and completely overcast conditions)

7.88 StripInvalid

Syntax: StripInvalid {series1} {series2}...{seriesn}

Description: Strip elements from all series where any of the series has got an invalid value.

Parameters:

{seriesi} (in/series) – (out/series) – the series to strip of invalid values

7.89 TASinAC

Syntax: TASinAC {Tx} {Ty} {Tz} {TAS} {alpha} {beta} {pchrt} {rllrt} {yawrt} {x} {y} {z}

Description: Compute TAS-components in aircraft system using angle of attack and angle of side slip as measured by a five-hole probe.

Parameters:

{Tx}, {Ty}, {Tz} (out/series) – TAS-components in aircraft system [units as TAS]

{TAS} (in/series) – TAS (true airspeed) [*m/s* or *knots*]

{alpha} (in/series) – angle of attack [*deg*]

{beta} (in/series) – angle of sideslip [*deg*]

{pchrt} (in/series) – pitch rate [*deg/s*]

{rllrt} (in/series) – roll rate [*deg/s*]

{yawrt} (in/series) – yaw rate [*deg/s*]

{x} {y} {z} (in/scalar) – distances between centre of gravity of aircraft and 5-hole probe [*m*, forward/up/right positive]

7.90 TAS(TS,TT)

Syntax: TAS(TS,TT) {tas} {ts} {tt}

Description: True airspeed from static temperature and true temperature

Parameters:

{tas} (out/series) – true airspeed

{ts} (in/series) – static temperature

{tt} (in/series) – true pressure

7.91 TC(T,W,P,AH,RhoC)

Syntax: TC(T,W,P,AH,RhoC) {Tc} {T} {W} {P} {aH} {RhoC}

Description:

Parameters:

{Tc} (out/series) –

{T} (in/series) – temperature [*C*]

{W} (in/series) –

{P} (in/series) – static pressure [*hPa*]

{aH} (in/series) – *H2O* vapour partial density [*g/m³*]

{RhoC} (in/series) – CO2 partial density [mg/m^3]

7.92 TE(T,w,P,aH)

Syntax: TE(T,w,P,aH) {ecor} {H} {T} {wair} {press} {aH}

Description: Provides the 'Sensible heat flux' and 'True Latent Heat Flux' corrected for density effects of both sensible heat, and water vapour.

Parameters:

{ecor} (out/series) – Corrected Latent Heat Flux [W/m^2]

{H} (out/series) – Sensible Heat Flux [W/m^2]

{T} (in/series) – Temperature [C]

{wair} (in/series) – Vertical velocity fluctuations [m/s]

{press} (in/series) – Atmospheric Pressure [hPa]

{aH} (in/series) – H2O vapour partial density [g/m^3]

Note: variation of cp with q, and Lwv with T are also taken into account

7.93 tHDG(mHDG)

Syntax: tHDG(mHDG) {tHDG} {mHDG} {var}

Description: True heading from magnetic heading and correction.

Parameters:

{tHDG} (out/series) – true heading [deg]

{mHDG} (in/series) – magnetic heading [deg]

{var} (in/scalar) – magnetic variation [deg], east positive

7.94 tHDG(mHDG,lat,lon,ht)

Syntax: tHDG(mHDG,lat,lon,ht) {tHDG} {mHDG} {lat} {lon} {ht} {year} {CoSys} {Accuracy}

Description: true heading (deg) from magnetic heading (deg), calculating the magnetic deviations from latitude (decimal degrees), longitude (decimal degrees), and height above MSL (km).

Parameters:

{tHDG} (out/series) – true heading (deg)

{mHDG} (in/series) – magnetic heading (deg)

{lat} (in/series) – latitude (decimal degrees)

{lon} (in/series) – longitude (decimal degrees)

{ht} (in/series) – height above MSL (km)

{year} (in/scalar) – decimal year (eg. 1992.14 for February 20, 1992). Must be in the range 1990-1995

{CoSys} (keyword) – geocentric or geodetic

{Accuracy} (keyword) – high or low

Note: if "high", magnetic variation is computed for every element (this is SLOW!)

if "low", magnetic variation is computed from the average of the first and last elements (accurate enough for most purposes).

7.95 ThreshCross

Syntax: ThreshCross {out} {in} {value}

Description: Generate a series of the element indices where {in} crosses from below {value} to above.

Parameters:

{out} (out/series) – index series
 {in} (in/series) – series to analyze
 {value} (in/scalar) – threshold value

7.96 SymDist

Syntax: SymDist {dist} {x} {y}

Description: Compute distance along "mean" track with origin at first point. "Mean" track = linear regression line through the track coordinates data.

7.97 Dist(Orig)

Syntax: Dist(Orig) {dist} {x} {y} {x0} {y0}

Description: as 'SymDist', but origin specified.

7.98 Dist(Orig,Trk)

Syntax: Dist(Orig,Trk) {dist} {x} {y} {x0} {y0} {trkangle}

Description: as 'SymDist', but origin and track angle specified and used to define the line for projection (instead of a regression line, as in SymDist)

Parameters:

{dist} (out/series) – distance along mean track(positive in direction of track angle)
 {x} (in/series) – original x-coordinates of track
 {y} (in/series) – original y-coordinates of track
 {x0} (in/scalar) – new origin (uses the projection of this point onto the regression line)
 {y0} (in/scalar) – new origin (uses the projection of this point onto the regression line)
 {trkangle} (in/scalar) – track angle (from North)

Note: "positive" direction is along regression line from (projection of) origin to (projection of) last point.

7.99 Truncate

Syntax: Truncate {out} {in} {left} {right}

Description: Convert time in decimal days from beginning of reference year into date and time in 6-digit format

Parameters:

{out} (out/series) – output series
{in} (in/series) – input series
{left} (in/scalar) – number of elements to cut of at beginning of series
{right} (in/scalar) – number of elements to cut of at end of series

7.100 Ts(Tt,sp,dp)

Syntax: Ts(Tt,sp,dp) {Ts} {Tt} {sp} {dp}

Description: Static temperature from true temperature, static pressure, and dynamic pressure assuming dry atmosphere.

Parameters:

{Ts} (out/series) – static temperature
{Tt} (in/series) – true temperature
{sp} (in/series) – static pressure
{dp} (in/series) – dynamic pressure

7.101 t,tas(t,qc,p,td,r)

Syntax: t,tas(t,qc,p,td,r) {t} {tas} {traw} {qc} {p} {td} {r}

Description: Accurate TAS and speed-corrected Temperature from static/dynamic pressures, uncorrected temperature series and absolute humidity

Parameters:

{t} (out/series) – speed-corrected temperature [*degC*]
{tas} (out/series) – true air speed [*m/s*]
{qc} (in/series) – dynamic pressure [*hPa*]
{p} (in/series) – static pressure [*hPa*]
{traw} (in/series) – uncorrected temperature [*degC*]
{td} (in/series) – dewpoint temperature [*deg C*]
{r} (in/scalar) – recovery factor for temperature probe

7.102 Tt(Ttm,sp,dp,r)

Syntax: Tt(Ttm,sp,dp,r) {Tt} {Ttm} {sp} {dp} {r}

Description: True temperature from measured temperature, static pressure, dynamic pressure, and recovery factor - assuming dry atmosphere

Parameters:

{Tt} (out/series) – true temperature
{Ttm} (in/series) – measured temperature
{sp} (in/series) – static pressure
{dp} (in/series) – dynamic pressure
{r} (in/scalar) – recovery factor

7.103 uvw(ECEFuvw)

Syntax: `uvw(ECEFuvw) {u} {v} {w} {ue} {ve} {we} {lat} {lon}`

Description: (u,v,w) in local coordinates from (u,v,w) in GPS ECEF coordinates and geographic location (lat/lon in degrees)

Parameters:

`{u}` `{v}` `{w}` (out/series) – (u,v,w) in local coordinates [*m/s*]
`{ue}` `{ve}` `{we}` (in/series) – (u,v,w) in ECEF coordinates [*m/s*]
`{lat}` `{lon}` (in/series) – geographic coordinates [*deg*]

7.104 WarnInvalid

Syntax: `WarnInvalid {series1} {series2}...{seriesn}`

Description: Count the occurrences of invalid values in one or more series; if any invalid values are found, a warning is issued.

Parameters:

`{seriesi}` (in/series) – the series to check

7.105 XYZ2LLA

Syntax: `XYZ2LLA {lat} {lon} {alt} {x} {y} {z}`

Description: Transform coordinates from GPS ECEF (Earth Centered Earth Fixed) into WGS84 latitude, longitude and altitude above geoid.

Parameters:

`{lat}` (out/series) – latitude [*deg*]
`{lon}` (out/series) – longitude [*deg*]
`{alt}` (out/series) – altitude above WGS84 geoid [*m*]
`{x}` (in/series) – x coordinate [*m*]
`{y}` (in/series) – y coordinate [*m*]
`{z}` (in/series) – z coordinate [*m*]

7.106 zm(p,Tv)

Syntax: `zm(p,Tv) {zm} {p} {Tv} {QFE} {z0}`

Description: mean altitude (m) from pressure (hPa) using the isothermal altimetric equation directly (no integration). Assumes constant temperature in the atmosphere, so the temperature series (deg C) is used only to calculate the mean)

Parameters:

`{zm}` (out/series) – altitude [*m*]
`{p}` (in/series) – pressure [*hPa*]
`{Tv}` (in/series) – virtual temperature [*C*]
`{QFE}` (in/scalar) – static pressure [*hPa*] at reference level
`{z0}` (in/scalar) – height [*m*] of reference level above MSL

7.107 $z(p',Tv')$

Syntax: $z(p',Tv')$ {z} {p} {Tv} {QFE} {T0} {z0}

Description: altitude (m) from pressure (hPa) and virtual temperature (deg C) for level flights; numerically integrates the hydrostatic equation; the mean altitude is computed first from mean pressure, mean Tv and ground data, then only deviations from these means are used.

Parameters:

- {z} (out/series) – altitude [*m*]
- {p} (in/series) – pressure [*hPa*]
- {Tv} (in/series) – virtual temperature [*C*]
- {QFE} (in/scalar) – static pressure [*hPa*] at reference level
- {T0} (in/scalar) – air temperature [*C*] at reference level
- {z0} (in/scalar) – height [*m*] of reference level above MSL

7.108 $z(p,Tv)$

Syntax: $z(p,Tv)$ {z} {p} {Tv} {QFE} {T0} {z0}

Description: altitude (m) from pressure (hPa) and virtual temperature (deg C) by numerically integrating the hydrostatic equation upwards; only useful for ascending flight; otherwise use $zm(p,Tv)$ or $z(p',Tv')$

Parameters:

- {z} (out/series) – altitude [*m*]
- {p} (in/series) – pressure [*hPa*]
- {Tv} (in/series) – virtual temperature [*C*]
- {QFE} (in/scalar) – static pressure [*hPa*] at reference level
- {T0} (in/scalar) – air temperature [*C*] at reference level
- {z0} (in/scalar) – height [*m*] of reference level above MSL

8 Grid Manipulation Commands

8.1 2Daxes

Syntax: 2Daxes {xaxis} {yaxis} {xgrid} {ygrid} {rows}

Description: Create x/y-axes of a grid from x/y-coordinates

Parameters:

{xaxis} (out/series) – x-axis values

{yaxis} (out/series) – y-axis values

{xgrid} (in/grid) – grid of x coordinates

{ygrid} (in/grid) – grid of y coordinates

{rows} (in/scalar) – number of rows of grid

8.2 2Dblank

Syntax: 2Dblank {zout} {x} {y} {zin} {xbound} {ybound}

Description: Blank area of a grid, i.e. set grid point values to 'invalid'

Parameters:

{zout} (out/grid) – grid points with blanking applied

{xgrid} (in/grid) – grid of x coordinates

{ygrid} (in/grid) – grid of y coordinates

{zin} (in/grid) – grid of z values

{xbound} (in/series) – x coordinate list of boundary

{ybound} (in/series) – y coordinate list of boundary

8.3 2Dcoord

Syntax: 2Daxes {x-grid} {y-grid} {x-axis} {y-axis}

Description: Create x/y-coordinates for each point of a grid from the x/y-axes

Parameters:

{xgrid} (out/grid) – grid of x coordinates

{ygrid} (out/grid) – grid of y coordinates

{xaxis} (in/series) – x-axis values

{yaxis} (in/series) – y-axis values

8.4 2DdiffX

Syntax: 2DdiffX {diff} {grid} {rows} {delta}

8.5 2DdiffY

Syntax: 2DdiffX {diff} {grid} {rows} {delta}

Description: Differentiate a grid along in x or y direction

Parameters:

{diff} (out/grid) – differentiated grid

{grid} (in/grid) – input grid
{rows} (in/scalar) – number of rows of grid
{delta} (in/scalar) – data interval

8.6 2DflipTB

Syntax: 2DflipTB {out} {in} {columns}

Description: Reverse 2D-grid top/bottom

Parameters:

{out} (out/grid) – output grid
{in} (in/grid) – input grid
{columns} (in/scalar) – number of columns

8.7 2Drotate

Syntax: 2Drotate {out} {in} {ncols} {angle}

Description: Create x/y-axes of a grid from x/y-coordinates

Parameters:

{out} (out/grid) – output grid
{in} (in/grid) – input grid
{ncols} (in/scalar) – number of columns
{angle} (in/scalar) – rotation angle (-270,-180,-90,0,90,180,270)

8.8 2DSelect

Syntax: 2DSelect {xn} {yn} {zn} {x} {y} {z} {xs} {xe} {xq} {ys} {ye} {yq} {rows}

Description: Select values from a grid.

Parameters:

{xn} (out/grid) – output x grid
{yn} (out/grid) – output y grid
{zn} (out/grid) – output z grid
{x} (in/grid) – input x grid
{y} (in/grid) – input y grid
{z} (in/grid) – input z grid
{xs} (in/scalar) – x start of selection
{xe} (in/scalar) – x end of selection
{xq} (in/scalar) – x step of selection
{ys} (in/scalar) – y start of selection
{ye} (in/scalar) – y end of selection
{yq} (in/scalar) – y step of selection
{rows} (in/scalar) – number of rows of input grid

8.9 2DSmooth

Syntax: 2DSmooth {zout} {zin} {nSmooth} {nColumns}

Description: Laplacian smoothing of a grid.

Parameters:

{zout} (out/grid) – smoothed grid

{zin} (in/grid) – input grid

{nSmooths} (in/scalar) – number of desired smoothings

{nColumns} (in/scalar) – number of columns of the grid

8.10 InterpolateColumn

Syntax: InterpolateColumn {yax} {cout} {yin} {cin} {ncols} {nlevels} {ymin} {ymax}

Description: Interpolate grid to new range and number of data points per column

Parameters:

{yax} (out/series) – y axis of interpolated grid

{cout} (out/grid) – output grid

{yin} (in/grid) – original y grid

{cin} (in/grid) – original data grid

{ncols} (in/scalar) – number of columns in input grid

{nlevels} (in/scalar) – number of new data points per column (default = 10)

{ymin} (in/scalar) – minimum of new y axis (default = minimum of data grid)

{ymax} (in/scalar) – maximum of new y axis (default = maximum of data grid)

8.11 InterpolatePath

Syntax: InterpolatePath {out} {xpath} {ypath} {xaxis} {yaxis} {grid}

Description: Interpolate gridded data onto a sampling path using 2D quadratic interpolation

Parameters:

{out} (out/series) – interpolated values at path locations

{xpath} (in/series) – x coordinates of sampling path

{ypath} (in/series) – y coordinates of sampling path

{xaxis} (in/series) – x axis of input grid (m elements)

{yaxis} (in/series) – y axis of input grid (n elements)

{grid} (in/grid) – data grid (n * m elements)

8.12 ThinGrid

Syntax: ThinGrid {out} {in} {ncols}

Description: Thin out a grid by a factor of two using linear interpolation, i.e. if the input grid is of the dimension (16,16), ThinGrid returns a grid of the

dimension (8,8). If any dimension reaches 1, it is not further reduced (i.e. an input of (1,20) results in an output of (1,10).

Parameters:

- `{out}` (out/grid) – output grid
- `{in}` (in/grid) – input grid
- `{ncols}` (in/scalar) – number of columns in input grid

9 Examples

References

- [1] H. KRAUS, *Die Atmosphäre der Erde*, Vieweg Verlag, 2000.
- [2] B. KÜMMEL, *Temp, Humidity and DewPoint ONA*.
<http://mf.ruc.dk/bek/relhum.htm>, 1997.
- [3] P. SCHWERDTFEGER, *Introduction to Thermodynamics of the Atmosphere*. Lecture Notes, 1995.
- [4] R. STULL, *An Introduction to Boundary Layer Meteorology*, Kluwer Academic Publishers, 1988.

10 Appendices

10.1 Fonts

10.1.1 Recommended Text Fonts

It is recommended that only 'standard' PostScript fonts should be used when generating **RAMF** output, e.g. those included in the GhostScript package:

Helvetica
Helvetica-Bold
Helvetica-Oblique
Helvetica-BoldOblique
Helvetica-Narrow
Helvetica-Narrow-Bold
Helvetica-Narrow-Oblique
Helvetica-Narrow-BoldOblique
Times-Roman
Times-Bold
Times-Italic
Times-BoldItalic
Courier
Courier-Bold
Courier-Oblique
Courier-BoldOblique
Symbol
AvantGarde-Book
AvantGarde-BookOblique
AvantGarde-Demi
AvantGarde-DemiOblique
Bookman-Demi
Bookman-DemiItalic
Bookman-Light
Bookman-LightItalic
NewCenturySchlbk-Roman
NewCenturySchlbk-Italic
NewCenturySchlbk-Bold
NewCenturySchlbk-BoldItalic
Palatino-Roman
Palatino-Italic
Palatino-Bold
Palatino-BoldItalic
ZapfChancery-MediumItalic
ZapfDingbats

Table 1: Recommended PostScript fonts

10.1.2 RAMFsymbols Font

A symbol font containing 252 centered (i.e. the centre of each character is co-located with the centre of the font cell) symbols is part of the **RAMF** package. It can be found in the directory `R12/lib/fonts` in Postscript Type 1 ASCII (*RAMFsymbols.pfa*) and TrueType (*RAMFsymbols.ttf*) formats. While the main use of this font is for the generation of symbols in plots, it can also be used in PostScript and LaTeX table output, as well as in other applications away from **RAMF** (e.g. in a word-processing package when writing articles incorporating plots generated by the **RAMF** plot system).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	○	◐	◑	◒	◓	◔	◕	◖	◗	◘	◙	◚	◛	◜	◝	◞
0010	●	◐	◑	◒	◓	◔	◕	◖	◗	◘	◙	◚	◛	◜	◝	◞
0020	↑	↓	↗	↘	↙	↘	↗	↖	↗	↘	↙	↘	↗	↖	↗	↘
0030	①	②	③	④	⑤	⑥	⑦	⑧	⑨	☂	☃	☄	★	♦	♥	♣
0040	①	②	③	④	⑤	⑥	⑦	⑧	⑨	☂	☃	☄	★	♠	♣	♣
0050	○	+	△	△	□	□	□	□	□	□	□	□	□	□	□	□
0060	○	+	△	△	□	□	□	□	□	□	□	□	□	□	□	□
0070	●	◐	◑	◒	◓	◔	◕	◖	◗	◘	◙	◚	◛	◜	◝	◞
0080	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘
0090	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙
00A0	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙
00B0	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙
00C0	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙
00D0	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙
00E0	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙
00F0	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙	⊕	⊗	⊘	⊙

Table 2: Symbol table for the font “RAMFsymbols”

The characters of this font can either be accessed directly by their ASCII equivalent (e.g.. the character '1' generates an encircled number one when printed in the RAMFsymbols font) or by their numerical encoding.

As most of the characters are outside the normal ASCII character range, the most convenient way of accessing them in **RAMF** is via hex code to ASCII substitution using format expansion.

Example(s):

In order to generate the kangaroo symbol, one first has to look up its hex code in table 2. To use this hex code as part of a **RAMF** string, one then has to insert the format expansion code `~\#7E:A~` at the desired location.

10.1.3 Predefined Colours

7 - white	15 - black
6 - maroon	14 - yellow
5 - purple	13 - fuchsia
4 - red	12 - olive
3 - teal	11 - blue
2 - green	10 - lime
1 - navy	9 - aqua
0 - grey	8 - lightgrey

Table 3: Default definitions of colour numbers 0 - 15

10.1.4 Predefined Line Types

16 - filled
4 - dashdotdot
3 - dashdot
2 - dotted
1 - dashed
0 - solid

Table 4: Default line type descriptors

11 GNU Free Documentation License

Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other written document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

11.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the

Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, L^AT_EX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

11.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

11.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual

cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

11.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled “History”, and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled “Acknowledgements” or “Dedications”, preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled “Endorsements”. Such a section may not be included in the Modified Version.
- Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

11.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled “History” in the various original documents, forming one section entitled “History”; likewise combine any sections entitled “Acknowledgements”, and any sections entitled “Dedications”. You must delete all sections entitled “Endorsements.”

11.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

11.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

11.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

11.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

11.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.