# Gcal

# Gcal

An Extended Calendar Program
Edition 3.01
June 2000

**by Thomas Esken**

Any suggestions, improvements, extensions, bug reports, donations, proposals for contract work, and so forth are welcome! Please send them directly to my eMail address esken@gmx.net. If you like my work, I'd appreciate a postcard from you!

```
  ------------------------oOO \\\_''/ OOo--------------------------
            Thomas Esken O (/o-o\) O eMail:  esken@gmx.net
            Im Hagenfeld 84 (( ^ )) Phone:  +49 251 232585
D-48147 Muenster; Germany \____) ~ (____/ MotD : 2old2live, 2young2die
```

This is Edition 3.01 of *Gcal, an Extended Calendar Program*,
for the 3.01 (or later) version of the GNU implementation
of `cal` and `calendar`.

# Short Contents

# Table of Contents

# 1 Preface

The reason why I have written Gcal was mainly to have an *off-line* tool[1] which is able to display Julian and Gregorian calendar sheets in a highly flexible and internationalized manner, fixed date lists for reminding purposes and country respectively territory specific holiday lists. At a later stage, Gcal was extended in the way to perform some basic astronomical calculations like Moon phase, moonrise, moonset and other Moon based data, sunset, sunrise, different twilight times and other Sun based data, zodiacal markers, solar and lunar eclipses, equinoxes and solstices. Other gimmicks like biorhythm calculation and the computation of a distance between two geographical point locations and their course angles (true tracks) are also done by it now, plus a lot more.

The accuracy of most of the astronomical calculations done by Gcal is low because it is really hard to perform accurate astronomical predictions more that 100 years ahead. Of course, this also has consequences for some country specific holidays which are based on such astronomical calculations. So it is really possible that some output may differ from the real astronomical date and time of the event calculated, and that by one or more minutes/hours/days. This can happen when the date of a real astronomical event occurs near around midnight. So don't be surprised! Strictly speaking, take all such astronomically based output as a more or less good approximation of the true value, but do not rely on it!

Because Gcal is not primarily designed for the purpose of having a high precision astronomical calculation tool, the world currently has to live with these limitations. But if **you** have C source code functions, which

- perform the astronomical calculations more precise for the period that is respected by Gcal,
- have the similar or faster processing speed than Gcal's actual functions,
- are either put into the public domain or are covered by the GPL[2],
- have the same calling conventions and return types like Gcal's actual functions,
- conform to the GNU coding standard and are therefore easy to maintain and extend,
- do not break Gcal internal limitations and concepts,

and you want to grant them to the public, I would be very pleased if you would contact me so I am able to implement your result to a next release of this software.

Some words to the build-in country specific holidays now...

As you certainly know, there are several other calendar systems actually existing around the world among the Gregorian calendar. The Gregorian calendar that Gcal mainly represents as a software is only one calendar of many others, but it is that one which is most respected around the world today. Even countries which use another calendar system officially mostly respect the Gregorian calendar for civil and administrative purposes.

But when Gcal offers the ability to include holidays from a definite country into its eternal holiday list, it is possible that not all holidays are displayed which are respected in that country. One reason is that my various sources simply don't tell me about the existence of a holiday. A second reason is that the holiday is just designed or changed or removed by political or society reality during Gcal's development phase. And another reason is that the method to calculate a holiday is not build-in Gcal yet. This case is marked by a '`#`' character behind the country name in the list of countries, which are respected by Gcal. See [Calendar option '`--cc-holidays=`*cc*[`_`*tt*][`+`...]'], page 16, for the above mentioned list.

[1] I.e. no need to search for calendar information in the net so phone and Internet Service Provider (ISP) costs are reduced, and furthermore, neither searching for such information in books or data files nor the need to use several other software which altogether do what Gcal actually also does.

[2] The "General Public License" of the Free Software Foundation.

The holidays of *early* years that are included into the eternal holiday list —i.e. those within the period AD 30 until the midth of the 20th century— are certainly not accomplishing the requirements of historians to such a list. Gcal cannot be understood as a complete and historically correct data base, and it is not designed for that purpose! This is caused by the fact that the territorial borders and names of the national structures and the settlement in earlier times was partly quite different as it is nowadays. Moreover, I am yet lacking reliable information that tells me the precise date of the introduction of a holiday in a region. Here I appeal to the historians to support me with the concrete country and culture based data if they are bothered by this fact and they want to have it removed.

Still unsupported **alive** and **major** calendar systems, respectively methods to derive country specific holidays from them, are:

- Religious Islamic lunar calendar
- Hindu solar and lunar calendars
- Buddhist calendar
- Sikh calendar
- Jain calendar

Even if the list of supported countries seems to be almost complete if you compare it with the actual list of all nations existing around the world, you should note that there are still some countries missing! If you can grant me support to complete Gcal's list of respected countries, or if you would work out or have C functions which manage the missing calendar systems, I also would be very pleased about if you would contact me.

I have recorded the country specific holiday dates as good as I can, but I cannot guarantee that I did not made any mistakes while I hacked them in. So it is possible that some country specific holiday lists contain errors.

Well, if you think that I forgot to provide Gcal with the specific holidays of a/your country, or you think that a country specific holiday list is incomplete, or you think that some of the already implemented holidays are wrong, please feel free to send me a complete list of all the holidays which are celebrated in this country respectively to give me hints how to correct or to complete them!

Such a list should ideally include the following information:

- The English name of the holiday.
- The rule how it is built (like 10th October each year, or 47 days after Easter Sunday's date, or three Fridays before the last Saturday in June. . . ). If there are any exceptions from such a general rule, please state them in a detailed manner. And please tell me the date since what time the holiday is or was celebrated for the first time, respectively.

  By the way, if the holiday is tied to the Easter Sunday's date, please tell me whether you mean the Easter Sunday of the Christian Western churches or the Easter Sunday of the Orthodox Christian Eastern churches.
- Whether it is a *legal holiday* which is respected in the whole country. (Use a '`+`' character for marking this case.) In this sense, a legal holiday means a day people don't have to work — as in most companies, and businesses are closed; like '`New Year's Day`' on January 1st.
- Whether it is a legal holiday in some parts of your country only. Please tell me whether such a holiday is valid in major parts of the country or not, try to express that by using percent points. (If such a holiday is celebrated in more than 70% of the country, use a '`#`' character for marking this case, otherwise use a '`*`' character.)
- Whether it is a day only used for memorial or remembering purposes. (Use a '`-`' character for marking this case.) Like '`All Fool's Day`' on April 1st is not a holiday in the sense that anyone has a day off from school or work, just a *play jokes* day, pretty ordinary. Other

holidays may be celebrated with cards or presents etc. — but they are working days such as:

```
Valentine's day, Father's day, Mother's day.
```

The Islamic holidays inserted into the country specific holiday list are calculated arithmetically, not astronomically. This is what people call the *civil* Islamic lunar calendar. This type of calendar is just a rough estimate of the religious Islamic lunar calendar, which is determined quite astronomically. So don't be surprised too much if some holiday dates differ from the religious Islamic calendar about ± 1...2 days in some years.
The main difficulty to implement the religious Islamic calendar correctly into Gcal are the several different criteria that are existing for the first sighting of the crescent Moon after a New Moon phase — that event, which defines the beginning of a new (lunar) month in the religious oriented Islamic calendar. So on the one hand, there are quite astronomically-based criteria, and on the other hand, there are also quite religiously-based criteria. Furthermore, the local predominant weather conditions, the geographical co-ordinate of the observer's location and the kind of the observance method used also affect these criteria. Moreover, to render all more difficult, I as yet do not have any reliable information that tells me which kind of criteria is used in the countries (around 60 world-wide) which respect this religious Islamic calendar. So due to all the above mentioned facts of problematic nature, this religious Islamic calendar is not included into Gcal yet.

The Chinese (and Japanese) holidays also inserted into the country specific holiday list are calculated by using that algorithm as it is used by the Purple Mountain Observatory in Nanjing to determine the astronomical lunisolar Chinese calendar. This calendar is obligatory for (the Peoples Republic of) China since 1929 and is determined annually always new. Nevertheless and strictly speaking, in the historic reality and depending on the region, the Chinese calendar, especially in the 17th and 18th century, was either calculated using this *modern* algorithm, or by the use of a different method, so that a different calendar structure for one and the same year was possibly calculated as result.

And last...

This manual shows the possible use of the Gcal program only briefly and deliberately renounces to present the topics that are covered by this software arranged in a novel-like style. The use of Gcal is illustrated in an exemplary manner only! A detailed description of all possible use is inexpedient here because there are simply too many of such possibilities which are caused by the existing variety of combinations of all useable modes of operation.

Furthermore, this manual does not contain detailed explanations about the calendar systems supported by Gcal, nor it can be understood as a reference-book for the basics in celestial mechanics or spherical trigonometry! A detailed discussion of the above topics is completely beyond the scope of this manual. If you are interested to learn more about these subjects, you unfortunately have to read the specific pertinent literature.

Any further errors occurring in the output of the above mentioned holiday lists and in the astronomical calculations are my own fault, and are not intended to offend members of any culture, religion or profession!

# 2 Gcal Introduction

Apart from the usual and well known calendar functions like the output of a month or a year calendar sheet, or the output of an eternal holiday list, Gcal, the Gregorian calendar program of the Free Software Foundation, offers the facility to display fixed dates on the day of their occurrence and to remind or inform the user about them. So it is imaginable after booting the computer or starting the work session, that the user is informed on screen or by means of electronic mail about all holidays or appointments which are observed or scheduled for that day.

The period, for which Gcal respects occurring fixed dates, may be freely selected by the user. So it is possible that Gcal displays all fixed dates which occur on tomorrow's date, the whole week, the whole month or in the whole year. Fixed dates which occur on a selected date of the year, and those that occur relative to another given date, are displayed either related to this single date only, or in listed manner starting on this date and ending on the actual date[1].

There are two methods to display a preview of fixed dates[2] or retrospective view of fixed dates[3]. On the one hand, Gcal can be started by using an option that sets the system date of the computer to the given date during the time of the program execution with the result, the program assumes the system date is set to this given date and the user can define any needed period that should be respected by an option. On the other hand, Gcal can be started with a command which forces the program to use a different year instead of the actual year, so Gcal will display all occurring fixed dates for this particular year. But this limits the user in that it disables defining any needed period by an option, because the period is always set to the whole year by default.

Gcal isn't only able to display fixed dates which are stored for a concrete date, e.g. '`Fixed date on 1st December 1995`', rather than fixed dates occurring periodically again and again. So it is possible to define repeated events like '`This fixed date occurs every day in May 1995`' or '`Every 15th November in any years`'. These fixed date definitions are stored in resource files and whenever Gcal is started, an option to evaluate the necessary resource files can be given.

Once the user has set his/her preferred command line arguments for querying the fixed dates data base, it is possible to store them in a response file or shell script file. A response file contains all arguments delivered to Gcal, but unlike a shell script file, such a response file isn't executable; it is only a pool of command line arguments which can be preloaded if needed. A shell script file can be started and calls Gcal directly with all arguments stored in it and all arguments which are given further in the command line.

A list of all usable command line arguments and their descriptions can be found in the next chapter, which helps one to use Gcal in the most efficient and productive way possible. After it follows the description how to use the eternal holiday list, and in the succeeding chapter, how to use the fixed date list. Comprehensive explanations respectively summaries and tables to definite details or themes from other disciplines can be found in the numerous appendices.

[1] Inclusive or exclusive.

[2] Future years.

[3] Past years.

# 3 Invoking gcal

Gcal is a command line oriented program. It is usually called from the *shell*[1] and processes given arguments that are options and commands. Options must be given before commands, i.e. you must call Gcal like this:

```
gcal [ [option...] [%date] [@file...] ] [command]
```

If Gcal is started without any options or commands, a calendar of the current month is displayed. If the calendar of a definite year is wanted, the year must be fully specified, e.g. ‘gcal 94’ displays a year calendar of the year 94, not of the year 1994.

If two arguments are given in the command part, the **first** argument denotes the month, and the **second** argument denotes the year. In case any incorrect commands are given running Gcal, the program will use internal defaults.

In the English program version, Gcal assumes the Gregorian Reformation has occurred in 1752 on the 3rd of September. See Appendix A [Genesis of the Gregorian Calendar], page 117, and Appendix C [Aspects in Internationalization], page 123, for further details.

[1] This is an operating system program which interprets and runs given command lines.

## 3.1 Command line arguments

This section describes all command line arguments processed by Gcal. Four different types of command line arguments exists. One important type of arguments are the *options* which control how Gcal behaves. Other types of arguments are the ‘%*date*’ and the ‘@*file*’ options. The ‘%*date*’ option sets the period Gcal shall work on to any starting date; the ‘@*file*’ option preloads options and commands from a response file. The most important arguments are the *commands* which control the periods Gcal respects.

An option is defined by a leading *switch* character; either the ‘-’ (dash) or the ‘/’ (slash) character for traditional short-style options, or ‘--’ for mnemonic long-style options; a command may not have a leading switch character. Options[1] **must** be given before commands!

Depending on operating system and used shell, some of the arguments and texts given in command line must be quoted by ‘"’ or ‘'’ characters respectively protected or depreciated by a ‘\’ character to avoid expansion by the shell.

Here is an incomplete list of characters which must potentially be protected:

‘(’, ‘)’, ‘<’, ‘>’, ‘[’, ‘]’, ‘{’, ‘}’, ‘\’, ‘|’, ‘$’, ‘@’, ‘!’, ‘&’, ‘~’, ‘"’, ‘'’, ‘`’, ‘;’

[1] Inclusive ‘%*date*’ and ‘@*file*’.

### 3.1.1 Options

The options processed by Gcal can be grouped into four major option classes. The options of the *common option class* are the standard options all GNU software should implement at least partially. The *global option class* contains options which affect the program output. The options of the *calendar option class* control the calendar layout, and the options of the *fixed date option class* control the fixed date layout and intensity.

Gcal supports both short-style options and GNU long-style options. Traditional short-style options are indicated by a single switch character, and trailed by the option character itself and perhaps a modifier or an argument. The most single character options[1] can be composed into a single command line word: '`-Ax`' is equivalent to '`-A -x`'. GNU long-style options are indicated with '`--`', and trailed by the mnemonic option name itself and perhaps an argument. Long-style options and their arguments may be abbreviated if done unambiguously. When a long-style option takes an argument, connect the option name and the argument with '`=`'.

Brackets ([ and ]) indicate in the following tables, that an option takes an optional argument. The '`|`' character is used to separate several arguments from each other.

Gcal processes the GNU long-style options in a special, non-standard way. There are five different types of long-style options:

1. '`--foo`'
   Enables option '`--foo`'.
2. '`--foo=`*bar*'
   Enables option '`--foo`' with the required argument *bar*.
3. '`--foo`[`=`*bar*[`|`...`|`*bar*]]'
   Option '`--foo`' may have one *bar* argument. If no argument list is given, any argument can be given to this option. If an argument list is given, exactly one *bar* argument may be selected from the given list. If there is no argument chosen in this case, the first *bar* argument of the argument list is preselected by default.
4. '`--foo=`*bar*`|`...`|`*bar*'
   Option '`--foo`' requires exactly one *bar* argument which must be selected from the given argument list.
5. '`--foo=`*bar*`|`...`|`*bar*`|`*baz*'
   Option '`--foo`' requires exactly one *bar* argument which must be selected from the given *bar* argument list, or the alternative *baz* argument.

Traditional short-style options differ as follows:

1. '`-x`'
   Enables option '`-x`'.
2. '`-x` *bar*'
   Enables option '`-x`' with the required argument *bar*. The *bar* argument may be separated by a leading *whitespace* character from the short-style option character '`x`'. This means, the following notations are valid for giving an argument, namely '`-x` *bar*' or '`-x`*bar*'.
3. '`-x`[*bar*`|`...`|`*bar*]'
   Option '`-x`' may have one or more *bar modifier*. In this sense, modifiers are one or more characters which define a special mode of operation enabled by the '`-x`' option. A modifier may not be separated by a leading *whitespace* character from the short-style option character.

[1] Unless they take an argument.

### 3.1.1.1 Common options

'`-?`'  
'`-h`'  
'`--help`' Print a short usage message listing only some few of all available options, then exit successfully.

'`-??`'  
'`-hh`'  
'`--usage`[`=`*argument*]'  
'`--long-help`[`=`*argument*]'

Print an extended usage message listing all available options, then exit successfully. If an *argument* is given and it is a valid long-style option name, an extended help text related to the given long-style option name is displayed only, e.g.:

```
--long-help=long-help
```

displays the extended help text for the long option '`--long-help`'.

If the *argument* only consists of the single '`?`' character or is no valid long-style option name, a list of all valid long-style option names is displayed.

'`-L`'  
'`--license`'  
'`--copyleft`'  
'`--copyright`'

Print the software license message, then exit successfully.

'`-V`'  
'`--version`'

Print the version number and compilation options, then exit successfully.

'`--exit-status-help-non-zero`'

Set the exit state of program to 127 instead to 0, if one of the other options of the *common option class* is used.

### 3.1.1.2 Global options

'`-R` *name*'  
'`--response-file=`*name*'

Write the contents of the environment variable `GCAL` (see [Environment Variable `GCAL`], page 237), and then the arguments of command line (in the given order) to file *name*, i.e. create response file '`name`'. See Section 3.1.2 [Response file], page 60, for more details.

'`-S` *name*'  
'`--shell-script=`*name*'

Write the contents of the environment variable `GCAL` (see [Environment Variable `GCAL`], page 237), and then the arguments of command line (in the given order) to file *name*, i.e. create shell script file '`name`'. An automatically created shell script file is executable and calls Gcal directly with the arguments stored in it. You may start the shell script with other command line arguments which are directed to Gcal, too.

'`--debug`[`=internal|handled|unhandled|all|abort`]'

Display some debug information.

'`--debug=internal`'

Display informational texts if program internal maxima are reached or other conditions occurred, respectively.

'--debug=handled'
: Like '--debug=internal', additionally display the file names which can be processed respectively handled.

'--debug=unhandled'
: Like '--debug=internal', additionally display file names which cannot be processed respectively handled.

'--debug=all'
: Like '--debug=handled' and '--debug=unhandled' together.

'--debug=abort'
: Like '--debug=all' and abort program with an error code if the file name cannot be handled or other unmanageable conditions occurred, respectively. See Appendix M [Error Codes], page 239.

'-p'
'--pager'
: Enables either an *external* pager or a simple, *internal* pager. If an environment variable PAGER is set, its contents will be used for detecting the external pager program. See [Environment Variable PAGER], page 238, for more information.

If no PAGER environment variable is set or if its contents is invalid, Gcal tries to use the less pager; if this program cannot be found during scanning the PATH environment variable, Gcal tries to use the more pager, if this program cannot be found, the pg pager in the same way[2]. See [Environment Variable PATH], page 238.

If all these actions fail, Gcal will use its simple, built-in pager. If the internal pager is used, Gcal detects the number of lines shown before it prompts and waits for user input by using these methods:

1. Gcal respects the values set in the environment variables GCAL_LINES and GCAL_COLUMNS. See [Environment Variable GCAL_LINES], page 237, and [Environment Variable GCAL_COLUMNS], page 237, for further information.
2. If above action fails, Gcal respects the values set in the environment variables LINES and COLUMNS. See [Environment Variable LINES], page 237, and [Environment Variable COLUMNS], page 237, for more details.
3. If above action fails, Gcal respects the values set in the 'termcap'[3] file which refers to the terminal used (see [Environment Variable TERM], page 238). This step is only done on systems which support the use of Termcap by default.

   On MS-DOS, OS/2 and some other operating systems, Gcal uses a system dependent *low-level* function and respects the reported values.
4. If all above actions have failed, Gcal uses default values[4].

'-H yes'
'--force-highlighting'
'--highlighting=yes'
: If the output of the program is redirected[5] or piped[6], the highlighting sequences are **not** automatically converted into the according marking characters, they remain unchanged. This option takes no effect if the output of the program is sent by means of electronic mail. See [Global option '--mail[=*address*]'], page 14.

[2] See the standard manual pages for less, more and pg.

[3] See the standard manual pages for Termcap.

[4] Either 23 or 24 lines, and 80 columns.

[5] This means, sent to another device.

[6] This means, used as an input data stream for another program.

‘`-H no`’<br>
‘`--disable-highlighting`’<br>
‘`--highlighting=no`’

> Disable highlighting sequence / marking character pairs of current day, holiday or text explicitly.

‘`-H` *`text`*’<br>
‘`--highlighting=`*`text`*’

> Set highlighting sequence / marking character pairs explicitly. In this sense, *highlighting* sequences are control character sequences which cause a color or intensity switch in output text. Typical control character sequences are the ANSI escape sequences which have a leading escape character, and trailing more characters that define the type of the ANSI escape sequence. In this sense, *marking* characters are single, printable characters which lead and trail the output text.
>
> The *text* argument must be a (‘`:`’) colon-separated text which is structured in this way: *seq1_start*:*seq1_end*:*seq2_start*:*seq2_end*. The **first** sequence is used for highlighting/marking an actual day, the **second** for a holiday. The sequences must be given in form of a sequence pair; *seq?_start* enables the highlighting/marking, *seq?_end* disables it. Only two sequence pairs will be processed, others are ignored. Either highlighting sequence pairs or marking character pairs may be defined, i.e. using them both in a mixed couple is not permitted!
>
> For example:
>
> > ‘`-H \x20:\x20:\x1:#`’ respectively<br>
> > ‘`--highlighting=\x20:\x20:\x1:#`’<br>
> > marks the actual day like ‘`\x20`*`actual date`*`\x20`’[7] and the holiday date like ‘`\x1`*`holiday date`*`#`’ using the given marking characters.
> >
> > ‘`-H \x1b[34;42m:\x1b[0;40m`’ or<br>
> > ‘`-H \033[34;42m:\033[0;40m`’ or<br>
> > ‘`-H \E[34;42m:\E[0;40m`’<br>
> > defines a starting ANSI escape highlighting sequence ‘`\x1b[34;42m`’ used for actual day and ending ANSI escape highlighting sequence ‘`\x1b[0;40m`’ with no given highlighting sequence for holidays, so default highlighting sequences for holidays are used (non-given entries are always skipped). Please note the last abstract of this text part which informs you more detailed of this context. See [Environment Variable `GCALANSI`], page 237, too.
>
> Control code definitions may contain any printable characters. Non-printable characters may be encoded in octal or hexadecimal notation. The abbreviation ‘`\E`’ directly encodes the escape character (octal `\033` respectively hexadecimal `\x1B`).
>
> A character can be encoded octal by typing ‘`\`*`nnn`*’ (backslash-octal digit(s)), where *n* must be a valid octal digit (0. . . 7). Normally, three octal digits must be given. If the octal character code consists of one or two octal digits, leading zeroes must be added, except the case, where the encoded octal character is given at last in the single sequence.
>
> A character can be encoded hexadecimal by typing ‘`\x`*`nn`*’ (backslash-x hexadecimal digit(s)), where *n* must be a valid hexadecimal digit (0. . . 9A. . . Fa. . . f). Normally, two hexadecimal digits must be given. If the hexadecimal character code consists of one hexadecimal digit, a leading zero must be added, except the case, where the encoded hexadecimal character is given at last in the single sequence.

[7] This means with a leading and a trailing blank.

If the sequence separator character, thus the ‘:’ (colon) character itself, is used as a marking character, it must be encoded either octal by \072 or hexadecimal by \x3A.

If the C Preprocessor symbol USE_PAGER was defined and the output of the program is redirected or used in a pipeline, the highlighting sequences are automatically converted into the according marking characters; if USE_PAGER was not defined, they remain untouched.

Incomplete or non-given highlighting sequences will be replaced by internal default ANSI escape highlighting sequences if a GCALANSI environment variable is defined; otherwise completely replaced by their according marking characters. See [Environment Variable GCALANSI], page 237.

'`--mail`[`=`*address*]'

Send Gcal's output via `mail`[1] program to the given *address*, e.g.:

```
--mail=esken@gmx.net
```

If no *address* is given, Gcal tries to send the eMail by using the following methods:

1. If an environment variable `MAILTO` is defined and set, the eMail is send to the address which is listed in this environment variable. See [Environment Variable `MAILTO`], page 238, for more information.
2. If above action fails, and if an environment variable `USER` is defined and set, the eMail is send to the address which is listed in this environment variable. See [Environment Variable `USER`], page 238, for more information.
3. If above action fails, and if an environment variable `LOGNAME` is defined and set, the eMail is send to the address which is listed in this environment variable. See [Environment Variable `LOGNAME`], page 238, for more information.
4. If all above actions have failed, no eMail is send.

Generally, Gcal does not send electronic Mails whose message body is empty! An informational message will be shown on the *standard error channel* if this case occurs.

All highlighting sequences produced by Gcal itself are always disabled respectively automatically converted into the according marking characters if an eMail must be send; no matter if the '`--force-highlighting`' option was given or not. This behavior of Gcal is an imperative necessity, because it is possible that the `mail` program cannot perform the mailing correctly. Please pay attention in this context to the further explanations concerning the limitations of the *text part* of a resource file line (see Section 5.1.4 [Text part of a line], page 91).

If an environment variable `MAILPROG` is defined and set, its contents will be used as the program name of the mailer instead of the standard name `mail`. See [Environment Variable `MAILPROG`], page 238, for more information.

### 3.1.1.3 Calendar options

'`-n|N`[`-`]'
'`--holiday-list`[`=long|short`]'
'`--descending-holiday-list`[`=long|short`]'

Display the eternal holiday list. By default, there are no entries in the eternal holiday list. You have to choose country specific holidays or holidays from other calendar systems to provide the eternal holiday list with entries. See Chapter 4 [Eternal Holidays], page 73, and [Calendar option '`--cc-holidays=`*cc*[`+`...]'], page 16, for additional information.

'`-n`'
'`--holiday-list=long`'

Display all holidays of eternal holiday list —this means, all legal holidays and all further memorial days— sorted in ascending order.

'`-n-`'
'`--descending-holiday-list=long`'

Display all holidays of eternal holiday list —this means, all legal holidays and all further memorial days— sorted in descending order.

[1] See the standard manual pages for `mail`.

‘`-N`’<br>
‘`--holiday-list=short`’

Display legal holidays only of eternal holiday list, sorted in ascending order.

‘`-N-`’<br>
‘`--descending-holiday-list=short`’

Display legal holidays only of eternal holiday list, sorted in descending order.

‘`-G`’<br>
‘`--suppress-holiday-list-separator`’

Suppress displaying of the blank line which is always leading an eternal holiday list.

‘`-X`’<br>
‘`--exclude-holiday-list-title`’

Suppress the title text line of the eternal holiday list.

‘`--astronomical-holidays`’

Provide the eternal holiday list additionally with some astronomical data, and that the Full and New Moon phases, waning and waxing Half Moon phases, solar and lunar eclipses, and the solstices and equinoxes. See [Calendar option ‘`--time-offset=`*`argument`*’], page 31, how to change the timezone respectively base time for which the astronomical data is calculated.

‘`--bahai-holidays`’

Provide the eternal holiday list additionally with Bahá’ì holidays (only for dates after AD 1843).

‘`--celtic-holidays`’

Provide the eternal holiday list additionally with Celtic holidays. See [Calendar option ‘`--time-offset=`*`argument`*’], page 31, how to change the timezone for which the Celtic holidays are calculated.

‘`--chinese-flexible-holidays`’

Provide the eternal holiday list additionally with Chinese holidays, that are determined in a flexible manner (only for dates after AD 1644). See [Calendar option ‘`--time-offset=`*`argument`*’], page 31, how to change the timezone for which the Chinese holidays are calculated in a flexible manner. See Chapter 1 [Preface], page 1, for further details.

‘`--chinese-holidays`’

Provide the eternal holiday list additionally with Chinese holidays (only for dates after AD 1644). For dates until AD 1928 all computations done are depending fixed on Beijing local time, for later dates fixed on the timezone GMT-8. See Chapter 1 [Preface], page 1, for further details.

‘`--christian-holidays`’

Provide the eternal holiday list additionally with Christian holidays.

‘`--hebrew-holidays`’

Provide the eternal holiday list additionally with Hebrew holidays.

‘`--islamic-civil-holidays`’

Provide the eternal holiday list additionally with Islamic holidays (only for dates after AD 621), that are based on the civil Islamic calendar. See Chapter 1 [Preface], page 1, for further details.

'`--japanese-flexible-holidays`'

Provide the eternal holiday list additionally with Japanese holidays, that are determined in a flexible manner (only for dates after AD 1644). See [Calendar option '`--time-offset=`*`argument`*'], page 31, how to change the timezone for which the Japanese holidays are calculated in a flexible manner.

'`--japanese-holidays`'

Provide the eternal holiday list additionally with Japanese holidays (only for dates after AD 1644). For dates until AD 1887 all computations done are depending fixed on Beijing local time, for later dates fixed on the timezone GMT-9.

'`--multicultural-new-year-holidays`'

Provide the eternal holiday list additionally with multicultural New Year holidays. See [Calendar option '`--time-offset=`*`argument`*'], page 31, how to change the timezone for which the multicultural New Year holidays are calculated.

'`--orthodox-new-holidays`'

Provide the eternal holiday list additionally with Orthodox new calendar holidays, and it is assumed that the Gregorian Reformation has occurred from 10th till 22nd March 1924. See [Calendar option '`--orthodox-calendar`'], page 28, for further details.

'`--orthodox-old-holidays`'

Provide the eternal holiday list additionally with Orthodox new calendar holidays, and it is assumed that the Gregorian Reformation has occurred from 10th till 22nd March 1924. See [Calendar option '`--orthodox-calendar`'], page 28, for further details.

'`--persian-jalaali-holidays`'

Provide the eternal holiday list additionally with Persian holidays (only for dates after AD 621), which are based on the Persian *Jalaali* calendar. All computations done are depending on the timezone GMT-3.5.

'`--zodiacal-marker-holidays`'

Provide the eternal holiday list additionally with zodiacal marker holidays, i.e. the dates when the Sun enters a zodiac sign or when the Sun reaches the turning-point in the zodiac sign. See [Calendar option '`--time-offset=`*`argument`*'], page 31, how to change the timezone for which the zodiacal marker holidays are calculated.

'`-q` *`cc`*`[_`*`tt`*`][+...]`'
'`--cc-holidays=`*`cc`*`[_`*`tt`*`][+...]`'

Provide the eternal holiday list additionally with country specific holidays. Furthermore, all additionally highlighted days of the eternal holiday list are highlighted in the calendar sheets, too.

The *cc* argument is a two-letter country code as defined by the ISO-3166[2] like '`BE`' for Belgium or '`ES`' for Spain. See the pertinent literature for more details.

In some cases, such a country code may be trailed by a two-letter territory code *tt* for better specification, which is separated by a '`_`' character from the country code.

You can use more than one country code *cc*[_*tt*] by connecting them with a '`+`' character, e.g.:

> '`--cc-holidays=be+Fr+IT`' resp.,
> '`-q be+Fr+IT`'

includes all the country specific holidays given in the preceding argument into the eternal holiday list, i.e. Belgian, French and Italian holidays.

[2] ISO is the abbreviation of *International Organization for Standardization*.

Actually, Gcal respects the following country codes, at which countries marked by a ‘`#`’ character have only an incomplete recording of holidays:

| | |
|---|---|
| `AD` | Andorra |
| `AE` | United Arab Emirates |
| `AF` | Afghanistan |
| `AG` | Antigua and Barbuda |
| `AI` | Anguilla |
| `AL` | Albania |
| `AM` | Armenia |
| `AN_BO` | Netherlands Antilles/Bonaire |
| `AN_CU` | Netherlands Antilles/Curaçao |
| `AN_MA` | Netherlands Antilles/St Maarten |
| `AN_SA` | Netherlands Antilles/Saba and Statia |
| `AO` | Angola # |
| `AR` | Argentina |
| `AS` | American Samoa |
| `AT` | Austria |
| `AU_CT` | Australia/Canberra |
| `AU_NT` | Australia/Northern Territory |
| `AU_QU` | Australia/Queensland |
| `AU_SA` | Australia/Southern Australia |
| `AU_SW` | Australia/New South Wales |
| `AU_TA` | Australia/Tasmania |
| `AU_VI` | Australia/Victoria |
| `AU_WA` | Australia/Western Australia |
| `AW` | Aruba |
| `AZ` | Azerbaijan |
| `BA` | Bosnia-Herzegovina |
| `BB` | Barbados |
| `BD` | Bangladesh # |
| `BE` | Belgium |
| `BF` | Burkina Faso |
| `BG` | Bulgaria |
| `BH` | Bahrain |
| `BI` | Burundi # |

| | |
|---|---|
| BJ | Benin # |
| BM | Bermuda |
| BN | Brunei |
| BO | Bolivia |
| BR | Brazil |
| BS | Bahamas |
| BT | Bhutan # |
| BV | Bouvet Island |
| BW | Botswana |
| BY | Belarus |
| BZ | Belize |
| CA_AL | Canada/Alberta |
| CA_BC | Canada/British Columbia |
| CA_MA | Canada/Manitoba |
| CA_NB | Canada/New Brunswick |
| CA_NF | Canada/Newfoundland and Labrador |
| CA_NS | Canada/Nova Scotia |
| CA_NW | Canada/Nordwest Territories |
| CA_ON | Canada/Ontario |
| CA_PE | Canada/Prince Edward Island |
| CA_QU | Canada/Québec |
| CA_SA | Canada/Saskatchewan |
| CA_YU | Canada/Yukon |
| CC | Cocos Islands (Keeling) |
| CD | Democratic Republic of Congo # |
| CF | Central African Republic # |
| CG | Republic of Congo # |
| CH_AA | Switzerland/Appenzell Ausserrhoden |
| CH_AG | Switzerland/Aargau |
| CH_AI | Switzerland/Appenzell Innerrhoden |
| CH_BL | Switzerland/Basel-Land |
| CH_BN | Switzerland/Bern |
| CH_BS | Switzerland/Basel-Stadt |
| CH_FB | Switzerland/Fribourg |
| CH_GB | Switzerland/Graubünden |
| CH_GL | Switzerland/Glarus |

| | |
|---|---|
| CH_GV | Switzerland/Genève |
| CH_JR | Switzerland/Jura |
| CH_LZ | Switzerland/Luzern |
| CH_NC | Switzerland/Neuchâtel |
| CH_NW | Switzerland/Nidwalden |
| CH_OW | Switzerland/Obwalden |
| CH_SG | Switzerland/St Gallen |
| CH_ST | Switzerland/Solothurn |
| CH_SW | Switzerland/Schwyz |
| CH_TC | Switzerland/Ticino |
| CH_TG | Switzerland/Thurgau |
| CH_UI | Switzerland/Uri |
| CH_VD | Switzerland/Vaud |
| CH_VL | Switzerland/Valais |
| CH_ZG | Switzerland/Zug |
| CH_ZR | Switzerland/Zürich |
| CI | Côte d'Ivoire |
| CK | Cook Islands |
| CL | Chile |
| CM | Cameroon |
| CN | China |
| CO | Colombia |
| CR | Costa Rica |
| CU | Cuba |
| CV | Cape Verde |
| CX | Christmas Islands |
| CY | Cyprus |
| CZ | Czech Republic |
| DE_BA | Germany/Bavaria |
| DE_BB | Germany/Brandenburg |
| DE_BL | Germany/Berlin |
| DE_BR | Germany/Bremen |
| DE_BW | Germany/Baden-Württemberg |
| DE_HA | Germany/Hamburg |
| DE_HS | Germany/Hesse |
| DE_MV | Germany/Mecklenburg-West Pomerania |

| | |
|---|---|
| DE_NS | Germany/Lower Saxony |
| DE_NW | Germany/North Rhine-Westphalia |
| DE_RP | Germany/Rhineland Palatinate |
| DE_SA | Germany/Saxony-Anhalt |
| DE_SH | Germany/Schleswig-Holstein |
| DE_SL | Germany/Saar |
| DE_SN | Germany/Saxony |
| DE_TR | Germany/Thuringia |
| DJ | Djibouti |
| DK | Denmark |
| DM | Dominica |
| DO | Dominican Republic |
| DZ | Algeria |
| EC | Ecuador |
| EE | Estonia |
| EG | Egypt |
| EH | Western Sahara |
| ER | Eritrea |
| ES | Spain |
| ET | Ethiopia |
| FI | Finland |
| FJ | Fiji # |
| FK | Falkland Islands (Malvinas) |
| FM | Federated States of Micronesia |
| FO | Faroes |
| FR | France |
| GA | Gabon # |
| GB_EN | Great Britain/England and Wales |
| GB_NI | Great Britain/Northern Ireland |
| GB_SL | Great Britain/Scotland |
| GD | Grenada |
| GE | Georgia |
| GF | French Guiana |
| GH | Ghana |
| GI | Gibraltar |
| GL | Greenland |

| | |
|---|---|
| GM | Gambia |
| GN | Guinea |
| GP | Guadeloupe |
| GQ | Equatorial Guinea |
| GR | Greece |
| GS | South Georgia and South Sandwich Islands |
| GT | Guatemala |
| GU | Guam |
| GW | Guinea-Bissau # |
| GY | Guyana # |
| HK | Hong Kong |
| HM | Heard and Mc Donald Islands |
| HN | Honduras |
| HR | Croatia |
| HT | Haiti |
| HU | Hungary |
| ID | Indonesia # |
| IE | Ireland |
| IL | Israel |
| IN | India # |
| IQ | Iraq |
| IR | Islamic Republic of Iran |
| IS | Iceland |
| IT | Italy |
| JM | Jamaica |
| JO | Jordan |
| JP | Japan |
| KE | Kenya # |
| KG | Kyrgyzstan |
| KH | Cambodia # |
| KI | Kiribati # |
| KM | Comoros |
| KN | St Kitts and Nevis |
| KP | Democratic People's Republic of Korea # |
| KR | Republic of Korea |
| KW | Kuwait |

| | |
|---|---|
| KY | Cayman Islands |
| KZ | Kazakhstan |
| LA | Laos People's Democratic Republic # |
| LB | Lebanon |
| LC | St Lucia |
| LI | Liechtenstein |
| LK | Sri Lanka # |
| LR | Liberia |
| LS | Lesotho |
| LT | Lithuania |
| LU | Luxembourg |
| LV | Latvia |
| LY | Libyan Arab Jamahiriya (Libya) |
| MA | Morocco |
| MC | Monaco |
| MD | Republic of Moldova |
| MG | Madagascar # |
| MH | Marshall Islands |
| MK | Former Yugoslav Republic of Macedonia |
| ML | Mali |
| MN | Mongolia # |
| MO | Macau |
| MP | Northern Marian Islands (Saipan) |
| MQ | Martinique |
| MR | Mauritania |
| MS | Montserrat |
| MT | Malta |
| MU | Mauritius |
| MV | Maldives |
| MW | Malawi |
| MX | Mexico |
| MY | Malaysia # |
| MZ | Mozambique |
| NA | Namibia |
| NC | New Caledonia |
| NE | Niger |

| | |
|---|---|
| `NF` | Norfolk Islands |
| `NG` | Nigeria |
| `NI` | Nicaragua |
| `NL` | Netherlands |
| `NM` | Myanmar (Burma) # |
| `NO` | Norway |
| `NP` | Nepal # |
| `NR` | Nauru |
| `NU` | Niue |
| `NZ` | New Zealand |
| `OM` | Oman |
| `PA` | Panama |
| `PE` | Peru |
| `PF` | French Polynesia |
| `PG` | Papua New Guinea |
| `PH` | Philippines |
| `PK` | Pakistan |
| `PL` | Poland |
| `PM` | Saint-Pierre and Miquelon |
| `PN` | Pitcairn |
| `PR` | Puerto Rico |
| `PT` | Portugal |
| `PW` | Palau |
| `PY` | Paraguay |
| `QA` | Qatar |
| `RE` | Réunion |
| `RO` | Romania |
| `RU` | Russian Federation |
| `RW` | Rwanda |
| `SA` | Saudi Arabia |
| `SB` | Solomon Islands |
| `SC` | Seychellen |
| `SD` | Sudan |
| `SE` | Sweden |
| `SG` | Singapore # |
| `SH` | St Helena |

| | |
|---|---|
| `SI` | Slovenia |
| `SJ` | Svalbard and Jan Mayen Islands |
| `SK` | Slovakia |
| `SL` | Sierra Leone |
| `SM` | San Marino |
| `SN` | Senegal |
| `SO` | Somalia |
| `SR` | Suriname # |
| `ST` | Sao Tomé and Principe |
| `SV` | El Salvador |
| `SY` | Syrian Arab Republic (Syria) |
| `SZ` | Swaziland |
| `TC` | Turks and Caicos Islands |
| `TD` | Chad |
| `TG` | Togo # |
| `TH` | Thailand # |
| `TJ` | Tajikistan |
| `TK` | Tokelau |
| `TM` | Turkmenistan |
| `TN` | Tunisia |
| `TO` | Tonga |
| `TR` | Turkey |
| `TT` | Trinidad and Tobago # |
| `TV` | Tuvalu |
| `TW` | Taiwan |
| `TZ` | Tanzania |
| `UA` | Ukraine |
| `UG` | Uganda |
| `US_AK` | United States/Alaska |
| `US_AL` | United States/Alabama |
| `US_AR` | United States/Arkansas |
| `US_AZ` | United States/Arizona |
| `US_CA` | United States/California |
| `US_CO` | United States/Colorado |
| `US_CT` | United States/Connecticut |
| `US_DC` | United States/District of Columbia |

| | |
|---|---|
| `US_DE` | United States/Delaware |
| `US_FL` | United States/Florida |
| `US_GA` | United States/Georgia |
| `US_HI` | United States/Hawaii |
| `US_IA` | United States/Iowa |
| `US_ID` | United States/Idaho |
| `US_IL` | United States/Illinois |
| `US_IN` | United States/Indiana |
| `US_KS` | United States/Kansas |
| `US_KY` | United States/Kentucky |
| `US_LA` | United States/Louisiana |
| `US_MA` | United States/Massachusetts |
| `US_MD` | United States/Maryland |
| `US_ME` | United States/Maine |
| `US_MI` | United States/Michigan |
| `US_MN` | United States/Minnesota |
| `US_MO` | United States/Missouri |
| `US_MS` | United States/Mississippi |
| `US_MT` | United States/Montana |
| `US_NC` | United States/North Carolina |
| `US_ND` | United States/North Dakota |
| `US_NE` | United States/Nebraska |
| `US_NH` | United States/New Hampshire |
| `US_NJ` | United States/New Jersey |
| `US_NM` | United States/New Mexico |
| `US_NV` | United States/Nevada |
| `US_NY` | United States/New York |
| `US_OH` | United States/Ohio |
| `US_OK` | United States/Oklahoma |
| `US_OR` | United States/Oregon |
| `US_PA` | United States/Pennsylvania |
| `US_RI` | United States/Rhode Island |
| `US_SC` | United States/South Carolina |
| `US_SD` | United States/South Dakota |
| `US_TN` | United States/Tennessee |
| `US_TX` | United States/Texas |

| | |
|---|---|
| `US_UT` | United States/Utah |
| `US_VA` | United States/Virginia |
| `US_VT` | United States/Vermont |
| `US_WA` | United States/Washington |
| `US_WI` | United States/Wisconsin |
| `US_WV` | United States/West Virginia |
| `US_WY` | United States/Wyoming |
| `UY` | Uruguay |
| `UZ` | Uzbekistan |
| `VC` | St Vincent and Grenadines |
| `VE` | Venezuela |
| `VG` | British Virgin Islands |
| `VI` | U.S. Virgin Islands |
| `VN` | Viet Nam |
| `VU` | Vanuatu |
| `WF` | Wallis and Futuna Islands |
| `WS` | Samoa |
| `YE` | Yemen |
| `YT` | Mayotte |
| `YU` | Serbia and Montenegro |
| `ZA` | South Africa |
| `ZM` | Zambia |
| `ZW` | Zimbabwe |

‘`--bahai-months`’

Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Bahá’ì calendar. See [Calendar option ‘`--bahai-holidays`’], page 15, for further details.

‘`--chinese-flexible-months`’

Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Chinese calendar, that is determined in a flexible manner. See [Calendar option ‘`--chinese-flexible-holidays`’], page 15, for further details.

‘`--chinese-months`’

Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Chinese calendar. See [Calendar option ‘`--chinese-holidays`’], page 15, for further details.

‘`--coptic-months`’

Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Coptic calendar (only for dates after AD 283).

‘--ethiopic-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Ethiopic calendar.

‘--french-revolutionary-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the French Revolutionary calendar (only for dates after AD 1791).

‘--hebrew-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Hebrew calendar.

‘--indian-civil-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the civil Indian calendar (only for dates after AD 1956).

‘--islamic-civil-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the civil Islamic calendar. See [Calendar option ‘--islamic-civil-holidays’], page 15, for further details.

‘--japanese-flexible-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Japanese calendar, that is determined in a flexible manner. See [Calendar option ‘--japanese-flexible-holidays’], page 16, for further details.

‘--japanese-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Japanese calendar. See [Calendar option ‘--japanese-holidays’], page 16, for further details.

‘--old-armenic-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Old-Armenic calendar (only for dates after AD 551).

‘--old-egyptic-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Old-Egyptic calendar.

‘--persian-jalaali-months’
: Provide the eternal holiday list additionally with the starting dates of the common and leap months, as they result from the Persian *Jalaali* calendar. See [Calendar option ‘--persian-jalaali-holidays’], page 16, for further details.

‘-i[-]’
‘--type=special|standard’
: To obtain the *standard calendar format*[3], either start Gcal omitting the ‘-i[-]’ option because it is set by default, Appendix C [Aspects in Internationalization], page 123, or start Gcal with the ‘-i-’ respectively ‘--type=standard’ option:

[3] Similar the BSD-cal program.

```
$ gcal -i-
⊣
⊣    September 1994
⊣ Su Mo Tu We Th Fr Sa
⊣              1  2  3
⊣  4  5  6  7  8  9 10
⊣ 11 12 13 14 15 16 17
⊣ 18 19 20 21 22 23 24
⊣ 25 26 27 28 29 30
```

To obtain the *special calendar format*, start Gcal with the '`-i`' respectively '`--type=special`' option:

```
$ gcal -i
⊣
⊣ September 1994
⊣
⊣ Sunday           4 11 18 25
⊣ Monday           5 12 19 26
⊣ Tuesday          6 13 20 27
⊣ Wednesday        7 14 21 28
⊣ Thursday      1  8 15 22 29
⊣ Friday        2  9 16 23 30
⊣ Saturday      3 10 17 24
```

'`-O`'
'`--orthodox-calendar`'

Use the leap year rule as used by the Eastern Orthodox churches.

Without specifying the '`--orthodox-calendar`' option, Gcal is unable to display Gregorian years later than 2799 in the correct way for the Eastern churches, because they use a different scheme for calculating the leap years. The method for computing leap years within the common Gregorian calendar, which Gcal uses by default, is as follows:

> A leap year is any year which number can be divided by 4 without a remainder, and years ending in hundreds are no leap years unless they are divisible by 400.

But the Eastern Orthodox churches compute leap years within the Gregorian calendar by using another rule:

> A leap year is any year which number can be divided by 4 without a remainder, and years ending in hundreds are leap years, if a remainder of 2 or 6 occurs when such a year is divided by 9.

The first difference therefore occurs in the year 2800 which is a leap year in the common Gregorian calendar, but an ordinary year only in the calendar as used by the Eastern Orthodox churches.

'`-K`'
'`--with-week-number`'

Provide the calendar sheet with week numbers. See [Calendar option '`--starting-day=`*`argument`*'], page 30, [Calendar option '`--iso-week-number=yes|no`'], page 29, and Appendix C [Aspects in Internationalization], page 123, for further details.

'`--iso-week-number=yes|no`'

Determine the type of week numbers which are used in the calendar sheet, in the fixed date list and by the '`%`*`date`*' actual date modifier. See [Calendar option '`--starting-day=`*`argument`*'], page 30, and Appendix C [Aspects in Internationalization], page 123, for more details.

'`--iso-week-number=yes`'

The methods of the ISO-8601:1988 are used for detecting week numbers; this means a week starts on Mondays, and the first week of a year is the one which includes the first Thursday; equivalently, the one which includes the 4th January. This method is called *ISO week number* in the further context. If the starting day of the week is not set to Monday, the week numbers are not represented correctly in most cases. If you use this option, you should take care of setting Monday as the starting day of the week!

'`--iso-week-number=no`'

Weeks start on the respective starting day of the week, and the days in a new year that are preceding the first starting day of the week are in the last week of the previous year, respectively in week zero of the new year. This method is called *standard week number* in the further context.

'`-u`'
'`--suppress-calendar`'

Suppress output of calendar sheet explicitly.

'`-b` *`number`*'
'`--blocks=`*`number`*'

Set *number* of calendar sheet blocks (valid arguments: '`1|2|3|4|6|12`'). The default *number* for the *standard calendar format* is '`-b4 `' respectively '`--blocks=4`', and for the *special calendar format* '`-b 3`' respectively '`--blocks=3`'. If this option is found, the program sees that a year calendar output is desired!

'`-b 1`'
'`--blocks=1`'

Displays one block with twelve months at a time.

'`-b 2`'
'`--blocks=2`'

Displays two blocks with six months at a time.

'`-b 3`'
'`--blocks=3`'

Displays three blocks with four months at a time.

'`-b 4`'
'`--blocks=4`'

Displays four blocks with three months at a time.

'`-b 6`'
'`--blocks=6`'

Displays six blocks with two months at a time.

'`-b 12`'
'`--blocks=12`'

Displays twelve blocks with one month at a time.

'`-j`[b]'
'`--calendar-dates=special|both`'

Use alternative date format in calendar sheet instead of the default standard format which displays the days of month in consecutive manner.

'`-j`'
'`--calendar-dates=special`'

Display the calendar sheet by using the special date format. This means, the days of year are displayed in consecutive manner instead of the days of month.

'`-jb`'
'`--calendar-dates=both`'

Display the calendar sheet by using both the standard date format and special date format.

'`-jn`[b]'
'`--holiday-dates=special|both`'

Use alternative date format in eternal holiday list instead of the default standard format which displays the days of month in consecutive manner. See [Calendar option '`--holiday-list`[=`long|short`]'], page 14.

'`-jn`'
'`--holiday-dates=special`'

Display the eternal holiday list by using the special date format. This means, the days of year are displayed in consecutive manner instead of the days of month.

'`-jnb`'
'`--holiday-dates=both`'

Display the eternal holiday list by using both the standard date format and special date format.

'`-jc`[b]'
'`--fixed-dates=special|both`'

Use alternative date format in fixed date list instead of the default standard format which displays the days of month in consecutive manner. See [Fixed date option '`--list-of-fixed-dates`[=`short|long`]'], page 41.

'`-jc`'
'`--fixed-dates=special`'

Display the fixed date list by using the special date format. This means, the days of year are displayed in consecutive manner instead of the days of month.

'`-jcb`'
'`--fixed-dates=both`'

Display the fixed date list by using both the standard date format and special date format.

'`-s` *argument*'
'`--starting-day=`*argument*'

Set the starting day of the week (valid *argument*: '`0, 1...7 | today |` *weekday name*').

For example:

`--starting-day=Sunday` or
`--starting-day=7` or
`-s SUNDAY` or
`-s sund` or
`-sSu` or
`-s 7`

thus all specifies the Sunday (1≡Mon, 2≡Tue . . . 7≡Sun).

If the '`-s today`' option (or '`--starting-day=today`') or the '`-s 0`' option (or '`--starting-day=0`') is given, the starting day of the week is set to the actual weekday as it is delivered by the system date. See Appendix C [Aspects in Internationalization], page 123, for more details.

'`--time-offset=`*`argument`*'

Change the base time of the astronomical functions (valid *argument*: '`t|@|[t|@][+|-]`*`mmmm`*`|`*`hh`*`:[`*`mm`*`]`').

If no '`--time-offset=`*`argument`*' option is given, the astronomical data that is inserted into the eternal holiday list by the '`--astronomical-holidays`' option (see [Calendar option '`--astronomical-holidays`'], page 15), and all Sun and Moon related special texts are always calculated for 0 o'clock Universal time (UTC/GMT), thus *civil midnight time*.

See Section G.2.2.5 [Sun data], page 155, and Section G.2.2.6 [Moon data], page 176, likewise Section G.2.1.9 [Moon phase], page 146, for further information.

The *argument* is either the '`t`' or the '`@`' character —where '`t`' means a relation to the actual local time[4] and '`@`' denotes a relation to the actual Universal time—, or one of these characters followed by a displacement value, or only a displacement value which has to be specified either by using the `[+|-]`*`mmmm`* format or the `[+|-]`*`hh`*`:[`*`mm`*`]` format. `[+|-]`*`mmmm`* adds respectively subtracts the specified amount of minutes *mmmm* from the base time value 0 o'clock Universal time (range `0...9999`), while `[+|-]`*`hh`*`:[`*`mm`*`]` adds respectively subtracts the given amount of hours *hh* (range `0...99`) and minutes *mm* (range `0...59`) from the base time value 0 o'clock Universal time. The displacement value is always added to the base time value 0 o'clock Universal time in case it is specified without a `+|-` sign.

For example, the '`--time-offset=+1:`' option causes that while displaying eternal holiday lists and fixed date lists, the time 0 o'clock of the timezone GMT-1 (≡ CET) is used as the base time by the astronomical functions instead of the base time 0 o'clock Universal time (≡ GMT).

For example, the '`--time-offset=t-2:`' option causes that while displaying eternal holiday lists and fixed date lists, the time that is two hours earlier than the actual local time is used as the base time by the astronomical functions instead of the base time 0 o'clock Universal time (≡ GMT). For such a kind of relation, the term *relative time offset value* will be used in the further context.

'`--transform-year=`*`argument`*'

Change the base year of calendar (valid *argument*: '`-9999...[+]9999`').

For example, the '`--transform-year=-543`' option causes that while displaying calendar sheets, eternal holiday lists and fixed date lists, the year 543 BC is used as the base year of the calendar instead of the year AD 1. This results in the case that —for example— for the year 1999 (Christian era), the year number 1999 is not used in the above mentioned outputs, but the year number 2542 which is used in the western oriented Thai calendar.

[4] The time which is set as system time in the computer's clock.

Nevertheless, Gcal does not respect the '`--transform-year=`*argument*' option when using the actual date modifier '`%`*date*' (see Section 3.1.3 [Actual date modifier], page 62), the commands (see Section 3.1.4 [Commands], page 67), and the fixed date entries in a resource file (see Section 5.1.2 [Date part of a line], page 81). There, all references made to a definite year are always treated in the way that the year AD 1 is the base year of the calendar, so in fact it is **only** possible to use references based on the Christian era.

'`--gregorian-reform=1582|1700|1752|1753|`*argument*'

Set the period which was skipped during the Gregorian Reformation. By default, Gcal runs in the *hybrid* calendar mode, i.e. Gcal automatically changes from the Julian calendar system to the Gregorian calendar system if output is related to dates after the Gregorian Reformation has happened. See Appendix C [Aspects in Internationalization], page 123, for more details. Actually, four fixed default periods are supported, and that of the year 1582, of the year 1700, of the year 1752 and of the year 1753.

If Gcal is called with the '`--gregorian-reform=1582`' option, it assumes the Gregorian Reformation has occurred from 5th till 14th October 1582.

If Gcal is called with the '`--gregorian-reform=1700`' option, it assumes the Gregorian Reformation has occurred from 19th till 28th February 1700.

If Gcal is called with the '`--gregorian-reform=1752`' option, it assumes the Gregorian Reformation has occurred from 3rd till 13th September 1752.

If Gcal is called with the '`--gregorian-reform=1753`' option, it assumes the Gregorian Reformation has occurred from 18th till 28th February 1753.

In case another period shall be respected, it can be arranged by the option *argument* like '*yyyyy*`,`*mm*`,`*first-day*`,`*last-day*'. If the Gregorian Reformation has occurred for example on the 7th till the 17th April 1802, this can be arranged as follows:

```
--gregorian-reform=1802,4,7,17
```

Gcal is able to represent so-called *proleptic* calendars of a definite calendar system. This means, Gcal only uses a definite calendar system during a definite period, although there was a change to another calendar system in the historic reality during this definite period. The following proleptic calendar systems are actually supported by Gcal:

- Proleptic Gregorian calendars can be created from AD 302 until AD 9999 if '`--gregorian-reform=301,12,31,31`' is used. Unfortunately, Gcal actually cannot represent years before AD 302 in a proleptic-Gregorian manner.
- Proleptic Julian calendars can be created from AD 1 until AD 9999 if '`--gregorian-reform=10002,12,31,31`' is used. In this case, the actual system date that is normally based on the Gregorian calendar and that is used by Gcal, is internally and automatically converted into the according Julian date[5]. The same happens if the actual system date is modified by using the actual date modifier '`%`*date*'. In such a case, Gcal also assumes that the given date is a Gregorian date, and converts it automatically into the according Julian date internally. See Section 3.1.3 [Actual date modifier], page 62, for further details.

Please note that it is possible to corrupt the calendars likewise the fixed date feature logically (which works correctly now for the year in which the Gregorian Reformation

[5] The automatical conversion into the according Julian date is always done if the actual system date, that is based on the Gregorian calendar, is before the date of the Gregorian Reformation.

has occurred) if the argument of the '--gregorian-reform' option is not used with care.

'--date-format=de|us|gb|*text*'

Set the date format which affects the ordering and representation of a displayed date. See Appendix C [Aspects in Internationalization], page 123, for more details. The date format text is respected by Gcal in the eternal holiday list, in the fixed date list and the calendar sheets. Moreover, Gcal internally tries to obtain the best representation of a displayed date in case the day-of-year numbers instead of the day-of-month numbers must be displayed, or both types of numbers are used in a combined manner.

Actually, three fixed default date formats are supported, and that for German users, U.S. American users and for users in Great Britain.

If Gcal is called with the '--date-format=de' option,
the '%<2#K, %1%>2*D%2 %<3#U %>04*Y' date format text is used. This result in that a date is displayed by using the '*ww*, *dd mmm yyyy*' ordering, for example 'Sa, 28 Aug 1999'.

If Gcal is called with the '--date-format=us' option,
the '%<3#K, %<3#U %1%>2&*D%2 %>04*Y' date format text is used. This result in that a date is displayed by using the '*www*, *mmm dd yyyy*' ordering, for example 'Sat, Aug 28th 1999'.

If Gcal is called with the '--date-format=gb' option,
the '%<3#K, %1%>2&*D%2 %<3#U %>04*Y' date format text is used. This result in that a date is displayed by using the '*www*, *dd mmm yyyy*' ordering, for example 'Sat, 28th Aug 1999'.

In case another format text shall be respected, this format text can either be set in the GCAL_DATE_FORMAT environment variable[6], or it can be arranged by the option argument *text*. For example, '--date-format='%Y %D %>02*M ; %1(%>5u#K)%2'' displays a date by using the '[[[*y*]*y*]*y*]*y* [*d*]*d mm* ; (*wwwww*)' ordering, thus for example '1999 28 08 ; (SATUR)'. See [Environment Variable GCAL_DATE_FORMAT], page 237, for further information.

The format text may contain on the one hand all characters which can be managed by Gcal, and on the other hand character replacement instructions and format elements which are transformed into their according values at run-time. Some format elements may have a format instruction which is called *format* in the further context. See Appendix J [Format Instruction], page 231, for the detailed description of the format instruction and its components.

A minimum date format text must contain the following format elements minimum:

> The day number, one component of the month group, the year number and both components of the highlighting group.

A weekday name format element may be included optionally into the date format text. The following format elements and character replacement instructions are currently supported:

%[*format*]D

Day number (must be defined)

%[*format*]Y

Year number (must be defined)

[6] But GCAL_DATE_FORMAT is only used in case no '--date-format=de|us|gb|*text*' option is given.

`%[`*format*`]K`
Weekday name (may be defined)

Month group (exactly one member must be defined):

`%[`*format*`]M`
Month number

`%[`*format*`]U`
Month name

Highlighting group (all members must be defined and `%1` must be specified before `%2`):

`%1` Start of highlighting sequence / marking character

`%2` End of highlighting sequence / marking character

Character replacement instructions:

`_` Space/blank character ‘ ’

`\_` Underscore character ‘`_`’

`\%` Percent character ‘`%`’

`\\` Backslash character ‘`\`’

See Section I.3 [Table of Obsolete Date Formats], page 229, and Section I.2 [Table of Obsolete Date Format Elements], page 229, for further information.

‘`--translate-string=`*text*’
Define the country specific special character pairs which are respected or translated by a *style* format instruction component, respectively. See Appendix J [Format Instruction], page 231, for the detailed description of the format instruction and its components. The country specific special character pairs are arranged by the *text* option argument as a sequence of single character pairs, and that in any number and order. The upper-case representation of the country specific special character has to be specified at first, and after that its lower-case representation. Country specific special characters which do not have an upper-case resp., lower-case representation in the character set used, like e.g. the ‘`ß`’-character which is very usual in the German character set, are also specified as a special character pair (here: ‘`ßß`’), otherwise these special characters are not recognized and they are converted incorrectly by the *style* format instruction component. In case the lower-case representation of the country specific special character is specified at first, and after that its upper-case representation, this option does not cause any further affects to the *style* format instruction component; resulting, the country specific special characters specified are not recognized as such and they remain untranslated therefore.

For example, a ‘`ÄäÖöÜüßß`’ option argument causes the correct conversion of the preceding special characters in an individual date format, which has a *style* format instruction component, and that, how they are used by the character set used in Germany.

#### 3.1.1.4 Fixed date options

‘`-v` *argument*’
‘`--date-variable=`*argument*’
Define *global* date variable which is visible in all referenced resource files (see Section 5.1.8 [Date variables], page 97). The option *argument* must either be a valid single date variable definition or a (‘`:`’) colon-connected series of definitions.

A date variable name *dvar* consists of a single, case-insensitive letter[7]. In a definition, this name is trailed by the assignment operator character ‘`=`’, and the numerical values of month *mm* (or a short, three characters month name, e.g. ‘`Jan`’, ‘`Feb`’. . . ) and day *dd*, this variable has to store[8].

For example:

> ‘`--date-variable=a=1127:c=a`’ respectively
> ‘`-v a=1127:c=a`’
> stores the 27th November into the global date variable *a* and into the global date variable *c*.

‘`--export-date-variables`’

Causes the export of local date variables from one resource file to another resource file. See Section 5.1.8 [Date variables], page 97, for more details.

‘`-r` *argument*’
‘`--text-variable=`*argument*’

Define *global* text variable which is visible in all referenced resource files (see Section 5.1.9 [Text variables], page 100). The option *argument* must either be a valid single text variable definition or a (‘`:`’) colon-connected series of definitions.

A text variable name *tvar* consists of the ‘`$`’ prefix and a single, case-insensitive letter.

A text variable is defined as follows:

- If the name of a text variable is trailed by the assignment operator character ‘`=`’, all the text which appears at the right side of this assignment operator is assigned to this text variable.
- If the name of a text variable is trailed by the assignment operator character ‘`?`’, all the text which appears at the right side of this assignment operator is assumed as an external command and executed by Gcal. The output, which is created on the *standard output channel* by the command run, is assigned to this text variable; and that in an interpreted manner. This means, Gcal interprets all special texts and references to other text variables which are contained in this output. See Section 5.1.9 [Text variables], page 100, and Section 5.1.5 [Special Texts processed], page 94, for further information.
- If the name of a text variable is trailed by the assignment operator character ‘`:`’, all the text which appears at the right side of this assignment operator is assumed as an external command and executed by Gcal. The output, which is created on the *standard output channel* by the command run, is assigned to this text variable; and that in an uninterpreted manner. This means, Gcal does not interpret any special texts and references to other text variables which are contained in this output. See also Section 5.1.9 [Text variables], page 100, and Section 5.1.5 [Special Texts processed], page 94, for further details.

Indeed, Gcal executes external commands only if the ‘`--execute-command`’ option is given at program start-up. See [Fixed date option ‘`--execute-command`’], page 40, for more details.

The text which appears at the right side of one of the valid assignment operator characters may reference names of already defined global text variables. These

[7] Except the date variable *e* which is internally reserved for the Easter Sunday’s date, so it cannot be assigned or operated therefore. And except the date variable *t* which is internally reserved for today’s date, so it likewise cannot be assigned or operated.

[8] Or another global date variable name which is already defined, but **no** date given in the ‘`e|t|`*dvar*`[+|-]`*n*`[`*www*`]`’, ‘*mmwwwn*’, or the ‘`*d|w`*n*`[`*www*`]`’ format.

references are converted to the according textual values just before the assignment is performed by Gcal.

A global text variable is defined with an *empty contents* in case no text appears at the right side of one of the valid assignment operator characters.

For example:

> '`--text-variable=$a=foo:$c=$a`' respectively
> '`-r $a=foo:$c=$a`'
> stores the '`foo`' text into the global text variable *$a* and into the global text variable *$c*.
>
> '`--text-variable=$a\:foo:$c?bar`' respectively
> '`-r $a\:foo:$c?bar`'
> stores the uninterpreted output of the `foo` command into the global text variable *$a*, and the interpreted output of the `bar` command into the global text variable *$c*.

You may depreciate the special meaning of the '`:`' separator character —in case this character itself is needed— by placing a '`\`' (backslash) character before it, e.g. '`\:`'. If you need the '`\:`' characters themselves, you have to protect the '`\`' (backslash) character by another '`\`' (backslash) character, e.g. '`\\:`'.

You may depreciate the special meaning of the '`$`' prefix character —in case this character itself is needed in the text— by placing a '`\`' (backslash) character before it, e.g. '`\$`'. If you need the '`\$`' characters themselves in the text, you have to protect the '`\`' (backslash) character by another '`\`' (backslash) character, e.g. '`\\$`'.

Each time a '`_`' (underscore) character is found in *argument*, this character is replaced by a real '` `' (space) character.

You may depreciate the special meaning of the '`_`' (underscore) character —in case this character itself is needed— by placing a '`\`' (backslash) character before it, e.g. '`\_`'. If you need the '`\_`' themselves, you have to protect the '`\`' (backslash) character by another '`\`' (backslash) character, e.g. '`\\_`'.

'`--export-text-variables`'
: Causes the export of local text variables from one resource file to another resource file. See Section 5.1.9 [Text variables], page 100, for more details.

'`--adjust-value=`*`argument`*'
: Causes on the one hand, that the number given as *argument* to the '`--adjust-value`' option is used as reference altitude to calculate the rise and set times by the Sun and Moon oriented special texts. The option argument must be a rational number in range -90.0. . .+90.0, which declares the angle distance in decimal degrees from the center of astronomical object to the horizon. For example, you can detect the time when the center of the Sun or the Moon is 20.5 decimal degrees below the horizon if you use '`--adjust-value=-20.5`'.

And causes on the other hand, that the number that is given as *argument* is used as factor for computing the shadow length of a vertical pole by the Sun oriented special texts. The option argument must be a rational number in range 0.0. . .+90.0, which declares the shadow length. For example, you can detect the time when the shadow length of vertical pole is three and a half times longer than the shadow length that appears at astronomical noon if you use '`--adjust-value=3.5`'.

See Section G.2.2.5 [Sun data], page 155, and Section G.2.2.6 [Moon data], page 176, for further information.

'`--atmosphere=`*`air-pressure`*[,*`temperature`*]'

Set the base data of the Earth's atmosphere that is necessary to calculate the amount of *refraction*[9], that is used by the Sun and Moon oriented special texts. The option argument consists of one or two rational numbers, which are separated by a '`,`' character. The first number denotes the air pressure in millibar, and the second number specifies the air temperature in degree Celsius. The default value for the air pressure is 1013.25 millibar, and the default value for the temperature is 15.0 degree Celsius. For example, if you use '`--atmosphere=1010,10`', you define atmospheric conditions as they are existing for an air pressure of 1010 millibar and a temperature of 10 degree Celsius. If you set an air pressure value less or equal zero, no atmospheric refraction will be respected by the Sun and Moon oriented special texts, as for example done by '`--atmosphere=0`'.

See Section G.2.2.5 [Sun data], page 155, and Section G.2.2.6 [Moon data], page 176, for further information.

'`--limit`'

Causes when calculating the rise and set times by the Sun oriented special texts for high latitudes, that a rise or set event which possibly occurs on a previous or next day, is not displayed as occurring on the actual day as it is done by default. At high latitudes, i.e. above the solar Arctic Circle, it is fact that no, or more than one sunrise or sunset happens during a day at definite dates of the year. As a special text basically leads in a single, atomical output, even it should result in several output in reality, Gcal puts such an event on a previous or next day. Then, data represented in this manner could be misinterpreted very easily.

The following example is used to explain this. It calculates the time the astronomical twilight ends for the location Münster, Germany, during some days in May 2000, and that with and without the '`--limit`' option in each case:

```
$ gcal %20000506 -f/dev/null -cxl4+ \
>       -#'0*d1#999_%s9+5158+00738,120'
⊣
⊣ Sun, May   7th 2000: 23:52
⊣ Mon, May   8th 2000: 23:57
⊣ Tue, May   9th 2000: 00:01
⊣ Wed, May  10th 2000: 00:06
$ gcal %20000506 -f/dev/null -cxl4+ --limit \
>       -#'0*d1#999_%s9+5158+00738,120'
⊣
⊣ Sun, May   7th 2000: 23:52
⊣ Mon, May   8th 2000: 23:57
⊣ Tue, May   9th 2000: **:**
⊣ Wed, May  10th 2000: **:**
```

As one can see well in the above example, the output for the days since 9th May differs in that the times shown without the '`--limit`' option obviously exceeded the civil midnight time and actually cannot happen during these days, whereas no times are shown for the days since 9th May if the '`--limit`' option is used because the time circle is exceeded. Astronomically seen, it is the case that the end of the astronomical twilight is at 23:57 o'clock local time on 8th May, no end of astronomical twilight

[9] Refraction is an optical phenomenon caused by the Earth's atmosphere, which leads to an apparent raising of the location of a celestial body. The amount of refraction increases with growing air pressure and sinking temperature and vice-versa.

happens on 9th May because the center of the Sun's disk is always above 18 degrees (the reference altitude of the astronomical twilight) below a geometric horizon, but on 10th May at 00:01 o'clock local time, this reference altitude is passed again. Thus, the time that is detected for the 9th May has properly to be moved to the 10th May, et cetera, as it is expressed by the output that is created by using the '`--limit`' option.

See Section G.2.2.5 [Sun data], page 155, for further information.

'`--cycle-end=`*`argument`*'

Calculates the Sun and Moon oriented special texts that are dynamical, i.e. depending on the respective clocktime, for a series of clocktimes for the current day, i.e. determination is done for any time interval that can be individually defined by the user. Gcal is always put into such a *cycle mode* if the '`--cycle-end`' option is given at program start-up. The *argument* of the '`--cycle-end`' option is syntactically according to the argument which may be given to the '`--time-offset`' option, thus '`t|@|[t|@][+|-]`*`mmmm`*`|`*`hh`*`:[`*`mm`*`]`'. See [Calendar option '`--time-offset=`*`argument`*'], page 31, for the detailed explanation of the preceding option argument. Thus, the '`--cycle-end`' option is used to fix the ending time of the time interval, whereas the '`--time-offset`' option is **implicitly** used to fix the starting time of the time interval, that is 0 o'clock by default[10].

If values for the time interval are given to both above mentioned options that fall short of 0 o'clock, these values are automatically set to 0 o'clock. If values for the time interval are given that exceed 23:39 o'clock, these values are automatically set to 23:59 o'clock.

The timestep value between the starting and the ending time of the time interval that is respected by Gcal is one minute by default. This value can at pleasure be changed by using the '`--cycle-step`' option. See [Fixed date option '`--cycle-step=`*`argument`*'], page 39, for more details.

You should always keep in mind that the use of the cycle mode is efficient only if the current time of the time interval is also part of that resource file line that has to be evaluated in this manner. Otherwise, it is difficult to find out unobjectionably lateron when displaying the data and times, which Sun or Moon oriented value resulted in what time.

The following example shows how the topocentric elevation angle of the Sun for the location Münster, Germany, is calculated during two days in April 2000, and that for the time interval 12:00 o'clock until 12:05 o'clock local time in each case:

[10] Correct, the '`--time-offset`' option has multiple modes of operation that are depending on the context of its use!

```
$ gcal %20000420 -Hno -f/dev/null -xdt -g::: \
>       --here='0*d1#999_%ot+00+000_:_%oa+5158+00738,120' \
>       --cycle-end=12:05 --time=12:00
⊣
⊣ Thu, Apr  20th 2000: +12h00': +045d58'
⊣ Thu, Apr  20th 2000: +12h01': +046d03'
⊣ Thu, Apr  20th 2000: +12h02': +046d07'
⊣ Thu, Apr  20th 2000: +12h03': +046d12'
⊣ Thu, Apr  20th 2000: +12h04': +046d17'
⊣ Thu, Apr  20th 2000: +12h05': +046d22'
⊣ :::
⊣ Fri, Apr  21st 2000: +12h00': +046d18'
⊣ Fri, Apr  21st 2000: +12h01': +046d23'
⊣ Fri, Apr  21st 2000: +12h02': +046d28'
⊣ Fri, Apr  21st 2000: +12h03': +046d32'
⊣ Fri, Apr  21st 2000: +12h04': +046d37'
⊣ Fri, Apr  21st 2000: +12h05': +046d42'
```

See Section G.2.2.5 [Sun data], page 155, and Section G.2.2.6 [Moon data], page 176, for further information.

'`--cycle-step=`*argument*'

Changes the timestep value that is used between the starting and ending clocktime of the time interval, in which the Sun and Moon oriented special texts are calculated, that are dynamical, i.e. depending on the respective clocktime. See [Fixed date option '`--cycle-end=`*argument*'], page 38, how to define such a time interval. Without the '`--cycle-end`' option, a '`--cycle-step`' option is completely ignored. It also does not enable Gcal's cycle mode. The *argument* of the '`--cycle-step`' option is by change syntactically according to the argument which may be given to the '`--time-offset`' option, and that relative time offset value references based on the actual local time and Universal time cannot be made, respectively. Gcal always uses the amount of the given timestep value. Thus, the template for the argument that may be given to the '`--cycle-step`' option is '[`+`|`-`]*mmmm*|*hh*:[*mm*]'. See [Calendar option '`--time-offset=`*argument*'], page 31, for the detailed explanation of the preceding option argument.

The timestep value between the starting and the ending time of the time interval that is respected by Gcal is one minute by default.

The following example shows how the topocentric elevation angle of the Sun for the location Münster, Germany, is calculated during two days in April 2000, and that for the time interval 10:00 o'clock until 17:00 o'clock local time with a timestep value of one hour and 15 minutes in each case:

```
$ gcal %20000420 -Hno -f/dev/null -xdt -g::: \
>       --here='0*d1#999_%ot+00+000_:_%o*a+5158+00738,120' \
>       --cycle-end=10:00 --time=17:00 --cycle-step=1:15
⊣
⊣ Thu, Apr  20th 2000: +10h45': +38.038
⊣ Thu, Apr  20th 2000: +12h00': +45.962
⊣ Thu, Apr  20th 2000: +13h15': +49.665
⊣ Thu, Apr  20th 2000: +14h30': +47.882
⊣ Thu, Apr  20th 2000: +15h45': +41.267
⊣ Thu, Apr  20th 2000: +17h00': +31.633
⊣ :::
⊣ Fri, Apr  21st 2000: +10h45': +38.355
⊣ Fri, Apr  21st 2000: +12h00': +46.299
⊣ Fri, Apr  21st 2000: +13h15': +50.006
⊣ Fri, Apr  21st 2000: +14h30': +48.198
⊣ Fri, Apr  21st 2000: +15h45': +41.545
⊣ Fri, Apr  21st 2000: +17h00': +31.881
```

See Section G.2.2.5 [Sun data], page 155, and Section G.2.2.6 [Moon data], page 176, for further information.

'`--precise`'

The astronomical data and times that are created by the Sun and Moon oriented special texts, and the geographical data that is created by the distance and course angle oriented special text, respectively, are displayed by using a higher precision, i.e. according to the mode or style of representation and the type of special text, decimal values are displayed with more decimal places, or output of the time seconds part or the arcseconds part.

See Section G.2.2.5 [Sun data], page 155, and Section G.2.2.6 [Moon data], page 176, likewise Section G.2.2.4 [Geographical distance and course angle], page 152, for further information.

'`--execute-command`'

Causes on the one hand, that shell command `%![`*`argument`*`]` special texts are executed, and not displayed textually only. See Section G.3 [Shell command `%![`*`argument`*`]` special text], page 195, for more details. And causes on the other hand, that '*`tvar`*`?[`*`command`*`]`' and '*`tvar`*`:[`*`command`*`]`' text variable definitions is assigned the output of an external command, and not the text which contains the call of the external command. See Section 5.1.9 [Text variables], page 100, for further information.

'`-D` *`argument`*'
'`--filter-day=`*`argument`*'

Displays only those fixed dates, whose date is not excluded by the given *argument*. See Section 5.1.2 [Date part of a line], page 81, and likewise Section G.1.2 [Exclusions without any argument `%?` special texts], page 140. The *argument* consists of one or more characters as used in the exclusion without any argument `%?` special text; but without the leading '`%`' character in each case.

For example, it is possible to induce Gcal to display only those fixed dates from the resource file '`demo.rc`' for the whole year which are legals holidays, and moreover, Saturdays or Sundays:

```
gcal -f demo.rc --year --filter-day=Rv
```

'-P *argument*'
'--filter-period=*argument*'

Displays only those fixed dates, whose date is not excluded by the given *argument*. See Section 5.1.2 [Date part of a line], page 81, and likewise Section G.1.1 [Exclusions with date argument %?[*date*] special texts], page 138. The *argument* consists of one or more expressions as used in the exclusion with date argument %?[*date*] special text; but without the leading '%' character in each case. You can use more than one of these %?[*date*] expressions by connecting them with a ',' character.

For example, it is possible to induce Gcal to display only those fixed dates from the resource file 'demo.rc' for the whole year which appear within the period of the 17th and the 20th of any month, and which appear on the 31st of any month.

```
gcal -f demo.rc -y -P i00000017#00000020,i00000031#00000031
```

'-I *pattern*'
'--filter-text=*pattern*'

Displays only those fixed dates, whose completely expanded text[11] is matched by the given *pattern*. See Section 5.1.4 [Text part of a line], page 91, and Section G.2 [%?... special texts for text replacement], page 141. The *pattern* is a *regular expression*, as recognized by the Unix ed line-editor. See Appendix E [Regular Expressions], page 129, for further information.

For example, it is possible to induce Gcal to display all fixed dates from the resource file 'doctor.rc' for the current month, which are defined for dentists or oculists:

```
gcal -f doctor.rc --month -I "[dD]entist|[oO]culist"
```

Each time a '_' (underscore) character is found in the *pattern* argument, this character is replaced by a real ' ' (space) character.

You may depreciate the special meaning of the '_' (underscore) character —in case this character itself is needed— by placing a '\' (backslash) character before it, e.g. '_'. If you need the '_' characters themselves, you have to protect the '\' (backslash) character by another '\' (backslash) character, e.g. '\\_'.

'--ignore-case'

Ignores all case distinctions in both the completely expanded text and the *pattern* of the '--filter-text=*pattern*' option. See [Fixed date option '--filter-text=*pattern*'], page 41.

'--revert-match'

Displays all those fixed dates, whose completely expanded text does not match the *pattern* of the '--filter-text=*pattern*' option. See [Fixed date option '--filter-text=*pattern*'], page 41.

'-c|C[-]'
'--today'
'--list-of-fixed-dates[=short|long]'
'--descending-fixed-dates[=short|long]'

Activate the fixed date function and display fixed date messages, if any.

'-c'
'--today'
'--list-of-fixed-dates=short'

Activate fixed date function (use standard resource file implicitly) and list all dates related to the actual system date (≡today), sorted in

[11] Except Gcal's line break-up characters '~' and '^', so you have to use '\~' and '\\^' in the *pattern* argument if you want to search texts which contain these characters.

ascending order. If no fixed dates related to the current day are found, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 1], page 239.

'`-c-`'
'`--descending-fixed-dates=short`'
Activate fixed date function (use standard resource file implicitly) and list all dates related to the actual system date (≡today), sorted in descending order. If no fixed dates related to the current day are found, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 1], page 239.

'`-C`'
'`--list-of-fixed-dates=long`'
Activate fixed date function (use standard resource file implicitly) and list all dates related to the actual system date (≡today), sorted in ascending order. If no fixed dates related to the current day are found, an "empty" fixed date message is displayed which is only consisting of the date.

'`-C-`'
'`--descending-fixed-dates=long`'
Activate fixed date function (use standard resource file implicitly) and list all dates related to the actual system date (≡today), sorted in descending order. If no fixed dates related to the current day are found, an "empty" fixed date message is displayed which is only consisting of the date.

'`-f|F` *name*[`+`...]'
'`--resource-file=`*name*[`+`...]'
Activate fixed date function and use file *name* instead of the standard resource file. Then list all dates, sorted in ascending order which occur in the fixed date period. If no other fixed date period is specified, the actual system date (≡today) is used for the fixed date period. If the option letter '`f`' of the short-style option is used and no dates related to the fixed date period are found in *name*, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 1], page 239. If the option letter '`F`' of the short-style option is used and no dates related to the fixed date period are found in *name*, an "empty" fixed date message is displayed which is only consisting of the date.

You can use more than one resource file *name* by connecting them with a '`+`' character, e.g.:

> '`--resource-file=./foo+xyz+/data/bar+$HOME/.gcalrc`' resp.,
> '`-f ./foo+xyz+/data/bar+$HOME/.gcalrc`'

respects all files given in the preceding argument by using a special file search mechanism. See [File searching mechanism], page 77, for more details. Use '`/dev/null`' to avoid the use of the standard resource file, useful for creating *empty fixed date lists*. If a single '`-`' character is given as file *name*, like '`--resource-file=./foo+-+xyz`' or '`-F-`', but not '`-F -`', Gcal reads and processes all input received from the *standard input channel*.

You may depreciate the special meaning of the '`+`' separator character —in case this character itself is needed— by placing a '`\`' (backslash) character before it, e.g. '`\+`'. If you need the '`\+`' characters themselves, you have to protect the '`\`' (backslash) character by another '`\`' (backslash) character, e.g. '`\\+`'.

Each time a ‘`_`’ (underscore) character is found in *name*, this character is replaced by a real ‘ ’ (space) character.

You may depreciate the special meaning of the ‘`_`’ (underscore) character —in case this character itself is needed— by placing a ‘`\`’ (backslash) character before it, e.g. ‘`\_`’. If you need the ‘`\_`’ characters themselves, you have to protect the ‘`\`’ (backslash) character by another ‘`\`’ (backslash) character, e.g. ‘`\\_`’.

‘`-# line`’
‘`--here=line`’

Activate fixed date function and use the *line* argument together with the standard resource file respectively additional resource files. The *line* argument has to be a valid line as it may occur in a Gcal resource file (see Section 5.1.1 [Structure of resource file], page 79), and is always evaluated last **after** the processing of all resource files, so dealing with references to exported local date or text variables is enabled. This option may be given multiple and it will be processed exactly in the given order. All dates are listed, sorted in ascending order which occur in the fixed date period. If no other fixed date period is specified, the actual system date (≡today) is used for the fixed date period. If no dates related to the fixed date period are found in the *line*, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 1], page 239.

For example, the call:

```
gcal -# "0*d1su#99su.7 ^%Z" --here="#include <foo>" -y
```

causes the implicit processing of the standard resource file just before the further resource file line ‘`0*d1su#99su.7 ^%Z`’ and following ‘`#include <foo>`’ are processed, and that as if these lines were a physical part of the standard resource file.

Each time a ‘`_`’ (underscore) character is found in *line*, this character is replaced by a real ‘ ’ (space) character.

You may depreciate the special meaning of the ‘`_`’ (underscore) character —in case this character itself is needed— by placing a ‘`\`’ (backslash) character before it, e.g. ‘`\_`’. If you need the ‘`\_`’ characters themselves, you have to protect the ‘`\`’ (backslash) character by another ‘`\`’ (backslash) character, e.g. ‘`\\_`’.

**Please note:**
If the short-style option ‘`-c`’ and/or its modifiers ‘`t|[w|m|y[+|-]`’, or the short-style option ‘`-f`’ is/are given in **upper-case**, e.g.:

```
-C
-F foo.bar
-CeT
-c-aW+
-CeaxY-
-c-M
```

Gcal displays fixed date messages for **every** day of the requested period.

The modifiers of a combined/complex/packed short-style fixed date option must be constructed according to following steps, because Gcal treats them as positional parameters:

First step: *Representation-of-text-modifier*
‘`a`’, ‘`A`’, ‘`e|E`’, ‘`J`’, ‘`k`’, ‘`o`’, ‘`Q`’, ‘`U`’, ‘`x`’, ‘`z`’, ‘`Z`’

Second step: *Period-of-fixed-dates-modifier*
‘`d`’, ‘`l`’, ‘`t|T`’, ‘`w|W|m|M|y|Y[+|-]`’, ‘*n*`+|-`’, ‘*n*`d`’, ‘*n*`w`’, ‘`*d`*n*[*www*]’, ‘`*w`*n*[*www*]’, ‘`@e|t|`*dvar*[[`+|-`]*n*]’, ‘`@e|t|`*dvar*[`+|-`]*n www*’, ‘*mmdd*’, ‘*mm wwwn*’

If you want to operate with the preceding modifiers, an **explicit** ‘`-c|C[-]`’ short-style option must be leading on the one hand, e.g.:

```
-c-t
-cw-
-CZdekloQUzJaxA12+
```

and only **one** period defining fixed date period modifier may be given[12].

Or on the other hand, the modifiers must be given as single character options or composed into a single command line word, and additionally either the short-style option ‘`-c|C[-]`’, ‘`-f|F` *name*[`+`. . .]’ or ‘`-#` *line*’[13] must be given like:

```
-c -t
-w- -c
-a -Ad -e -kloQUxz -C12+
-F foo.bar -dJZA -l*d10
--here="; Comment" -dZAa
```

If the modifiers are given as single character options or composed into a single command line word, and one of the ‘`t|T|[w|W|m|M|y|Y[+|-]]`’ modifiers is used[14], it is unnecessary to give a ‘`-c|C[-]`’ respectively ‘`-f|F` *name*[`+`. . .]’ short-style option[15] to activate the fixed date function of Gcal, because it is triggered implicitly by means of these modifiers / alias names.

See Appendix K [Coding Scheme], page 233, whose tables explain the relations between the *date part* (*yyyy*. . .) of a resource file line and the modifiers, the short-style option ‘`-c|C[]`’ —which is used for enabling the fixed date feature— respectively the long-style option ‘`--period-of-fixed-dates=`*argument*’ may have, i.e. they point out all possible correspondences. This is necessary because Gcal does not respect all entries found in a resource file if a *fixed date* argument is given, it respects only those entries which have a definite relation to the *fixed date* argument to avoid the output of redundant information!

[12] This means, exclusive the ‘`--include-today`’ option respectively ‘`d`’ modifier, the ‘`--list-mode`’ option or the ‘`l`’ modifier, and the ‘`--leap-day=february|march`’ option.

[13] Respectively their according long-style options.

[14] Or their according alias names, like ‘`--today`’, ‘`--tomorrow`’. . .

[15] Or their according long-style options.

**Fixed date text modifiers and options:**

'`--biorhythm-axis=`*`number`*'

The width of the biorhythm text graphics —which can be created by means of the biorhythm text graphics `%;`[*`date`*] special text— is changed to *number* characters for each axis of the bar. The *number* argument must be an integer value between 1...100, the default number of characters is 20. If the *number* divides 100 with a remainder, it is internally reduced to the nearest number that divides 100 without any remainder. This means, only the numbers 1, 2, 4, 5, 10, 20, 25, 50 and 100 are factually respected, so a *number* argument of 19 is internally reduced to 10.

For example:

```
$ cat bio.rc
⊣ 0*d1#999 %;19620921
$ gcal -fbio.rc %19961212 -s0 -xw --biorhythm-axis=19
⊣
⊣ Thu, Dec <12th>1996: 0- I       P1 E        +0
⊣ Fri, Dec  13th 1996: 0-  I   P   1          +0
⊣ Sat, Dec  14th 1996: 0-    @   E 0          +0
⊣ Sun, Dec  15th 1996: 0-  P  IE   0          +0
⊣ Mon, Dec  16th 1996: 0- P  E  I  0          +0
⊣ Tue, Dec  17th 1996: 1-P  E     I0          +0
⊣ Wed, Dec  18th 1996: 0- @        1          +0
```

See Section G.2.1.10 [Biorhythm `%?`[*`date`*] special texts], page 147, for more details.

'`--moonimage-lines=`*`number`*'

The height of the Moon phase text graphics —which can be created by means of the Moon phase text graphics `%Z`[*`date`*] special text— is changed to *number* lines. The *number* argument must be an integer value between 6...30, the default number of lines is 12.

For example:

```
$ gcal -f/dev/null -#0*d1#999_%Z__%O %19950407 -x --moon=8
⊣
⊣  Fri, Apr < 7th>1995:          (    @@@@@
⊣                           (           @@@@@@@@
⊣                        (              @@@@@@@@@@@
⊣                       (               @@@@@@@@@@@@
⊣                       (               @@@@@@@@@@@@
⊣                        (              @@@@@@@@@@@
⊣                         (             @@@@@@@@@@
⊣                             (        @@@@@@@ 42%+
```

See Section G.2.1.9 [Moon phase %[*format*]?[*date*] special texts], page 146, for more details.

'a'
'--include-resource-file-name'

Extend fixed date text by the name of the resource file and the line number which the fixed date text comes from, e.g.:

```
Mon, Jan   1st 1995: ('Eternal holiday list'#00003) New Year's day
Mon, Jan   1st 1995: (.gcalrc#00987) No fixed dates
...
```

'A'
'--alternative-format'

Instead of using the *standard* list format, e.g.:

```
Mon, Jan   1st 1995: New Year's day
Mon, Jan   1st 1995: No fixed dates
...
```

Gcal uses the *alternative* list format, e.g.:

```
Mon, Jan   1st 1995:
New Year's day
No fixed dates
...
```

'e'
'--include-holidays=long'

Enable inclusion of all built-in eternal holiday dates. This includes legal holidays and memorial days.

'E'
'--include-holidays=short'

Enable inclusion of those built-in eternal holiday dates which are legal holidays only.

'g[*text*]'
'--grouping-text[=*text*]'

Group fixed dates by day by using the *text* separator. If *text* is not given, the built-in *text*[1] is used for grouping, e.g.:

[1] RC_GROUP_SEP "" ≡ empty line only.

```
$ gcal -Cxw -cg
⊣
⊣ Sun, Nov   6th 1994: Text 1
⊣ Sun, Nov   6th 1994: Text 2
⊣
⊣ Mon, Nov   7th 1994: Text 3
⊣
⊣ Tue, Nov   8th 1994: Text 4
...


$ gcal -Cxw -g'-A_user\_defined-~-grouping text-'
⊣
⊣ Sun, Nov   6th 1994: Text 1
⊣ Sun, Nov   6th 1994: Text 2
⊣ -A user_defined-
⊣ -grouping text-
⊣ Mon, Nov   7th 1994: Text 3
⊣ -A user_defined-
⊣ -grouping text-
⊣ Tue, Nov   8th 1994: Text 4
...
```

The *text* may contain references to global date and text variables (see [Fixed date option ‘--date-variable=*argument*’], page 34, and [Fixed date option ‘--date-variable=*argument*’], page 35). Furthermore, all special texts which cause a text replacement may be used in the *text* (see Section G.2 [%?... special texts for text replacement], page 141).

Each time a ‘~’ (tilde) or ‘^’ (caret) character is found in *text*, this character is replaced by a real ‘\n’ (newline) character.

You may depreciate the special meaning of the ‘~’ (tilde) character —in case this character itself is needed— by placing a ‘\’ (backslash) character before it, e.g. ‘\~’. If you need the ‘\~’ characters themselves, you have to protect the ‘\’ (backslash) character by another ‘\’ (backslash) character, e.g. ‘\\_’. All above mentioned facts are also valid for the ‘^’ (caret) character.

Each time a ‘_’ (underscore) character is found in *text*, this character is replaced by a real ‘ ’ (space) character.

You may depreciate the special meaning of the ‘_’ (underscore) character —in case this character itself is needed— by placing a ‘\’ (backslash) character before it, e.g. ‘_’. If you need the ‘_’ characters themselves, you have to protect the ‘\’ (backslash) character by another ‘\’ (backslash) character, e.g. ‘\\_’.

‘k’
‘--include-week-number’

The leading date of a fixed date message is either extended by the ISO week number or the standard week number. See [Calendar option ‘--starting-day=*argument*’], page 30, [Calendar option ‘--iso-week-number=yes|no’], page 29, and Appendix C [Aspects in Internationalization], page 123, for further details.

‘o’
‘--omit-multiple-date-part’

Omit displaying the leading date of a fixed date message in case more than one message refers to that date[2], e.g.:

[2] Only if the ‘--alternative-format’ option or the ‘A’ modifier is not given.

```
$ gcal -cxo
⊣
⊣ Sun, Nov   6th 1994: Text 1
⊣                      Text 2
...
```

‘Q’
‘--suppress-fixed-dates-list-separator’

Suppress displaying of the blank line which is always leading a fixed date list. For example, to cause Gcal to print only the number of days between 1st January 1970 and 1st January 1980, you can call the program like this:

```
$ gcal -f /dev/null -QUx %19800101 -#0_%j-2440588
⊣ 3652
```

‘U’
‘--suppress-date-part’

Suppress displaying the leading date of a fixed date message, e.g.:

```
$ gcal -cxU
⊣
⊣ Text 1
⊣ Text 2
...
```

‘J’
‘--suppress-text-part’

Suppress displaying the text part of a fixed date message, e.g.:

```
$ gcal -cxJ
⊣
⊣ Sun, Nov   6th 1994:
⊣ Sun, Nov   6th 1994:
...
```

‘x’
‘--exclude-fixed-dates-list-title’

Suppress the title text line of the fixed date list.

‘--heading-text=*text*’

Changes the title text line of the fixed date list. Instead of displaying the default ‘`Fixed date list:`’ text, any other text can be used as heading text of the fixed date list, e.g.:

```
$ gcal -cUJ --heading-text=Hello,_it_is_%K_,~%t__o'clock
⊣
⊣ Hello, it is Sunday,
⊣ 14:32 o'clock
⊣
⊣ Sun, Nov   6th 1994:
...
```

The *text* may contain references to global date and text variables (see [Fixed date option '--date-variable=*argument*'], page 34, and [Fixed date option '--date-variable=*argument*'], page 35). Furthermore, all special texts which cause a text replacement may be used in the *text* (see Section G.2 [%?... special texts for text replacement], page 141).

Each time a '~' (tilde) or '^' (caret) character is found in *text*, this character is replaced by a real '\n' (newline) character.

You may depreciate the special meaning of the '~' (tilde) character —in case this character itself is needed— by placing a '\' (backslash) character before it, e.g. '\~'. If you need the '\~' characters themselves, you have to protect the '\' (backslash) character by another '\' (backslash) character, e.g. '\\_'. All above mentioned facts are also valid for the '^' (caret) character.

Each time a '_' (underscore) character is found in *text*, this character is replaced by a real ' ' (space) character.

You may depreciate the special meaning of the '_' (underscore) character —in case this character itself is needed— by placing a '\' (backslash) character before it, e.g. '_'. If you need the '_' characters themselves, you have to protect the '\' (backslash) character by another '\' (backslash) character, e.g. '\\_'.

'z'
'--include-consecutive-number'
Display consecutive numbers of fixed date messages.

'Z'
'--zero-dates-only'
Display only those dates, for which fixed dates do not exist, i.e. all "empty" fixed dates only consisting of the date.

**Fixed date period modifiers and options:**

'`--leap-day=february|march`'

The *leap day*[1] is observed in non-leap years.

'`--leap-day=february`'

A fixed date that appears in every year and which is related to the leap day, is displayed on the 28th February in non-leap years.

'`--leap-day=march`'

A fixed date that appears in every year and which is related to the leap day, is displayed on the 1st March in non-leap years.

'`d`'
'`--include-today`'

If lists of periods are generated, include the actual date into the list. See [Fixed date option '`--list-mode`'], page 50, how to create a list of periods.

'`l`'
'`--list-mode`'

Generate a list of periods instead of a single period.

**Please note:**
The following examples assumes the actual system date is the **17th February of any year** (00000217), weeks start on Mondays and ISO week numbers are used!

'`--period-of-fixed-dates=`*`argument`*'

'*`n`*`d`'
'`*d`*`n`*'

Single date of absolute day $n$ ≡ '`1...365|366|999`' of the actual year; the intensity level is the same as the simple '`-c`' option. If the value 999 for $n$ is given, the last day of the year (31st December) is assumed.

- E.g. '`gcal -c10d`' respectively '`gcal --period-of-fixed-dates=10d`' or '`gcal -c*d10`' respectively '`gcal --period-of-fixed-dates=*d10`' displays all fixed dates which occur on the 10th day of the year.

'`l`*`n`*`d`'
'`l*d`*`n`*'

List dates starting yesterday or tomorrow (depending on the given day number $n$) and ending on the $n$'th absolute day of the actual year; the intensity level is the same as the '`-cy`' option.

- E.g. '`gcal -cl10d`' respectively '`gcal -cl*d10`' displays a list of all fixed dates which start on the 10th day of year and end on the 16th February.

[1] The 29th February.

'*n*+|-' Single date of day *actual day* ± *n days* of the actual year; the intensity level is the same as the simple '-c' option.

- E.g. 'gcal -c10+' displays all fixed dates which occur 10 days after the 17th February (today).
- E.g. 'gcal -c10-' displays all fixed dates which occur 10 days before the 17th February (today).

'l*n*+|-' List dates starting yesterday '-' or tomorrow '+' and ending on *actual day* ± *n days* of the actual year; the intensity level is the same as the '-cy' option.

- E.g. 'gcal -cl10+' displays a list of all fixed dates which start on the 18th February and end 10 days after.
- E.g. 'gcal -cl10-' displays a list of all fixed dates which start 10 days before the 16th February and end on the 16th February.

'*n*w' Single dates of week number *n* ≡ '0|1...52|53|99' of the actual year; the intensity level is the same as the '-cw' option. See [Calendar option '--starting-day=*argument*'], page 30, [Calendar option '--iso-week-number=yes|no'], page 29, and Appendix C [Aspects in Internationalization], page 123, for more details.

- E.g. 'gcal -c0w' displays all fixed dates which occur in the 53rd week of the previous year, in case the previous year has a 53rd week. If the previous year has no 53rd week, all fixed dates occurring in the 1st week of the actual year are displayed.
- E.g. 'gcal -c6w' displays all fixed dates which occur in the 6th week of year.
- E.g. 'gcal -c52w' displays all fixed dates which occur in the 52nd week of year.
- E.g. 'gcal -c53w' displays all fixed dates which occur in the 53rd week of the actual year, in case the actual year has a 53rd week. If the actual year has no 53rd week, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. 'gcal -c99w' displays all fixed dates which occur in the last week of the actual year, i.e. either the 52nd or the 53rd week.

'l*n*w' List dates starting yesterday or tomorrow (depending on the given week number *n*) and ending on the first respectively last day of given week number *n* ≡ '0|1...52|53|99' of the actual year; the intensity level is the same as the '-cy' option. See [Calendar option '--starting-day=*argument*'], page 30, [Calendar option '--iso-week-number=yes|no'], page 29, and Appendix C [Aspects in Internationalization], page 123, for further details.

- E.g. 'gcal -cl12w' displays a list of all fixed dates which start on the 18th February and end on the last day of the 12th week.
- E.g. 'gcal -cl3w' displays a list of all fixed dates which start on the first day of the 3rd week and end on the 16th February.

'*mmdd*' Single date of day *dd* in month *mm* of the actual year; the intensity level is the same as the simple '-c' option.

- E.g. 'gcal -c0225' displays all fixed dates which occur on the 25th February.

'l*mmdd*' List dates starting yesterday or tomorrow (depending on the value given in *mmdd*) and ending on day *dd* of month *mm* of the actual year; the intensity level is the same as the '-cy' option.

- E.g. 'gcal -cl0225' displays a list of all fixed dates which start on the 18th February and end on the 25th February.

- E.g. ‘`gcal -cl0109`’ displays a list of all fixed dates which start on the 9th January and end on the 16th February.

‘*mmwwwn*’ Single date of *n*’th ≡ ‘`1...5|9`’ weekday *dd*|*www* in month *mm* of the actual year; the intensity level is the same as the simple ‘`-c`’ option.

- E.g. ‘`gcal -c01mon3`’ displays all fixed dates which occur on the 3rd Monday in January.
- E.g. ‘`gcal -c02fri9`’ displays all fixed dates which occur on the last Friday in February.

‘`l`*mmwwwn*’ List dates starting yesterday or tomorrow (depending on the value given in *mmwwwn*) and ending on *n*’th ≡ ‘`1...5|9`’ weekday *dd* of month *mm* of the actual year; the intensity level is the same as the ‘`-cy`’ option.

- E.g. ‘`gcal -cl01mon3`’ displays a list of all fixed dates which start on the 3rd Monday in January and end on the 16th February.
- E.g. ‘`gcal -cl02fri9`’ displays a list of all fixed dates which start on the 18th February and end on the last Friday in February.

‘`*d`*nwww*’ Single date of *n*’th ≡ ‘`1...51|52|53|99`’ weekday *www* of the actual year; the intensity level is the same as the simple ‘`-c`’ option.

- E.g. ‘`gcal -c*d16sun`’ displays all fixed dates which occur on the 16th Sunday of the actual year.
- E.g. ‘`gcal -c*d52sun`’ displays all fixed dates which occur on the 52nd Sunday of the actual year. If the actual year has no 52nd Sunday, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -c*d53sun`’ displays all fixed dates which occur on the 53rd Sunday of the actual year. If the actual year has no 53rd Sunday, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -c*d99sun`’ displays all fixed dates which occur on the last Sunday[1] of the actual year.

[1] That is either the 51st, 52nd or 53rd Sunday.

‘`l*dn`*www*’ List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on *n*'th ≡ ‘`1...51|52|53|99`’ weekday *www* of the actual year; the intensity level is the same as the ‘`-cy`’ option.

- E.g. ‘`gcal -cl*d16sun`’ displays a list of all fixed dates which start on the 18th February and end on the 16th Sunday of the actual year.
- E.g. ‘`gcal -cl*d52sun`’ displays a list of all fixed dates which start on the 18th February and end on the 52nd Sunday of the actual year. If the actual year has no 52nd Sunday, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -cl*d53sun`’ displays a list of all fixed dates which start on the 18th February and end on the 53nd Sunday of the actual year. If the actual year has no 53nd Sunday, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -cl*d99sun`’ displays a list of all fixed dates which start on the 18th February and end on the last Sunday[1] of the actual year.

‘`*wn`[*www*]’ Single date of weekday *www* of *n*'th ≡ ‘`0|1...52|53|99`’ week of the actual year; the intensity level is the same as the simple ‘`-c`’ option. If no weekday *www* is given, the starting day of the week is assumed for the weekday *www*. See [Calendar option ‘`--starting-day=`*argument*’], page 30, [Calendar option ‘`--iso-week-number=yes|no`’], page 29, and Appendix C [Aspects in Internationalization], page 123, for more details.

- E.g. ‘`gcal -c*w0sun`’ displays all fixed dates which occur on the Sunday of the 52nd respectively 53rd week of the previous year. In case such a Sunday does not exist, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -c*w16sun`’ displays all fixed dates which occur on the Sunday of the 16th week of the actual year.
- E.g. ‘`gcal -c*w52sun`’ displays all fixed dates which occur on the Sunday of the 52nd week of the actual year. If the actual year has no Sunday of the 52nd week, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -c*w53sun`’ displays all fixed dates which occur on the Sunday of the 53rd week of the actual year. If the actual year has no Sunday of the 53rd week, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. ‘`gcal -c*w99sun`’ displays all fixed dates which occur on the last Sunday[2] of the actual year.

‘`l*wn`[*www*]’

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on weekday *www* of *n*'th ≡ ‘`0|1...52|53|99`’ week of the actual year; the intensity level is the same as the ‘`-cy`’ option. If no weekday *www* is given, the starting day of the week is assumed for the weekday *www*. See [Calendar option ‘`--starting-day=`*argument*’], page 30, [Calendar option ‘`--iso-week-number=yes|no`’], page 29, and Appendix C [Aspects in Internationalization], page 123, for further details.

- E.g. ‘`gcal -cl*w0sun`’ displays a list of all fixed dates which start on the Sunday of the 52nd respectively 53rd week of the previous year and end on the 16th

[1] That is either the 51st, 52nd or 53rd Sunday.

[2] That is either the Sunday of the 51st, 52nd or 53rd week.

February. In case such a Sunday does not exist, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.

- E.g. '`gcal -cl*w16sun`' displays a list of all fixed dates which start on the 18th February and end on the Sunday of the 16th week of the actual year.
- E.g. '`gcal -cl*w52sun`' displays a list of all fixed dates which start on the 18th February and end on the Sunday of the 52nd week of the actual year. If the actual year has no Sunday of the 52nd week, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. '`gcal -cl*w53sun`' displays a list of all fixed dates which start on the 18th February and end on the Sunday of the 53rd week of the actual year. If the actual year has no Sunday of the 53rd week, no fixed date messages are displayed and the program is terminated with an error code. See [Error Code 126], page 239.
- E.g. '`gcal -cl*w99sun`' displays a list of all fixed dates which start on the 18th February and end on the last Sunday[3] of the actual year.

'`@e`[[`+|-`]*n*]'

Single date of day *n* relative to the Easter Sunday's date of the actual year; the intensity level is the same as the simple '`-c`' option.

- E.g. '`gcal -c@e`' displays all fixed dates which occur on the Easter Sunday's date.
- E.g. '`gcal -c@e10`' respectively '`gcal -c@e+10`' displays all fixed dates which occur 10 days after the Easter Sunday's date.
- E.g. '`gcal -c@e-10`' displays all fixed dates which occur 10 days before the Easter Sunday's date.

[3] That is either the Sunday of the 51st, 52nd or 53rd week.

'`l@e[[+|-]`*n*`]`'

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on *n*'th day relative to the Easter Sunday's date of the actual year; the intensity level is the same as the '`-cy`' option.

- E.g. '`gcal -cl@e`' displays a list of all fixed dates which start on the 18th February and end on the Easter Sunday's date.
- E.g. '`gcal -cl@e10`' respectively '`gcal -cl@e+10`' displays a list of all fixed dates which start on the 18th February and end 10 days after the Easter Sunday's date.
- E.g. '`gcal -cl@e-10`' displays a list of all fixed dates which start on the 18th February and end 10 days before the Easter Sunday's date.

'`@e[+|-]`*n www*'

Single date of *n*'th weekday *www* relative to the Easter Sunday's date of the actual year; the intensity level is the same as the simple '`-c`' option.

- E.g. '`gcal -c@e3fri`' respectively '`gcal -c@e+3fri`' displays all fixed dates which occur on the 3rd Friday after the Easter Sunday's date.
- E.g. '`gcal -c@e-3fri`' displays all fixed dates which occur on the 3rd Friday before the Easter Sunday's date.

'`l@e[+|-]`*n www*'

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on *n*'th weekday *www* relative to the Easter Sunday's date of the actual year; the intensity level is the same as the '`-cy`' option.

- E.g. '`gcal -cl@e3fri`' respectively '`gcal -cl@e+3fri`' displays a list of all fixed dates which start on the 18th February and end on the 3rd Friday after the Easter Sunday's date.
- E.g. '`gcal -cl@e-3fri`' displays a list of all fixed dates which start on the 18th February and end on the 3rd Friday before the Easter Sunday's date.

'`@t[[+|-]`*n*`]`'

Single date of day *n* relative to today's date of the actual year; the intensity level is the same as the simple '`-c`' option.

- E.g. '`gcal -c@t`' displays all fixed dates which occur on today's date (≡ '`gcal -c`').
- E.g. '`gcal -c@t10`' respectively '`gcal -c@t+10`' displays all fixed dates which occur 10 days after today's date (≡ '`gcal -c10+`').
- E.g. '`gcal -c@t-10`' displays all fixed dates which occur 10 days before today's date (≡ '`gcal -c10-`').

‘`l@t`[[`+`|`-`]*n*]’

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on *n*’th day relative to today’s date of the actual year; the intensity level is the same as the ‘`-cy`’ option.

- E.g. ‘`gcal -cl@t`’ displays nothing.
- E.g. ‘`gcal -cl@t10`’ respectively ‘`gcal -cl@t+10`’ displays a list of all fixed dates which start on the 18th February and end 10 days after today’s date (≡ ‘`gcal -cl10+`’).
- E.g. ‘`gcal -cl@t-10`’ displays a list of all fixed dates which start on the 18th February and end 10 days before today’s date (≡ ‘`gcal -cl10-`’).

‘`@t`[`+`|`-`]*n* *www*’

Single date of *n*’th weekday *www* relative to today’s date of the actual year; the intensity level is the same as the simple ‘`-c`’ option.

- E.g. ‘`gcal -c@t3fri`’ respectively ‘`gcal -c@t+3fri`’ displays all fixed dates which occur on the 3rd Friday after today’s date.
- E.g. ‘`gcal -c@t-3fri`’ displays all fixed dates which occur on the 3rd Friday before today’s date.

‘`l@t`[`+`|`-`]*n* *www*’

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on *n*’th weekday *www* relative to today’s date of the actual year; the intensity level is the same as the ‘`-cy`’ option.

- E.g. ‘`gcal -cl@t3fri`’ respectively ‘`gcal -cl@t+3fri`’ displays a list of all fixed dates which start on the 18th February and end on the 3rd Friday after today’s date.
- E.g. ‘`gcal -cl@t-3fri`’ displays a list of all fixed dates which start on the 18th February and end on the 3rd Friday before today’s date.

‘`@`*dvar*[[`+`|`-`]*n*]’

Single date of the day *n* relative to that date of the actual year, which is referenced by the date variable *dvar*; the intensity level is the same as the simple ‘`-c`’ option.

- E.g. ‘`gcal -v a=0422 -c@a`’ displays all fixed dates which occur on the 22nd April.
- E.g. ‘`gcal -v a=0422 -c@a10`’ respectively ‘`gcal -v a=0422 -c@a+10`’ displays all fixed dates which occur 10 days after the 22nd April.
- E.g. ‘`gcal -v a=0422 -c@a-10`’ displays all fixed dates which occur 10 days before the 22nd April.

'`l@`*dvar*[[`+`|`-`]*n*]'

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on the *n*'th day relative to that date of the actual year, which is referenced by the date variable *dvar*; the intensity level is the same as the '`-cy`' option.

- E.g. '`gcal -v a=0422 -cl@a`' displays a list of all fixed dates which start on the 18th February and end on the 22nd April.
- E.g. '`gcal -v a=0422 -cl@a10`' respectively '`gcal -v a=0422 -cl@a+10`' displays a list of all fixed dates which start on the 18th February and end 10 days after the 22nd April.
- E.g. '`gcal -v a=0422 -cl@a-10`' displays a list of all fixed dates which start on the 18th February and end 10 days before the 22nd April.

'`@`*dvar*[`+`|`-`]*n www*'

Single date of the *n*'th weekday *www* relative to that date of the actual year, which is referenced by the date variable *dvar*; the intensity level is the same as the simple '`-c`' option.

- E.g. '`gcal -v a=0122 -c@a2sat`' respectively '`gcal -v a=0122 -c@a+2sat`' displays all fixed dates which occur on the 2nd Saturday after the 22nd January.
- E.g. '`gcal -v a=0122 -c@a-2sat`' displays all fixed dates which occur on the 2nd Saturday before the 22nd January.

'`l@`*dvar*[`+`|`-`]*n www*'

List dates starting yesterday or tomorrow (depending on the value given in *n*) and ending on the *n*'th weekday *www* relative to that date of the actual year, which is referenced by the date variable *dvar*; the intensity level is the same as the '`-cy`' option.

- E.g. '`gcal -v a=0122 -cl@a2sat`' respectively '`gcal -v a=0122 -cl@a+2sat`' displays a list of all fixed dates which start on the 2nd Saturday after the 22nd January and end on the 16th February.
- E.g. '`gcal -v a=0122 -cl@a-2sat`' displays a list of all fixed dates which start on the 2nd Saturday before the 22nd January and end on the 16th February.

'`t|T`'
'`--tomorrow`' long-style option

List dates related to tomorrow.

- E.g. '`gcal -ct`' displays all fixed dates which occur on the 18th February (tomorrow).
- E.g. '`gcal -cdt`' displays all fixed dates which occur on the 17th February (today) and on the 18th February (tomorrow).

'w|W[+|-]'
'--week' long-style option
'--end-of-week' long-style option
'--start-of-week' long-style option

Display dates related to the week.
See [Calendar option '--starting-day=*argument*'], page 30.

- 'w' or '--week' long-style option
  List dates between the *starting day* of the current week and the last day of the current week
  - E.g. 'gcal -cw' displays all fixed dates which occur in the whole week the 17th February is part of.
- 'w+' or '--end-of-week' long-style option
  List dates between the day after the current day of the current week and the last day of the current week.
  - E.g. 'gcal -cw+' displays all fixed dates which start on the 18th February (tomorrow) and end on the last day of the week.
  - E.g. 'gcal -cdw+' displays all fixed dates which start on the 17th February (today) and end on the last day of the week.
- 'w-' or '--start-of-week' long-style option
  List dates between the *starting day* of the current week and the day before the current day of the current week
  - E.g. 'gcal -cw-' displays all fixed dates which start on the first day of the week and end on the 16th February (yesterday).
  - E.g. 'gcal -cdw-' displays all fixed dates which start on the first day of the week and end on the 17th February (today).

'm|M[+|-]'
'--month' long-style option
'--end-of-month' long-style option
'--start-of-month' long-style option

Display dates related to the month.

- 'm' or '--month' long-style option
  List dates between the first day of the current month and the last day of the current month.
  - E.g. 'gcal -cm' displays all fixed dates which occur in the whole month of February.
- 'm+' or '--end-of-month' long-style option
  List dates between the day after the current day of the current month and the last day of the current month.
  - E.g. 'gcal -cm+' displays all fixed dates which start on the 18th February (tomorrow) and end on the last day of the month February.
  - E.g. 'gcal -cdm+' displays all fixed dates which start on the 17th February (today) and end on the last day of the month February.
- 'm-' or '--start-of-month' long-style option
  List dates between the first day of the current month and the day before the current day of the current month.
  - E.g. 'gcal -cm-' displays all fixed dates which start on the first day of the month February and end on the 16th February (yesterday).

  - E.g. ‘`gcal -cdm-`’ displays all fixed dates which start on the first day of the month February and end on the 17th February (today).

‘`y|Y[+|-]`’<br>
‘`--year`’ long-style option<br>
‘`--end-of-year`’ long-style option<br>
‘`--start-of-year`’ long-style option

Display dates related to the year.

- ‘`y`’ or ‘`--year`’ long-style option
  List dates between the first day of the current year and the last day of the current year.
  - E.g. ‘`gcal -cy`’ displays all fixed dates which occur in the whole year.
- ‘`y+`’ or ‘`--end-of-year`’ long-style option
  List dates between the day after the current day of the current year and the last day of the current year.
  - E.g. ‘`gcal -cy+`’ displays all fixed dates which start on the 18th February (tomorrow) and end on the last day of the year.
  - E.g. ‘`gcal -cdy+`’ displays all fixed dates which start on the 17th February (today) and end on the last day of the year.
- ‘`y-`’ or ‘`--start-of-year`’ long-style option
  List dates between the first day of the current year and the day before the current day of the current year.
  - E.g. ‘`gcal -cy-`’ displays all fixed dates which start on the first day of the year and end on the 16th February (yesterday).
  - E.g. ‘`gcal -cdy-`’ displays all fixed dates which start on the first day of the year and end on the 17th February (today).

### 3.1.2 Response file

If a ‘`@file`’ command line argument is given, a *response file* by the name of *file* will be used, i.e. options and commands are preloaded from that file. A response *file* may contain options and commands —which are preloaded by Gcal—, but **no** references to further response files. If such references to further response files are found, they are ignored. See [Global option ‘`--response-file=name`’], page 10, for hints how to generate a response file automatically.

The searching scheme for a response file is the same as that for resource files, except **no** standard response file is respected. See [File searching mechanism], page 77, for more details. Multiple response files may be given in the command line when Gcal is started, and they are processed exactly in that order as given, i.e. strictly sequential.

Each option and command must be on a single line, i.e. separated by a real ‘`\n`’ (newline) character. A line beginning with a ‘`;`’ (semicolon) character in the response file is treated as a remark and will not be used by Gcal (see Section 5.1.6 [Comment line], page 95).

Options must be defined before commands. If a response file contains any commands, then all further arguments after the ‘`@file`’ option of the command line are ignored.

For example:

1. Contents of response file ‘`file`’:

```
Line      Text
1         -j
2         -Cw
EOF
```

```
$ gcal -i -b 3 @file -s 3 1994

↦ gcal -i -b 3 -j -Cw -s 3 1994
```

2. Contents of response file ‘`file`’:

```
Line      Text
1         -j
2         -Cw
3         may 1992
EOF
```

```
$ gcal -i -b 3 @file -s 3 1994

↦ gcal -i -b 3 -j -Cw may
```

3. Contents of response file ‘file’:

   ```
   Line      Text
   1         -j
   2         -Cw
   3         may
   4         1992
   EOF
   ```

   ```
   $ gcal -i -b 3 @file -s 3 1994

   ↦ gcal -i -b 3 -j -Cw may 1992
   ```

### 3.1.3 Actual date modifier

It is allowed to use that date —instead of the actual system date— which is defined by the '*%date*' command line argument. This means, fixed dates can be checked for **any** year and are respected in the same way, as if they would be fixed dates of the actual year.

The date must be denoted in one of these formats:

- *yyyy*[*mm*[*dd*|*wwwn*]]
- *yyyy**d|w*n*[*www*]
- *yyyy*@e|t|*dvar*[[+|-]*n*[*www*]]
- *month name*[*dd*]
- *weekday name*[*n*]
- *dd*

Some examples to this:

> **Please note:**
> The following examples assumes the actual system date is **Wednesday, the 17th February 1993** (19930217), weeks start on Mondays and ISO week numbers are used!

The *yyyy*[*mm*[*dd*|*wwwn*]] format:

- 'gcal %1994' (≡ *yyyy*)
  The actual date is set to the date 17th February 1994 ⇒ 19940217.
- 'gcal %1994Mar'
- 'gcal %19943'
- 'gcal %199403' (≡ *yyyymm*)
  The actual date is set to the date 17th March 1994 ⇒ 19940317.
- 'gcal %1994Mar3'
- 'gcal %1994033'
- 'gcal %19940303' (≡ *yyyymmdd*)
  The actual date is set to the date 3rd March 1994 ⇒ 19940303.
- 'gcal %1994MarWed3'
- 'gcal %199403wed3' (≡ *yyyymmwwwn*)
  The actual date is set to the date 3rd Wednesday in March 1994 (*n* ≡ '1...5|9') ⇒ 19940316.
- 'gcal %0'
- 'gcal %00'
- 'gcal %000'
- 'gcal %0000' (≡ *yyyy*)
  The actual date remains unchanged ⇒ 19930217.

- ‘gcal %0000Mar’
- ‘gcal %00003’
- ‘gcal %000003’ (≡ *yyyymm*)
  The actual date is set to the date 17th March 1993 ⇒ 19930317.
- ‘gcal %0000Mar3’
- ‘gcal %0000033’
- ‘gcal %00000303’ (≡ *yyyymmdd*)
  The actual date is set to the date 3rd March 1993 ⇒ 19930303.
- ‘gcal %0000MarWed3’
- ‘gcal %000003wed3’ (≡ *yyyymmwwwn*)
  The actual date is set to the date 3rd Wednesday in March 1993 (*n* ≡ ‘1...5|9’) ⇒ 19930317.

The *yyyy**d|w*n*[*www*] format:

- ‘gcal %1994*d10’ (≡ *yyyy**d*n*)
  The actual date is set to the date 10th absolute day in 1994 (*n* ≡ ‘1...365|366|999’) ⇒ 19940110.
- ‘gcal %1994*d10sun’ (≡ *yyyy**d*n* *www*)
  The actual date is set to the date 10th Sunday in 1994 (*n* ≡ ‘1...51|52|53|99’) ⇒ 19940306.
- ‘gcal %0*d10’ (≡ *yyyy**d*n*)
  The actual date is set to the date 10th absolute day in 1993 ⇒ 19930110.
- ‘gcal %0*d10sun’ (≡ *yyyy**d*n* *www*)
  The actual date is set to the date 10th Sunday in 1993 ⇒ 19930307.
- ‘gcal %1994*w10’ (≡ *yyyy**w*n*)
  The actual date is set to the date Monday of the 10th week in 1994 (*n* ≡ ‘0|1...52|53|99’) ⇒ 19940307.
- ‘gcal %1994*w10sun’ (≡ *yyyy**w*n* *www*)
  The actual date is set to the date Sunday of the 10th week in 1994 ⇒ 19940313.
- ‘gcal %0*w10’ (≡ *yyyy**w*n*)
  The actual date is set to the date Monday of the 10th week in 1993 ⇒ 19930308.
- ‘gcal %0*w10sun’ (≡ *yyyy**w*n* *www*)
  The actual date is set to the date Sunday of the 10th week in 1993 ⇒ 19930314.

The *yyyy*@e|t|*dvar*[[+|-]*n*[*www*]] format:

- ‘gcal %1994@e’ (≡ *yyyy*@e)
  The actual date is set to the date Easter Sunday 1994 ⇒ 19940403.

- ‘`gcal %1994@e10`’
- ‘`gcal %1994@e+10`’ (≡ *yyyy*`@e[+]`*n*)
  The actual date is set to the date 10 days after Easter Sunday 1994 ⇒ 19940413.
- ‘`gcal %1994@e-10`’ (≡ *yyyy*`@e-`*n*)
  The actual date is set to the date 10 days before Easter Sunday 1994 ⇒ 19940324.
- ‘`gcal %0@e`’
- ‘`gcal %00@e`’
- ‘`gcal %000@e`’
- ‘`gcal %0000@e`’ (≡ *yyyy*`@e`)
  The actual date is set to the date Easter Sunday 1993 ⇒ 19930411.
- ‘`gcal %0@e10`’
- ‘`gcal %0@e+10`’ (≡ *yyyy*`@e[|]`*n*)
  The actual date is set to the date 10 days after Easter Sunday 1993 ⇒ 19930421.
- ‘`gcal %0@e-10`’ (≡ *yyyy*`@e-`*n*)
  The actual date is set to the date 10 days before Easter Sunday 1993 ⇒ 19930401.
- ‘`gcal %1994@e3Wed`’
- ‘`gcal %1994@e+3Wed`’ (≡ *yyyy*`@e[+]`*n www*)
  The actual date is set to the date 3rd Wednesday after Easter Sunday 1994 ⇒ 19940420.
- ‘`gcal %1994@e-3wed`’ (≡ *yyyy*`@e-`*n www*)
  The actual date is set to the date 3rd Wednesday before Easter Sunday 1994 ⇒ 19940316.
- ‘`gcal %0@e3wed`’
- ‘`gcal %0@e+3wed`’ (≡ *yyyy*`@e[+]`*n www*)
  The actual date is set to the date 3rd Wednesday after Easter Sunday 1993 ⇒ 19930428.
- ‘`gcal %0@e-3we`’ (≡ *yyyy*`@e-`*n www*)
  The actual date is set to the date 3rd Wednesday before Easter Sunday 1993 ⇒ 19930324.
- ‘`gcal %1994@t`’ (≡ *yyyy*`@t`)
  The actual date is set to today’s date in 1994 ⇒ 199400217.
- ‘`gcal %1994@t10`’
- ‘`gcal %1994@t+10`’ (≡ *yyyy*`@t[+]`*n*)
  The actual date is set to the date 10 days after today’s date in 1994 ⇒ 19940227.
- ‘`gcal %1994@t-10`’ (≡ *yyyy*`@e-`*n*)
  The actual date is set to the date 10 days before today’s date in 1994 ⇒ 19940207.
- ‘`gcal %0@t`’
- ‘`gcal %00@t`’
- ‘`gcal %000@t`’
- ‘`gcal %0000@t`’ (≡ *yyyy*`@t`)
  The actual date is set to today’s date ⇒ 19930217.
- ‘`gcal %0@t10`’
- ‘`gcal %0@t+10`’ (≡ *yyyy*`@t[|]`*n*)
  The actual date is set to the date 10 days after today’s date ⇒ 19930227.

- ‘gcal %0@t-10’ (≡ *yyyy*@t-*n*)
  The actual date is set to the date 10 days before today’s date ⇒ 19930207.
- ‘gcal %1994@t3Wed’
- ‘gcal %1994@t+3Wed’ (≡ *yyyy*@t[+]*n www*)
  The actual date is set to the date 3rd Wednesday after today’s date in 1994 ⇒ 19940309.
- ‘gcal %1994@t-3wed’ (≡ *yyyy*@t-*n www*)
  The actual date is set to the date 3rd Wednesday before today’s date in 1994 ⇒ 19940302.
- ‘gcal %0@t3wed’
- ‘gcal %0@t+3wed’ (≡ *yyyy*@t[+]*n www*)
  The actual date is set to the date 3rd Wednesday after today’s date ⇒ 19930310.
- ‘gcal %0@t-3we’ (≡ *yyyy*@t-*n www*)
  The actual date is set to the date 3rd Wednesday before today’s date ⇒ 19930127.
- ‘gcal -v a=0303 %1994@a’ (≡ *yyyy*@*dvar*)
  The actual date is set to the date 3rd March 1994 ⇒ 19940303.
- ‘gcal -v a=0303 %1994@a10’
- ‘gcal -v a=0303 %1994@a+10’ (≡ *yyyy*@*dvar*[+]*n*)
  The actual date is set to the date 10 days after 3rd March 1994 ⇒ 19940313.
- ‘gcal -v a=0303 %1994@a-10’ (≡ *yyyy*@*dvar*-*n*)
  The actual date is set to the date 10 days before 3rd March 1994 ⇒ 19940221.
- ‘gcal -v a=0303 %0@a’ (≡ *yyyy*@*dvar*)
  The actual date is set to the date 3rd March 1993 ⇒ 19930303.
- ‘gcal -v a=0303 %0@a10’
- ‘gcal -v a=0303 %0@a+10’ (≡ *yyyy*@*dvar*[+]*n*)
  The actual date is set to the date 10 days after 3rd March 1993 ⇒ 19930313.
- ‘gcal -v a=0303 %0@a-10’ (≡ *yyyy*@*dvar*-*n*)
  The actual date is set to the date 10 days before 3rd March 1993 ⇒ 19930221.

- ‘`gcal -v a=0303 %1994@a3wed`’
- ‘`gcal -v a=0303 %1994@a+3wed`’ (≡ *yyyy*`@`*dvar*[`+`]*n www*)
  The actual date is set to the date 3rd Wednesday after 3rd March 1994 ⇒ 19940323.
- ‘`gcal -v a=0303 %1994@a-3wed`’ (≡ *yyyy*`@`*dvar*`-`*n www*)
  The actual date is set to the date 3rd Wednesday before 3rd March 1994 ⇒ 19940216.
- ‘`gcal -v a=0303 %0@a3wed`’
- ‘`gcal -v a=0303 %0@a+3wed`’ (≡ *yyyy*`@`*dvar*[`+`]*n www*)
  The actual date is set to the date 3rd Wednesday after 3rd March 1993 ⇒ 19930324.
- ‘`gcal -v a=0303 %0@a-3wed`’ (≡ *yyyy*`@`*dvar*`-`*n www*)
  The actual date is set to the date 3rd Wednesday before 3rd March 1993 ⇒ 19930210.

The *month name*[*dd*] format:

- ‘`gcal %April`’ (≡ *month name*)
  The actual date is set to the date 17th April 1993 ⇒ 19930417.
- ‘`gcal %april3`’
- ‘`gcal %april03`’ (≡ *month name dd*)
  The actual date is set to the date 3rd April 1993 ⇒ 19930403.

The *weekday name*[*n*] format:

- ‘`gcal %Friday`’ (≡ *weekday name*)
  The actual date is set to the date Friday of current week in February 1993 ⇒ 19930219.
- ‘`gcal %Friday2`’
- ‘`gcal %Friday02`’ (≡ *weekday name n*)
  The actual date is set to the date 2nd Friday in February 1993 (*n* ≡ ‘`1...5|9`’) ⇒ 19930212.

The *dd* format:

- ‘`gcal %3`’
- ‘`gcal %03`’ (≡ *dd*)
  The actual date is set to the date 3rd February 1993 ⇒ 19930203.

### 3.1.4 Commands

The commands control the periods Gcal respects. They can be partitioned into four major classes, namely *single* commands, special *3-Month mode* commands, *lists* of commands and *ranges* of commands. Single commands and special 3-Month mode commands only create **one** calendar sheet, ranges of commands or list of commands create **more than one** calendar sheet per program run. If a list of commands is given, Gcal works sequentially on each single command given in the list, one by one. A range of commands is expanded first by Gcal and after expansion is done, Gcal works sequentially on each single command produced during the internal expansion step.

For understanding the notation used in the text below, *mm* is either a number, a month name or one of the special 3-Month mode commands '`.`', '`..`', '`.+`' and '`.-`'. No lists or ranges of months or years can be created in case a special 3-Month mode command is given.

The range of *mm* is valid from 1 to 12 or valid from January to December. Both notations may be mixed. Month names may be abbreviated up to their first three characters. *yyyy* is a number. The range of *yyyy* is valid from 1 to 9999. A range ('*mm*-*mm*' or '*yyyy*+*yyyy*' or ...) must consist of two elements. A list ('*mm*,...,*mm*' or '*yyyy*;...;*yyyy*' or ...) must contain two elements minimum and may have any number of elements.

Here is a list of all ranges of commands, lists of commands and other command separator characters:

- The '`-`' character is the *range of months* separator.
- The '`,`' character is the *list of months* separator.
- The '`+`' character is the *range of years* separator.
- The '`;`' character is the *list of years* separator.
- The '`/`' character is the *month of year* separator.
- The '`:`' character is the *fiscal year* separator.

#### 3.1.4.1 Single commands

'*NOTHING*' Current month of current year, called *single month mode* in the further context.

'*month name*'

Selected month of current year, e.g.:

```
gcal may
gcal OCTOBER
gcal ja
```

‘*yyyy*’ Selected year, called *single year mode* in the further context, e.g.:

```
gcal 1992
```

But there is an exception of this general rule. If the specified year number is less or equal twelve, Gcal assumes that a selected month of the current year is wanted. If you want to have a selected year less or equal twelve, call Gcal like this

```
gcal 1:6
```

or like this

```
gcal 6+6
```

to obtain the year AD 6.

‘`mm` *yyyy*’
‘`mm/`*yyyy*’ Single month of selected year, e.g.:

```
gcal 9 1992
gcal 9/1992
gcal Nov 1777
gcal Nov/1777
```

‘`:`’ Single fiscal year, starting on the *actual month* of the *actual year* and ending on month *actual month*-1 of the *actual year*+1, called *implicit fiscal year* in the further context, e.g.:

```
gcal :
```

‘`mm:`’ Single fiscal year, starting on month *mm* of the *actual year* and ending on month *mm*-1 of the *actual year*+1, called *explicit fiscal year* in the further context, e.g.:

```
gcal 6:
gcal feb:
gcal NOVEMBER:
```

‘`:`*yyyy*’ Single fiscal year, starting on the *actual month* of year *yyyy* and ending on month *actual month*-1 of year *yyyy*+1, also called *explicit fiscal year* in the further context, e.g.:

```
gcal :1999
```

‘`mm:`*yyyy*’ Single fiscal year, starting on month *mm* of year *yyyy* and ending at month *mm*-1 of year *yyyy*+1, called *explicit fiscal year* too in the further context, e.g.:

```
gcal 7:1999
gcal oct:1992
gcal February:1777
```

#### 3.1.4.2 3-Month mode commands

‘`.`’ Previous, actual and next month of the current year, e.g.:

```
gcal .
```

‘`.` *yyyy*’ Previous, actual and next month of selected year, e.g.:

```
gcal . 1992
```

‘`..`’ Actual quarter of the current year, e.g.:

```
gcal ..
```

‘`..` *yyyy*’ Actual quarter of selected year, e.g.:

```
gcal .. 1992
```

'`.+`' Actual and next two months of the current year, e.g.:

```
gcal .+
```

'`.+` *yyyy*' Actual and next two months of selected year, e.g.:

```
gcal .+ 1992
```

'`.-`' Actual and previous two months of the current year, e.g.:

```
gcal .-
```

'`.-` *yyyy*' Actual and previous two months of selected year, e.g.:

```
gcal .- 1992
```

### 3.1.4.3 Lists of commands

'*mm*`,...,`*mm*'

List of specified months of the current year, e.g.:

```
gcal 1,5,12
gcal 3,1,5,3
gcal june,9,jan
```

'*mm*`/`*yyyy*`,...,`*mm*`/`*yyyy*'

List of months in specified year, e.g.:

```
gcal 1/1992,5,12/2001
gcal june/1991,9/1801,jan
```

'*mm*`,...,`*mm yyyy*'

List of specified months of selected year, e.g.:

```
gcal 3,7 1999
gcal 1,dec,july 1777
```

'*yyyy*`;...;`*yyyy*'

List of specified years, e.g.:

```
gcal 1992;1777;1899
```

'*mm*`:`*yyyy*`;...;`*mm*`:`*yyyy*'

List of specified fiscal years, e.g.:

```
gcal 7:1999;8:1992;April:3
```

**Other valid lists:**

'*mm*`,`*mm*`/`*yyyy*`,...`'

Mixed list of months consisting of: '*mm mm*`/`*yyyy*'

```
gcal 6,8,3/1999,feb/3,january
```

'*mm*`:;`*mm*`:`*yyyy*`;...`'

Mixed list of fiscal years of the given / actual year consisting of: '*mm*`: :`*yyyy mm*`:`*yyyy yyyy*'

```
gcal 3:;1994;february:1999;:1777;JAN:
```

'*mm*`:;`*mm*`:;...`'

List of fiscal years of the actual year consisting of: '*mm*`:`'

```
gcal 3:;august:;DEC:
```

### 3.1.4.4 Ranges of commands

'*mm*`-`*mm*' Range of current months, e.g.:

```
gcal 3-7
gcal 11-8
gcal MAY-dec
gcal 12-aug
```

‘*mm/yyyy-mm/yyyy*’
Range of months of year, e.g.:

```
gcal 3/1991-july/1992
gcal dec/1994-3
gcal sep-dec/2000
```

‘*mm-mm yyyy*’
Range of specified months of selected year, e.g.:

```
gcal 3-7 1999
gcal nov-3 1777
gcal aug-dec 1992
```

‘*mm-mm yyyy+yyyy*’
Range of specified months of selected years, e.g.:

```
gcal 3-7 1999+2001
gcal nov-3 1777+1600
gcal aug-dec 1992+1994
```

‘*mm yyyy-yyyy*’
‘*mm/yyyy-yyyy*’
Range of specified month of selected years, e.g.:

```
gcal AUGUST 1494-1497
gcal 3/1993-1999
gcal nov/3-1777
```

‘*yyyy+yyyy*’
Range of specified years, e.g.:

```
gcal 1992+1994
```

‘*mm:yyyy+mm:yyyy*’
Range of specified fiscal years, e.g.:

```
gcal 7:1999+8:1992
gcal aug:1992+july:1999
```

## 3.2 The GCAL environment variable

The environment variable GCAL can hold a set of default options for Gcal (see [Environment Variable GCAL], page 237). These options are interpreted first at program start-up, and some can be overwritten by explicit command line arguments or values found in a '*@file*' response file. See Section 3.1 [Command line arguments], page 8, and Section 3.1.2 [Response file], page 60, for further information.

For example:

- for **sh**:

```
GCAL="-i @color -b 3"; export GCAL
```

- for **csh**:

```
setenv GCAL "-i @color -b 3"
```

- for **MS-DOS**:

```
set GCAL=-i @color -b 3
```

On Vax/VMS, the name of the environment variable is GCAL_OPT, to avoid a conflict with the symbol GCAL set for invocation of the program.

For example:

- for **VMS**:

```
define GCAL_OPT "-i @color -b 3"
```

# 4 Eternal Holidays

The *eternal holiday list* is created only for dates after AD 29. It is assumed that the Gregorian Reformation has occurred from 10th till 22nd March 1924 for all Christian-Orthodox holidays, which are inserted into the *eternal holiday list*.

The *eternal holiday list* can be displayed in different ways:

If you start Gcal without an explicit (fiscal) date and the *eternal holiday list* argument (see [Calendar option '`--holiday-list[=long|short]`'], page 14), e.g.:

```
gcal -qfr --holiday-list
gcal -qfr -n
gcal -qfr -n :
```

Gcal displays all holidays of the eternal holiday list that refer to the actual / fiscal year, without a leading calendar sheet.

In case you start Gcal with the *eternal holiday list* argument and a month (plus an additional year), e.g.:

```
gcal -qbe --holiday-list july
gcal -qbe -n july
gcal -qbe -n 7 1993
gcal -qbe -n 7/1993
```

Gcal displays a month calendar sheet of the specified month (of year), and trailing those holidays of the eternal holiday list that refer to the given month (of the given year).

If you start Gcal with the *eternal holiday list* argument and a simple year (or an explicit fiscal year), e.g.:

```
gcal -qbe --holiday-list 1993
gcal -qbe -n 1993
gcal -qbe -n july:
gcal -qbe -n :1993
gcal -qbe -n 7:1993
```

Gcal displays a year calendar sheet of the specified year or fiscal year, and trailing all holidays of the eternal holiday list that refer to the given year or fiscal year.

In case you start Gcal with the *eternal holiday list* argument and a special 3-Month mode command (see Section 3.1.4.2 [3-Month mode commands], page 68), e.g.:

```
gcal -qbe --holiday-list .
gcal -qbe -n .
gcal -qbe -n ..
gcal -qbe -n .+
gcal -qbe -n .-
```

Gcal displays the according three months by using a fixed-style year calendar sheet, and trailing all holidays of the eternal holiday list that refer to these months.

In case you start Gcal with the *eternal holiday list* argument and a *list* or *range* of commands, it produces the according series of eternal holiday lists. See Section 3.1.4.3 [Lists of commands], page 69, and Section 3.1.4.4 [Ranges of commands], page 69, for further details.

Some annotations:

The following table lists all those characters which are used for marking an entry in the eternal holiday list, i.e. directly lead its date:

| **Character** | **Description** |
|---|---|
| + | Legal holiday which is valid in the whole country. Is automatically provided with highlighting sequences respectively marking characters. |
| # | Legal holiday which is valid in major parts of the whole country. Is automatically provided with highlighting sequences respectively marking characters. |
| * | Legal holiday which is valid in minor parts of the whole country. Is not provided with highlighting sequences respectively marking characters. |
| - | Other holiday which serves for memorial or remarking purposes only. Is not provided with highlighting sequences respectively marking characters. |

By using the '`--cc-holidays=`*cc*[`+`...]' option, it is possible to provide the eternal holiday list with additional country specific holidays. Thereafter, those country specific holidays are part of the list. See [Calendar option '`--cc-holidays=`*cc*[`+`...]'], page 16. For example:

```
--cc-holidays=be+fr
```

causes the inclusion of Belgian and French holidays into the eternal holiday list. The inclusion of country specific holidays is not coupled to Gcal's internationalization (see Appendix C [Aspects in Internationalization], page 123). This means, no country specific holidays are automatically respected at program start-up for a definite territory or country. So it is up to the user to decide which country specific holidays are included into the eternal holiday list. See Section 3.2 [The `GCAL` environment variable], page 71, for information how Gcal can be induced to include the country specific holidays automatically into the eternal holiday list at program start-up.

The free selection of the different country specific holidays can be used very ingenious in case information of a definite territory or country is required. Assuming a business man from Germany proposes to travel to Italy in November and needs the information which legal holidays are celebrated there on working days so he is able to plan his dates. This can be determined as follows:

```
gcal -f /dev/null -q IT -D r -cE nov
```

# 5 Fixed Dates

This chapter describes how to use the *fixed date* feature of Gcal. Normally, the fixed dates are stored line by line in a so-called resource file (see Section 5.1 [Resource file], page 77). On request, Gcal searches any of such resource files for those fixed dates, that are happening in the period for which a fixed date list is wanted. If Gcal has found any fixed dates that are happening in the requested period, the program displays them collectively as a fixed date list.

Thus, each time you execute Gcal in simple single month mode, i.e. no explicit month, year or fiscal year is given in the command line, but the *fixed date* argument, e.g.:

```
gcal --list-of-fixed-dates
gcal -c
```

it checks the resource file for dates and processes those that refer to the actual system date (≡today). Then Gcal displays all fixed dates found in the resource file which refer to the actual day (week/month/year) of the actual year, without a leading calendar sheet.

Some more examples:

```
gcal --ast -ce
gcal -cw-
gcal -c-m
gcal -c-y-
gcal -qbe -jcb -c-et
gcal --zod -cey-
gcal -qfr -jc -cey
```

The fixed date list can be embraced by different other leading and trailing output, respectively, that is also created by Gcal:

In case you start Gcal with the *fixed date* argument and/or a month and/or an explicit year or fiscal year; respectively with the special 3-Month mode command, e.g.:

```
gcal -c july
gcal -qfr -ce 7 1994
gcal -c-y 7/1994
gcal -c 7:1993
gcal -c 1993
gcal -c .
gcal -c .+
```

Gcal displays a (three) month / year calendar sheet of the specified respectively actual month / year, and trailing those fixed dates found in the resource file that refer to the given period; the intensity level is the same as the '`-cy`' option if the period refers to a year or fiscal year or to a three month period, respectively, is the same as the '`-cm`' option if the period refers to a month.

If you start Gcal without an explicit date and the *fixed date* argument **in connection with** the *eternal holiday list* argument, e.g.:

```
gcal -qbe -n -c
gcal -qbe -n -Cm
gcal -qbe -n -Cl1224
gcal -qbe -n -cl3+
gcal -qbe -n -4-
```

then Gcal displays all fixed dates found in the resource file that refer to the requested period, and after that all holidays of the eternal holiday list that refer to the whole actual year, without a leading calendar sheet. See Chapter 4 [Eternal Holidays], page 73, for more details how to use an *eternal holiday list*.

Annotation to previous examples:

Each time you execute Gcal in simple single year mode, i.e. no explicit month, year or fiscal year is given in the command line, but the *fixed date* argument (with optional modifiers) and the *number of blocks* argument (see [Calendar option '--blocks=*number*'], page 29), e.g.:

```
gcal -c --blocks=3
gcal -C-em -b 3 --cc-holidays=be
gcal -c-t -b 6 -jc
```

it checks the resource file for dates and processes those that refer to the actual system date (≡today). Gcal displays a leading year calendar sheet, and all fixed dates found in the resource file that refer to the actual day (week/month/year) of the actual year.

You can list fixed dates of past, present or future month/years, too. This can be done on the one hand, if you use the '%*date*' option in the way you require (see Section 3.1.3 [Actual date modifier], page 62), and on the other hand, if a command respectively a *list* or *range* of commands is used.

For example:

'`gcal -qbe -c -n- 1993`'
: Lists all fixed dates of entire 1993 in ascending order, and the eternal holiday list of entire 1993 in descending order; the intensity level is the same as the '`-cy`' option.

'`gcal -c- july:1993`'
: Lists all fixed dates of fiscal year —starting in July 1993 and ending in June 1994— in descending order; the intensity level is the same as the '`-cy`' option.

'`gcal -qbe -ce july 1993`'
: Lists all fixed dates of July 1993 inclusive all eternal holidays merged into this list in ascending order; the intensity level is the same as the '`-cm`' option.

'`gcal -qbe -c-e -n 1993+1996`'
: Lists all fixed dates of entire 1993, 1994, 1995 and 1996 (series of years) inclusive all eternal holidays merged into this list in descending order, and the eternal holiday list related to all above years in ascending order; the intensity level is the same as the '`-cy`' option.

'`gcal -cm %19920317`'
: Lists all fixed dates of March 1992 in ascending order.

'`gcal -qbe %1994@e -n -c`'
: Lists all fixed dates of Easter Sunday 1994 in ascending order, and the eternal holiday list of entire 1994 in ascending order.

## 5.1 Resource file

This section describes how to use a *resource file* that contains fixed dates and appointments. The name of the standard resource file is '`HOME/.gcalrc`'[1].

Gcal uses a special file searching mechanism:

1. The Gcal program scans the file directories which are referenced in the environment variable `GCALPATH` (see [Environment Variable `GCALPATH`], page 237) to find the standard resource file. The `GCALPATH` environment variable contains a ('`:`') colon-separated[2] list of file directories. If no such environment variable is defined and set, Gcal omits this step.
2. If the above action fails, Gcal looks for the standard resource file in the file directory which is referenced in the environment variable `HOME`. If no `HOME` environment variable is defined and set, Gcal also omits this step. See [Environment Variable `HOME`], page 237.
3. If the above action fails, Gcal inspects the *user data* file directory[3] to find the standard resource file. This file directory is a file directory relative to the `HOME` file directory. If an environment variable `GCAL_USR_DATADIR` is set, Gcal appends the contents of this environment variable to the contents of the `HOME` environment variable, and tries to use this file directory instead of using the burned-in default name of this file directory (see [Environment Variable `GCAL_USR_DATADIR`], page 237). If no `HOME` environment variable is defined and set, Gcal of course also omits this step.
4. If all above actions fail, Gcal inspects the *system data* file directory[4] to find the standard resource file. This file directory is a file directory absolute to the *root*/*$prefix* file directory. If an environment variable `GCAL_SYS_DATADIR` is set, Gcal tries to use this file directory instead of using the burned-in default name of this file directory (see [Environment Variable `GCAL_SYS_DATADIR`], page 237).

If the standard resource file is not found during the steps 1. . . 4 of the file searching mechanism, no fixed date messages will be created!

In case the '`--debug=abort`' option is given and no standard resource file is found during the steps 1. . . 4 of the file searching mechanism, the Gcal program will be aborted with an error code. See [Error Code 118], page 239, and [Global option '`--debug=abort`'], page 11, for more information.

If a '`-f|F` *`name`*[`+`. . .]' respectively '`--resource-file=`*`name`*' command line argument is given, a file *name* will be used for the resource file instead of the standard resource file. See [Fixed date option '`--resource-file=`*`name`*'], page 42.

[1] On MS-DOS, OS/2 and some other operating systems, the name of the *standard resource file* is '`HOME/gcalrc`' (without a leading '`.`').

[2] On MS-DOS, OS/2 and some other operating systems, another character will be used, e.g. '`;`' on MS-DOS and OS/2.

[3] Normally '`HOME/share/gcal`'.

[4] Normally '`/usr/local/share/gcal`' or '*`$prefix`*`/share/gcal`'.

Annotation:

An *extended file name* contains one ore more '`/`' (slash) characters[5] and denotes a file, whose fixed access path either starts from the root file directory, like '`/foo/bar/file`', or from the actual file directory, like '`./bar/file`' or '`../bar/file`'. If *name* is an extended file name:

- In case file *name* is found, Gcal will use it.
- In case file *name* is not found, Gcal will not use it.

A *simple file name* denotes a file, whose access-path either does not start from the root file directory or from the actual file directory, like '`file`' or '`bar/file`'. If *name* is a simple file name:

- Gcal tries to find this file *name* by using the previously explained file searching mechanism.

[5] Operating systems like MS-DOS or OS/2 use the '\' (Backslash) character for separating a file directory from a file name.

### 5.1.1 Structure of resource file

First of all, a Gcal resource file is a plain ASCII text file. This text file may be created by any text editor or by redirecting the *standard output channel* to a file, e.g.:

```
$ echo '19930217 Text'>> resource-file ⟨RET⟩
```

A special —but simple— line structure is required so Gcal is able to interpret its contents. Each fixed date entry in a resource file must be split into two parts, namely a *date part* and an optional *text part* which **must** be separated by **one** *whitespace*[1] character minimum. It is unnecessary to give a whitespace separator character if no *text part* is specified.

A line must always end with a '`\n`' (newline) character, except it is the last line of a resource file. The maximum length of a line is limited to `INT_MAX`[2] characters. A newline character is automatically appended to the line if the *standard output channel* is directed to a file. A newline character is appended to the typed line in a text editor window if it is completed by pressing the ⟨RET⟩ key. In case the used text editor does not generate the newline character in this way, it should be set to this mode of operation, otherwise this text editor is useless for creating Gcal resource files.

The line structure of fixed date entries is:

```
date part [ whitespace text part ] newline
```

or more concrete, e.g.:

```
yyyy[mm[dd|wwwn]] [ whitespace text ] newline
```

or much more concrete, e.g.:

```
19940217 Hi, I'm the text!
```

[1] For example *Tab*, *Space*, *Form-feed* and similar characters.

[2] Normally, this is for example equivalent to a value of `32,767` ≡ `(2^15)-1` on a 16-Bit machine architecture; to a value of `2,147,483,648` ≡ `(2^31)-1` on a 32-Bit machine architecture etc.

Besides fixed date entries, a resource file may contain further entries like:

Comments...

```
; A remarked line
;     A formatted and multi-line \
      remark
```

Include directives...

```
#include <file name>
#include "file name"
```

Date variable assignments respectively operations...

```
dvar=NOTHING
dvar=mmdd
dvar=mmwwwn
dvar=*dn[www]
dvar=*wn[www]
dvar=dvar[[+|-]n[www]]
dvar++
dvar--
dvar+=[+|-]n
dvar-=[+|-]n
dvar+=nwww
dvar-=nwww
```

Text variable assignments respectively operations...

```
tvar=[text]
tvar?[command]
tvar:[command]
tvar++
tvar--
tvar+=[+|-]n
tvar-=[+|-]n
```

Text variable references...

```
tvar
```

Text variable references at the beginning of a Gcal resource file line may only be used if it is ensured that they are expanded to a valid Gcal resource file line.

### 5.1.2 Date part of a line

The structure of a *date part* —which gives Gcal the information at which date a fixed date happens— of a line in the resource file is

**either**: *yyyy*[*mm*[*dd*|*www n*]]

*yyyy* (4 digits), is the year including the century (range `0000...9999`). Leading zeroes are required in case the defined year is less than `1000` and other components of the date part are trailing the year.

*mm* (2 digits or 3 characters), is the month (range `00...12` or `99`, respectively `Jan`, `Feb...`). A given `99` for the month means the last month of the year (≡ December). Leading zeroes are required in case the defined month is less than `10` and other components of the date part are trailing the month.

*dd* (2 digits), is the day (range `00...31` or `99`). A given `99` for the day means the last day of the month. Leading zeroes are required in case the defined day is less than `10` and other components of the date part are trailing the day.

*www* (2...3 characters), is a short weekday name (range `Mon...Sun`).

*n* (1 digit), is the *n*'th weekday *www* of month (range `1...5` or `9`).

- *n* ≡ `1...5`
  *n*'th weekday *www* of month.
- *n* ≡ `9`
  Last weekday *www* of month.

**or**: *yyyy*`*d|w`*n*[*www*]

*yyyy* (4 digits), is the year including the century (range `0000...9999`). Leading zeroes are required in case the defined year is less than `1000` and other components of the date part are trailing the year.

`*d` (2 characters), is the reference to an ordinary date.

`*w` (2 characters), is either the reference to a date of an ISO week or the reference to a date of a standard week.

*n* (1...3 digits), is the value in days or in weeks, the fixed date occurs. A given `99` connected with a short weekday name means the last weekday *www* of the year. A given `999` connected with `*d` means the last day of a year, i.e. the 31st December. If the computed date does not occur in the year, i.e. exceeds the year bounds, the fixed date message is suppressed.

*www* (2...3 characters), is a short weekday name (range `Mon...Sun`).

**or**: *yyyy*`@e|t|`*dvar*[[`+|-`]*n*]

*yyyy* (4 digits), is the year including the century (range `0000...9999`). Leading zeroes are required in case the defined year is less than `1000` and other components of the date part are trailing the year.

`@e` (2 characters), is the reference to the Easter Sunday's date.

`@t` (2 characters), is the reference to today's date.

`@`*dvar* (2 characters), is the reference to a date variable.

[`+`|`-`]*n* (1. . .4 alphanumeric characters), is the optional displacement value in days, the fixed date occurs relative to the Easter Sunday's date, relative to today's date, or relative to a date variables date. A given `-999` means the first day of a year, i.e. the 1st January. A given `+999` or `999` means the last day of a year, i.e. the 31st December. If the computed date does not occur in the year, i.e. exceeds the year bounds, the fixed date message is suppressed.

**or**: *yyyy*`@e`|`t`|*dvar*[`+`|`-`]*n* *www*

*yyyy* (4 digits), is the year including the century (range `0000`. . .`9999`). Leading zeroes are required in case the defined year is less than `1000` and other components of the date part are trailing the year.

`@e` (2 characters), is the reference to the Easter Sunday's date.

`@t` (2 characters), is the reference to today's date.

`@`*dvar* (2 characters), is the reference to a date variable.

[`+`|`-`]*n* *www* (3. . .7 alphanumeric characters), is the optional displacement value in weekdays, the fixed date occurs relative to the Easter Sunday's date, relative to today's date, or relative to a date variables date. A given `-99` means the first weekday *www* of the year. A given `+99` or `99` means the last weekday *www* of the year. If the computed date does not occur in the year, i.e. exceeds the year bounds, the fixed date message is suppressed.

**Please note:**

- If *yyyy* is specified as `0000`, `000`, `00` or `0`:
  The month and day are assumed to be annual events and the *text part* will be displayed for any year.

- If *mm* is not defined or specified as `00` or `0`:
  The day is assumed to be a monthly event for the specified year and the *text part* will be displayed for any month.

- If *dd* is not defined or specified as `00` or `0`:
  Each day is assumed to be a daily event for the specified year and month and the *text part* will be displayed for any day.

- If *www* is specified as a short (2. . .3 characters) weekday name:
  The given weekday name is assumed to be a weekly event for the specified year and month and the *text part* will be displayed for any week.

**Hint:**
Preceding table ease misleading assumption that a fixed date, which shall be valid for any days of the year, can be defined with a date part only consisting of a `0`. But that is only correct in that the so-defined fixed date is respected in some few fixed date periods only (see Appendix K [Coding Scheme], page 233).

If a fixed date is defined which shall be respected for any days of the year, it must be designed with a `0*d1#999` date part. For example, the same is likewise valid for fixed dates which represent a weekly event, and that shall be valid during all the year. Instead of creating such a fixed date with a `000000`*`www`* date part (that is likewise respected in some few fixed date periods only), it should be defined with a date part like `0*d1`*`www`*`#99`*`www`*`.7`. See Section 5.1.3.2 [Ranges of days], page 85, and Section 5.1.3.4 [Appearance factor of days], page 88, for further information.

### 5.1.3 Further date part attributes

The date part —which is leading a fixed date text of a Gcal resource file line— may be provided with further attributes, that are either specifications of lists of days or ranges of days, which provide the information at which date a fixed date happens. More further attributes are the repetition factor as well as the appearance factor.

All of those date parts in a Gcal resource file line, which are structured as follows, may generally be provided with further attributes:

- *yyyymm*. . .
- *yyyy*`*d|w`. . .
- *yyyy*`@e|t|`*dvar*. . .

**Attention:**
If the date part is supplied with further attributes and the year of the fixed date is not given in a concrete manner, i.e. the year *yyyy* is set to zero, such kinds of fixed dates are **only** computed correctly in simple year bounds —which means during a current year— and not in fixed dates shown after the current year has changed, e.g. in **fiscal years**.

#### 5.1.3.1 Lists of days

A list of days is used to define recurrent fixed dates and to use only one line in the resource file for them, instead of using several lines in a resource file needed for defining these recurrent fixed dates separately.

A list of days is specified by a '`,`' separator character; it must contain two elements minimum and may have any number of elements. A single element of the list may not be set to a zero value and be not concrete therefore. (This would not make any sense in this context.)

Assuming a fixed date shall always occur on the 1st and the 15th day in every month of the year 1996, e.g. '`Tax returns`', one solution would be on the one hand a fixed date entry in the resource file for the 1st of the month and another entry for the 15th of the month, which would be a total of two entries in the resource file for such a fixed date.

On the other hand, this expense can be reduced to a total of only one entry in the resource file by using a list of days, which is likewise valid for the 1st and the 15th day in every month, that is

```
    19960001,15 Tax returns
```

The use of lists of days in the date part is permitted as follows:

- *yyyymmdd*|*www*|*wwwn*,[*mm*]*dd*|[*mm*]*www*|[*mm*]*wwwn* . . .
- *yyyy*`*d|w`*n*|*n www*,*n*|*n www* . . .
- *yyyy*`@e|t|`*dvar*,|[+|-]*n*[*www*],[+|-]*n*[*www*] . . .

Some examples to this:

`000001fr3,12,99,mon,apr14,993,julfri3,08fr`
: In every year in January: on the 3rd Friday, on the 12th, on the last day of the month and on every Monday. And in every year: on the 14th April, on the 3rd December, on the 3rd Friday in July and on every Friday in August.

`199600mon,fr,12`
: In the year 1996 in every month: on every Monday, on every Friday and on the 12th.

`0*d1,1fr,999,17mo`
: In every year: on the first day of the year, on the 1st Friday of the year, on the last day of the year and on the 17th Monday of the year.

`1996*w1fr,1,17mo,99fr`
: In the year 1996: on Friday of the 1st week, on Monday of the first week, on Monday of the 17th week and on Friday of the last week.

`0@a,+1,-3,5,+4fr,2fr,-10sa`
: In every year: on the date of the date variable *a*, one day after the date of *a*, three days before the date of *a*, five days after the date of *a*, four Fridays after the date of *a*, two Fridays after the date of *a* and ten Saturdays before the date of *a*.

`1996@e+1,-3,5,+4fr,2fr,-10sa`
: In the year 1996: one day after the Easter Sunday's date, three days before the Easter Sunday's date, five days after the Easter Sunday's date, four Fridays after the Easter Sunday's date, two Fridays after the Easter Sunday's date and ten Saturdays before the Easter Sunday's date.

`1996@t+1,-3,5,+4fr,2fr,-10sa`
: In the year 1996: one day after today's date, three days before today's date, five days after today's date, four Fridays after the today's date, two Fridays after today's date and ten Saturdays before today's date.

### 5.1.3.2 Ranges of days

A range of days just as a list of days is used to define recurrent fixed dates and to use only one line in the resource file for them, instead of using several lines in a resource file needed for defining these recurrent fixed dates separately.

A range of days is specified by a '`#`' separator character and must consist of two elements, namely the starting day and the final day. The starting day and likewise the final day of the range may not be set to a zero value and be not concrete therefore. (This would not make any sense in this context.)

Assuming a fixed date shall always occur during the 1st and the 15th day (inclusive) in every month of the year 1996, e.g. '`Inside work`', one solution would be on the one hand a fixed date entry in the resource file for the 1st of the month, another one for the 2nd of the month until the 15th of the month, which would be a total of 15 entries in the resource file for such a fixed date[1].

On the other hand, this expense can be reduced to a total of only one entry in the resource file by using a range of days, which is likewise valid for the 1st until the 15th day in every month, that is

```
19960001#15 Inside work
```

[1] Apart from the possibility, that a list of days can be used in such a case.

The use of ranges of days in the date part is permitted as follows:

- *yyyymm* *www*#*www*
- *yyyymm* *dd*|*wwwn*#[*mm*]*dd*|[*mm*]*wwwn*
- *yyyy**d|*wn*[*www*]#*n*[*www*]
- *yyyy*@e|t|*dvar*[[+|-]*n*[*www*]]#[+|-]*n*[*www*]

Preceding table shows that the starting and ending day can only be specified by using date formats of the same type left and right the '#' separator character. Thus, it is not explicitly possible to define a range of days by using different types of date formats, means, fixed date entries like:

```
1996@e+3fr#1012 In the year 1996: every day that appears within the\
                period of the date 3 Fridays after Easter Sunday's\
                date until the 12th October.
```

cannot be defined like this. But this problem can be solved by using an inclusive date period %i[*date*][#[*date*]] special text for defining the ending day of the range[1], and the fixed dates are simply produced until the last day of the year, e.g.:

```
1996@e+3fr#+999 In the year 1996: every day that appears within the\
                period of the date 3 Fridays after Easter Sunday's\
                date until the 12th October.%i0#19961012
```

See Section G.1.1.1 [Inclusive date period %i[*date*][#[*date*]] special text], page 138, for more details.

Some examples to this:

`199600mon#fr`
: In the year 1996 in every month: every day that appears within the day sequence 'mon, tue, wed, thu, fri'.

`000000fr#mon`
: In every year in every month: every day that appears within the day sequence 'fri, sat, sun, mon'.

`000001fr3#12`
: In every year in January: every day that appears within the period of the 3rd Friday of the month until the 12th of the month.

`00000112#fr3`
: In every year in January: every day that appears within the period of the 12th of the month until the 3rd Friday of the month.

`00000112#augfri3`
: In every year: every day that appears within the period of the 12th January until the 3rd Friday of August.

`0000sep13#99fr9`
: In every year: every day that appears within the period of the 13th September until the last Friday of December.

`0*d1#1fr`
: In every year: every day that appears within the period of the 1st day of the year until the 1rd Friday of the year.

`0*d99fr#333`
: In every year: every day that appears within the period of the last Friday of the year until the 333rd day of the year.

[1] Implicit solution.

`1996*w1fr#17mo`
In the year 1996: every day that appears within the period of the Friday of the 1st week until the Monday of the 17th week.

`0@a#+4fr` In every year: every day that appears within the period of the date of the date variable *a* until the 4th Friday after the date of the date variable *a*.

`1996@e-3#+9fr`
In the year 1996: every day that appears within the period of the date three days before the Easter Sunday's date until the 9th Friday after the date of the Easter Sunday's date.

`1996@t-3#+99fr`
In the year 1996: every day that appears within the period of the date three days before today's date until the last Friday of the year.

#### 5.1.3.3 Repetition factor of days

A repetition factor of days is used to define recurrent fixed dates and to use only one line in the resource file for them, instead of using several lines in a resource file needed for defining these recurrent fixed dates separately.

A repetition factor of days ('`:n`') is specified by a '`:`' character and must trail the day field —which must have a concrete value in a date part of a Gcal resource file and has not been set to a zero value— respectively either lead or trail an appearance factor of days. Except ranges of days or fixed dates which occur only on a definite weekday like '`199600mon Every Monday 1996`', such a repetition factor may be specified in all possible codings of date parts of a Gcal resource file.

This factor may have values in range `1...999`. Values greater than the real difference between the date of the first occurrence of the fixed date and the last day of the year are always reduced to this real difference. In this sense, any value greater `365`[2] means a repetition factor until the last day of the year.

Assuming a fixed date shall always occur on the 15th day in every month of the year 1996 and covers a period of four days (inclusive the 15th itself), e.g. '`Co-workers training`', one solution would be on the one hand a fixed date entry in the resource file for the 15th of the month and the succeeding three days, which would be a total of four entries in the resource file for such a fixed date[3].

On the other hand, this expense can be reduced to a total of only one entry in the resource file by using a repetition factor of days, which is likewise valid for the 15th and the three days which succeed the 15th in every month, namely

```
19960015:4 Co-workers training
```

The use of repetition factors of days in the date part is permitted as follows:

- `yyyymmdd:n|wwwn:n`
- `yyyy*d|wn:n|nwww:n`
- `yyyy@e|t|dvar[[+|-]n[www]]:n`
- `yyyymmdd:n|wwwn:n,[mm]dd:n|[mm]wwwn:n ...`
- `yyyy*d|wn:n|nwww:n,n:n|nwww:n ...`
- `yyyy@e|t|dvar:n,|[+|-]n[www]:n,[+|-]n[www]:n ...`

Some examples to this:

[2] Or `366` in leap years.

[3] Apart from the possibility, that a list or a range of days can be used in such a case.

`000001fr3:11`
: In every year in January: every day that appears within the period of the 3rd Friday of the month and the succeeding ten days.

`00000112:3`
: In every year in January: every day that appears within the period of the 12th of the month and the succeeding two days.

`00000112:3,fr3:5`
: In every year in January: every day that appears within the period of the 12th of the month and the succeeding two days, and that appears within the period of the 3rd Friday of the month and the succeeding four days.

`0*d1:1`
: In every year: every day that appears within the period of the 1st day of the year and no succeeding day.

`0*d1:2`
: In every year: every day that appears within the period of the 1st day of the year and the succeeding day.

`0*d99fr:333`
: In every year: every day that appears within the period of the last Friday of the year and the succeeding 332 days. Well, in cases a fixed date exceeds the year bounds, it will only be produced until the last day of the year.

`1996*w1fr:17`
: In the year 1996: every day that appears within the period of the Friday of the 1st week and the succeeding 16 days.

`0@a:4`
: In every year: every day that appears within the period of the date of the date variable *a* and the succeeding three days.

`1996@e-3:9`
: In the year 1996: every day that appears within the period of the date three days before the Easter Sunday's date and the succeeding eight days.

`1996@t-3:9`
: In the year 1996: every day that appears within the period of the date three days before today's date and the succeeding eight days.

#### 5.1.3.4 Appearance factor of days

An appearance factor of days is used to define a concrete displacement of recurrent fixed dates.

An appearance factor of days (‘`.`*n*’) is specified by a ‘`.`’ character and must trail the day field —which must have a concrete value in a date part of a Gcal resource file and has not been set to a zero value— respectively either lead or trail a repetition factor of days. Except fixed dates which occur only on a definite weekday and are not specified by using a range of days, like ‘`199600mon Every Monday 1996`’, such an appearance factor may be specified in all possible codings of date parts of a Gcal resource file. This factor may be specified with each single element of lists of days, but in a range of days, this factor may trail **only** the final day of the range. Well, the use of such an appearance factor is only helpful if it is either given in a range of days, or if it is given together with a repetition factor.

This factor may have values in range `1...999`. Fixed dates will be ignored if the factor takes values greater than the real difference between the date of the occurrence of the fixed date and the last day of the year, respectively the end of the period, for which the fixed dates shall either be produced or respected.

Assuming a fixed date shall always occur on the 15th day in every month of the year 1996 and covers a period of seven days (inclusive the 15th itself), but shall only be respected every third day (i.e. two days have to be skipped at a time) within this period, e.g. ‘`Training-college`’, one solution would be on the one hand a fixed date entry in the resource file for the 15th of the month, for the 18th and for the 21st of the month, which would be a total of three entries in the resource file for such a fixed date[4].

On the other hand, this expense can be reduced to a total of only one entry in the resource file by using an appearance factor of days, which is likewise valid for the 15th, the 18th and the 21st in every month, namely on the one hand by the use of a repetition factor

```
19960015:7.3 Training-college
```

or on the other hand by the use of a range of days

```
19960015#21.3 Training-college
```

The use of appearance factors of days in the date part is permitted as follows:

- `yyyymmdd:n.n|dd.n:n`
- `yyyymmwwwn:n.n|wwwn.n:n`
- `yyyy*d|wn:n.n|n.n:n`
- `yyyy*d|wnwww:n.n|nwww.n:n`
- `yyyy@e|t|dvar[[+|-]n[www]]:n.n`
- `yyyy@e|t|dvar[[+|-]n[www]].n:n`
- `yyyymmdd:n.n|wwwn:n.n,[mm]dd:n.n|[mm]wwwn:n.n ...`
- `yyyymmdd.n:n|wwwn.n:n,[mm]dd:n.n|[mm]wwwn.n:n ...`
- `yyyy*d|wn:n.n|nwww:n.n,n:n.n|nwww:n.n ...`
- `yyyy*d|wn.n:n|nwww.n:n,n.n:n|nwww.n:n ...`
- `yyyy@e|t|dvar:n.n,|[+|-]n[www]:n.n,[+|-]n[www]:n.n ...`
- `yyyy@e|t|dvar.n:n,|[+|.]n[www].n:n,[+|.]n[www].n:n ...`
- `yyyymmwww#www.n`
- `yyyymmdd|wwwn#[mm]dd|[mm]wwwn.n`
- `yyyy*d|wn[www]#n[www].n`
- `yyyy@e|t|dvar[[+|-]n[www]]#[+|-]n[www].n`

Some examples to this:

`000001fr3:11.3`

In every year in January: every day that appears within the period of the 3rd Friday of the month and the succeeding ten days, but only every 3rd day within this period (skip two days at a time).

`00000112:3.2`

In every year in January: every day that appears within the period of the 12th of the month and the succeeding two days, but only every 2nd day within this period (skip one day at a time).

`00000112:3.2,fr3:5.3`

In every year in January: every day that appears within the period of the 12th of the month and the succeeding two days, but only every 2nd day within this period (skip one day at a time), and that appears within the period of the 3rd Friday of the month and the succeeding four days, but only every 3rd day within this period (skip two days at a time).

[4] Apart from the possibility, that a list of days can be used in such a case.

`0*d1:10.1`
: In every year: every day that appears within the period of the 1st day of the year and the succeeding nine days, and this for any day within this period (skip zero days at a time).

`0*d1:2.5`
: In every year: every day that appears within the period of the 1st day of the year and the succeeding day, but only every 5th day within this period (skip four days at a time). Well, the succeeding days (only one in this example) of the starting day (1st day of year) are not respected, because the next day resulted by the appearance factor exceeds the final day (resulted by the repetition factor) of the period.

`0*d99fr:333.61`
: In every year: every day that appears within the period of the last Friday of the year and the succeeding 332 days, but only every 61st day within this period (skip 60 days at a time). Well, in cases a fixed date exceeds the year bounds, it will only be produced until the last day of the year. No succeeding day (332 in this example) of the starting day (last Friday of the year) is respected by reason of the displacement value of 60 days, because the next day resulted by the appearance factor exceeds the final day (resulted by the repetition factor) of the period (the last day of the year).

`1996*w1fr:17.8`
: In the year 1996: every day that appears within the period of the Friday of the 1st week and the succeeding 16 days, but only every 8th day within this period (skip seven days at a time).

`0@a:4.3`
: In every year: every day that appears within the period of the date of the date variable *a* and the succeeding three days, but only every 3rd day within this period (skip two days at a time).

`1996@e-3:9.4`
: In the year 1996: every day that appears within the period of the date three days before the Easter Sunday's date and the succeeding eight days, but only every 4th day within this period (skip three days at a time).

`1996@e3#-20sun.15`
: In the year 1996: every day that appears within the period of the date three days after the Easter Sunday's date until the 20th Sunday before the Easter Sunday's date, but only every 15th day within this period (skip 14 days at a time).

`1996@t3#-20sun.15`
: In the year 1996: every day that appears within the period of the date three days after today's date until the 20th Sunday before today's date, but only every 15th day within this period (skip 14 days at a time).

### 5.1.4 Text part of a line

The *text part* of a line in a resource file can be any text you like. Indeed, some characters have a special meaning (‘%’, ‘$’, ‘~’, ‘^’ and ‘\’) and must be protected should the occasion arise that special character combinations are formed with them[1] which might be used only textually. If the text part contains characters that are used for highlighting the text or format it for a printer, or characters with decimal values above 127 in the code table of the used character set[2] not produced by Gcal itself, such characters respectively sequences are displayed by Gcal in an **uninterpreted** manner! For that very reason, it can happen that the output of such characters can potentially create problems with the used screen device driver software and/or the external pager program, likewise mailing of such texts by means of electronic mail.

So-called NUL characters[3] may also occur in the text part, but they only cause the suppression of all succeeding characters in the line. In my opinion, it makes no perceptible sense to output these NUL characters uninterpreted, so they are used for remarking purposes only; besides, the NUL characters would lead to the same problems as already mentioned above.

A line of the resource file is continued on the next line if a ‘\-\n’ (backslash-newline) character sequence is found, e.g.:

The line:

```
000000Mo Every Monday
```

and the lines:

```
000000Mo \
Every \
Monday
```

produce the same output and are essentially equivalent.

Furthermore, you can break-up the text of a *long text part* at any place you like. The term *long* means in this context, that the text displayed by Gcal would override the right text margin of the screen respectively break-up at that margin.

Each time a ‘~’ (tilde) character is found in the text part, this character is replaced by a real ‘\n’ (newline) character. Such texts will be displayed by Gcal in a formatted manner at a left margin, this means, they are lead by a definite number of space characters.

You may depreciate the special meaning of the ‘~’ (tilde) character —in case this character itself is needed— by placing a ‘\’ (backslash) character before it, e.g. ‘\~’. If you need the ‘\~’ characters themselves, you have to protect the ‘\’ (backslash) character by another ‘\’ (backslash) character, e.g. ‘\\~’.

Each time a ‘^’ (caret) character is found in the text part, this character is also replaced by a real ‘\n’ (newline) character. Such texts will be displayed by Gcal at column 1, this means, they are not lead by space characters. The rules for protecting the ‘^’ character are the same as the rules used for protecting the ‘~’ character.

[1] Like ‘%i’, ‘$a’. . .

[2] These are potentially non-printable.

[3] That is the character with the value zero in the code table of the used character set.

The resource file ‘lbrk-1.rc’

```
$ cat lbrk-1.rc
⊣ ; Hi, I’m ‘lbrk-1.rc’ and alive now
⊣ ;
⊣ 0 I know I’m a short text
⊣ 0 I hope I’m long enough~here, a line break-up~\
⊣ and again~and now for the last time...
⊣ 0 I hope I’m also long enough^here, a line break-up\
⊣ ~and again^and now for the last time...
⊣ 0 Am I another short text? Dunno...
```

is displayed as follows:

```
$ gcal %00000101 --resource-file=./lbrk-1.rc --disable-highlighting
⊣
⊣ Fixed date list:
⊣
⊣ Sat, Jan   1st 2000: Am I another short text? Dunno...
⊣ Sat, Jan   1st 2000: I hope I’m also long enough
⊣ here, a line break-up
⊣                      and again
⊣ and now for the last time...
⊣ Sat, Jan   1st 2000: I hope I’m long enough
⊣                      here, a line break-up
⊣                      and again
⊣                      and now for the last time...
⊣ Sat, Jan   1st 2000: I know I’m a short text
```

Because *whitespace* characters are used to separate the date part from the text part[4], it is not possible to supply the text part with leading *whitespace* characters without further ado. If one or more *whitespace* characters shall lead the text part, this can be arranged by protecting the **first** of these *whitespace* characters (and that by placing a ‘\’ (backslash) character before it), e.g. ‘\ ’ if it is a space character. By means of such a character sequence, Gcal notices that the text trailing the ‘\’ character is no longer used for separating purposes, but is member of the text part. Gcal skips this specific, marking backslash character (avoids its output) and processes the rest of the line as usual.

[4] These are internally skipped and remain undisplayed therefore.

The following example should be enough to elucidate the above facts. The resource file ‘wspc-1.rc’

```
$ cat wspc-1.rc
⊣ 0 BANG BANG
⊣ 0 bogus
⊣ 0         bogomips
⊣ 0    \hello world
⊣ 0    \ main(){for(;;)fork();}
⊣ 0 \     sh $0 & $0 &
⊣ 0    \  a  \  b  \\  c  \\\  d
⊣ 0    What happens now?\
⊣ ~0  \  This!
⊣ 0    What happens now?\\
⊣ ~0  \  That!
```

is displayed as follows:

```
$ gcal --resource-file=./wspc-1.rc --disable-highlighting
⊣
⊣ Fixed date list:
⊣
⊣ Wed, Jun  14th 2000:      sh $0 & $0 &
⊣ Wed, Jun  14th 2000:   a  \  b  \\  c  \\\  d
⊣ Wed, Jun  14th 2000:  main(){for(;;)fork();}
⊣ Wed, Jun  14th 2000: BANG BANG
⊣ Wed, Jun  14th 2000: What happens now?~0  \  That!
⊣ Wed, Jun  14th 2000: What happens now?
⊣                      0  \  This!
⊣ Wed, Jun  14th 2000: \hello world
⊣ Wed, Jun  14th 2000: bogomips
⊣ Wed, Jun  14th 2000: bogus
```

### 5.1.5 %?... Special Texts

The text part of a resource file line may be provided with different special texts which may occur in any[1] number and order. All these texts start with a '%' (percent) character which may be protected by a leading '\' (backslash) character if the '%' (percent) character itself is wanted to be used in the text part, e.g. '\%'. The '%' character is mostly trailed by an optional format instruction, (see Appendix J [Format Instruction], page 231, for the detailed description of the format instruction and its components), and then trailed by a distinguishing mark — which defines the kind of the special text to be used— and if it is a letter, Gcal accepts it **case-sensitive**.

The purpose of these special texts is mainly to suppress output of fixed dates in definite cases, either, or to provide them with particular texts respectively to start external commands. See Appendix G [Description of all %?... Special Texts], page 137, for a complete description of all usable special texts, and Appendix H [Summary of all %?... Special Texts], page 197, for an according short-list of them.

For example, the resource file 'speci-1.rc'

```
$ cat speci-1.rc
⊣ ; I'm veracious very special :)
⊣ ;
⊣ 0 1. Today is %>1u*K , %>02&*D  %U  %Y !
⊣ 0 2. It's the %>03&*N  day of the year.
⊣ 0 3. The actual week number is: %k .
⊣ 0 4. Currently, it's %t*  o'clock, Mr. %-USER .
⊣ 0 5. Hurry up with your work~\
⊣    by reason sunrise is at %o+5158+00738+61,2: .
```

is displayed as follows (in case today's date is the 4th October 1996):

```
$ gcal --resource-file=speci-1.rc -H no -ox
⊣
⊣ Mon, Oct   4th 1999: 1. Today is MONDAY, 04th October 1999!
⊣                      2. It's the 277th day of the year.
⊣                      3. The actual week number is: 40.
⊣                      4. Currently, it's 06:53pm o'clock, Mr. esken.
⊣                      5. Hurry up with your work
⊣                         by reason sunrise is at 07:33.
```

[1] Rational!

### 5.1.6 Comment line

A line[1] beginning with a ‘;’ (semicolon) character in the resource file is treated as a remark and will not be used by Gcal! Any *whitespace* characters may lead the remark character. See Section 5.1.4 [Text part of a line], page 91, for information how a NUL character can be used for remarking purposes in a text part.

The following example of the ‘remark.rc’ resource file —which **only** consists of remarks— should be enough to elucidate the above facts:

```
$ cat remark.rc
⊣ ;00001031 ^\
⊣                         .......^\
⊣                         ;::;::,^\
⊣                         ;::;::;,^\
⊣                        ;;::;;:;;,^\
⊣           .vmnv\%vnv\%,.;;;:::;;:;;,   .,vnmv\%vnn,^\
⊣        vmmmn\%vmmnv\%vnmmnv\%;;;;;;\%nmmmnv\%vnmv\%vmmv^\
⊣      vmmnv\%vmmmnv\%vnmmmmmnv\%;:\%nmmmmmmnv\%vnm\%vnmmmv^\
⊣     vmmnv\%vmmmnv\%vnmmmmmmmmnv\%nmmmmmmmmnv\%vnmm\%vnmmmv^\
⊣    vmmnv\%vmmmnv\%vnmmmmmmmmmnv\%vmmmmmmmmmmnv\%vnmv\%vnmmmv^\
⊣   vmmnv\%vmmmnv\%vnmm;mmmmmmnv\%vmmmmmmmm;mmnv\%vnmv\%vnmmmv,^\
⊣  vmmnv\%vmmmnv\%vnmm;' mmmmmnv\%vmmmmmmm;' mmnv\%vnmv\%vnmmmv,^\
⊣  vmmnv\%vmmmnv\%vn;;    mmmmnv\%vmmmmmm;;    nv\%vnmmv\%vnmmmv^\
⊣ vmmnv\%vmmmmnv\%v;;      mmmnv\%vmmmmm;;      v\%vnmmmv\%vnmmmv^\
⊣ vmmnv\%vmmmmnv\%vnmmmmmmmmm;;      mmmmmmmmmnv\%vnmmmmv\%vnmmmv^\
⊣ vmmnv\%vmmmmnv\%vnmmmmmmmmmm;;    mmmmmmmmmmnv\%vnmmmmv\%vnmmmv^\
⊣ vmmnv\%vmmmmnv\%vnmmmmmmmmmmnv;,mmmmmmmmmmmmnv\%vn;mmmv\%vnmmmv^\
⊣ vmmnv\%vmmm. nv\%vnmmmmmmmmmnv\%;mmmmmmmmmmmnv\%vn; mmmv\%vnmmmv^\
⊣ ‘vmmnv\%vmm,  v\%vnmmmmmmmmmmnv\%nmmmmmmmmmmnv\%v;  mmv\%vnnmmv’^\
⊣  vmmnv\%vmm;,   \%vnmmmmmmmmmnv\%nmmmmmmmmmnv\%;’   mv\%vnmmmmv^\
⊣   vmmnv\%vmm;;,   nmmm;’             mmmm;;’    mv\%vnmmmmv’^\
⊣   ‘vmmnv\%vmmm;;,.         mmnvv;,             mmv\%vnmmmmv’^\
⊣    ‘vmmnv\%vmmmmnv\%vnmmmmmmmmnvvnmmmmmmmnv\%vnmmmv\%vnmmmmv’^\
⊣      ‘vmvn\%vmmmmnv\%vnmmmmmmmnvvnmmmmmmnv\%vnmmmm\%vnmmmv’^\
⊣          ‘v\%mmmmmn%:\%vnmnmmmmn\%vnmmmnv%:\%vnmv\%nmv’
⊣
⊣ ; A remarked line
⊣       ; Also a remarked line
⊣ ;     A                          \
⊣       formatted and multi-line   \
⊣       remark
```

[1] Physically and logically.

### 5.1.7 Include directives

You may add `#include` directives in a resource file for searching and loading further resource files[1]. An `#include` directive tells Gcal to suspend reading the current resource file and read one other resource file before continuing. The directive is a line in the resource file that looks like this template:

`#include` *whitespace argument* [*whitespace*] *newline*

The *argument* can either be

"*file name*"

or[2]

<*file name*>

One *whitespace* character minimum is required between `#include` and the *argument*. The *argument* may be trailed by extra *whitespace* characters and the line must always end with a '`\n`' (newline) character, except it is the last line of a resource file.

Let us inspect a example which is a bit more concrete:

1. #include "foo/bar"
2. #include <bar>

The first `#include` directive tells Gcal to load the file '`bar`' in the file directory '`foo`' from the actual file directory. If this fails, Gcal tries to load this file by using steps 1. . . 4 of the previously explained mechanism used for searching files (see [File searching mechanism], page 77).

The second `#include` directive tells Gcal to load the file '`bar`' from the user respectively system data file directory by using steps 3. . . 4 of the previously explained mechanism used for searching files. It is not allowed to include files which have a fixed access path starting from the root file directory by such an `#include` directive like '`#include </file>`' or '`#include </foo/bar/file>`'; just as the specification of a single or an extended file name containing a disk/drive specifier[3], because the resulting path name[4] would not be a valid file name any longer.

Included files may include other files again. But the nesting level of included files is limited on the one hand by the amount of usable working storage of the computer, and on the other hand by the value which is given by the operating system respectively the compiler for the number of files which can be opened simultaneously. Recursive includes[5] or cyclic includes[6] are not permitted, because such backward references would produce an infinite loop in the program internally! Gcal recognizes such situations and terminates immediately with a fatal error. See [Error Code 119], page 239, for more information.

If an included resource file cannot be found and no '`--debug=abort`' option is given, processing of the resource file containing the `#include` continues.

[1] A mechanism similar the method which is used by the C Preprocessor.

[2] Thus, a file name which is embraced by '`""`' or '`<>`'.

[3] Even if operating systems like MS-DOS or OS/2 permits such a disk/drive specification on principle.

[4] A path name lead by the user respectively system data file directory name.

[5] The included file tries to include itself again.

[6] The included file is included again by another file that occurs at a deeper place in such a chain.

### 5.1.8 Date variables

Gcal respects *global* and/or *local* date variables which may be used either in the date part of a Gcal resource file line, or which may be part of a special text in the text part of a Gcal resource file line. This means, up to **24** user-defined date variables can be set and referenced during program execution. See [Fixed date option '`--date-variable=argument`'], page 34, for more information how global date variables can be used.

A date variable name *dvar* consists of a single, case-insensitive letter[1]. In a definition, this name is trailed by the assignment operator character '`=`', and the numerical values of month *mm* (or a short, three characters month name, e.g. '`Jan`', '`Feb`'. . . ) and day *dd*, this variable has to store[2], e.g.:

```
a=1127
c=a
```

stores the 27th November into the local date variable *a* and into the local date variable *c*. You should always remember that date variables can only be referenced in a line if they are already defined, because Gcal processes a resource file line by line.

Only local date variables can store dynamical dates given in the *mm www n*, *dvar*[+|-]*n*[*www*] or `*d|w`*n*[*www*] format, e.g.:

```
a=03su2
b=JunMon9
c=b-3
d=c+4Sa
x=*d10
y=*d10fri
z=*w3Su
```

which means, date variable *a* stores the date of second Sunday in March and *b* stores date of last Monday in June. The date variable *c* stores the date which is three days before the date of the last Monday in June and this date is set in the date variable *d* to that date which is four Saturdays after the date of *c*. The date variable *x* stores the date of the 10th absolute day of the year ($\equiv$ 10th January). *y* stores the date of the 10th Friday of the year, and the assignment to *z* is the date of the Sunday which occurs in the 3rd week of the year.

**Caution:**
These kinds of assignments to date variables are **only** computed correctly in simple year bounds —which means during a current year— and not in fixed dates shown after the current year has changed, e.g. in **fiscal years**.

Date variables which are defined in a resource file are always local and only respected in this specific file, and that file which is included by it. This means for the included file, that **all** local variables of the caller are visible. The included file itself may define its own local variables in its own name-space, which are not exported to caller on return. All other definitions of date variables[3] are always global. If a resource file line contains a reference to a date variable which is not locally defined in that file, Gcal will try to use the global values held by this variable. If this fails because no global definition of this date variable is given, Gcal will not process this line. If the simple '`--debug`' respectively '`--debug=internal`' option is given, informational messages

[1] Except the date variable *e* which is internally reserved for the Easter Sunday's date, so it cannot be assigned or operated therefore. And except the date variable *t* which is internally reserved for today's date, so it likewise cannot be assigned or operated.

[2] Or the name of another date variable, which is already defined.

[3] In the command line, environment variable `GCAL` or response file.

for each undefined date variable will be shown on the *standard error channel* (see [Global option '--debug=internal'], page 10). If the '--debug=abort' option is given, the Gcal program will be aborted with an error code when the first undefined global date variable reference occurs. See [Error Code 113], page 239, and [Global option '--debug=abort'], page 11, for further information.

A local date variable can be deleted. This means, that this date variable is no longer valid in its local scope and be undefined therefore. So it is possible to use its potentially defined global value in succeeding lines if this date variable is referenced there. The following example of the resource file 'dvar-1.rc' elucidates these facts:

```
$ cat dvar-1.rc
⊣ ; dvar-1.rc
⊣ ;
⊣ z=0202
⊣ 0@z The local assigned date to 'z'
⊣ z=
⊣ 0@z The global assigned date to 'z'
⊣ z=0404
⊣ 0@z The local assigned date to 'z'
```

So Gcal creates the following output:

```
$ gcal %1777 -H no -x -v z=1212 -f ./dvar-1.rc -y
⊣
⊣ Sun, Feb   2nd 1777: The local assigned date to 'z'
⊣ Fri, Apr   4th 1777: The local assigned date to 'z'
⊣ Fri, Dec  12th 1777: The global assigned date to 'z'
$ gcal %1777 -H no -x -f ./dvar-1.rc -y
⊣
⊣ Sun, Feb   2nd 1777: The local assigned date to 'z'
⊣ Fri, Apr   4th 1777: The local assigned date to 'z'
```

Only advanced users should apply the '`--export-date-variables`' option which causes that the actual incarnation of a local date variable —which was previously defined in a resource file and not in an included file— is being exported to further resource files instead of using its global value, in case that date variable is not locally defined in that further resource file. See Section 5.1.7 [Include directives], page 96, for more details.

But be aware, the use of this option could create unwanted results, because the order of processing the resource files is an important entity in managing the '`--export-date-variables`' option, so it is not recommended to use it. You, the user, must exactly know what you are doing when applying this option; **you are expressively warned now!**

Some basic operations can be performed on date variables. These are:

| **Operation** | **Description** |
|---|---|
| *dvar*`++` | Simple increment by one day. |
| *dvar*`--` | Simple decrement by one day. |
| *dvar*`+=`[`+`\|`-`]*n* | Addition of a constant numerical day factor [`+`\|`-`]*n*. |
| *dvar*`-=`[`+`\|`-`]*n* | Subtraction of a constant numerical day factor [`+`\|`-`]*n*. |
| *dvar*`+=`*n* *www* | Addition of *n* weekdays *www*. |
| *dvar*`-=`*n* *www* | Subtraction of *n* weekdays *www*. |

The scope of the operations which are done on a local date variable, is that resource or include file, where the local date variable is defined.

If operations on a global date variable are performed in a resource file or that file which is included by it, these operations are only valid in that specific file[4], not in further resource files processed. This means, the initial values of global date variables are always restored if the resource file changes.

Please finally note, that each date variable assignment/operation must be given separately on a single line in the resource file.

[4] They are only done locally.

### 5.1.9 Text variables

Gcal respects *global* and/or *local* text variables which may be used anywhere in a line of a Gcal resource file. This means, up to **26** user-defined text variables can be set and referenced during program execution. See [Fixed date option '`--text-variable=`*`argument`*'], page 35, for more information how global text variables can be used.

A text variable name *tvar* consists of the '`$`' prefix and a single, case-insensitive letter.

A text variable is defined as follows:

- If the name of a text variable is trailed by the assignment operator character '`=`', all the text which appears at the right side of this assignment operator is assigned to this text variable.
- If the name of a text variable is trailed by the assignment operator character '`?`', all the text which appears at the right side of this assignment operator is assumed as an external command and executed by Gcal. The output which is created on the *standard output channel* by the command run is assigned to this text variable; and that in an interpreted manner. This means, Gcal interprets all special texts and references to other text variables that are contained in this output. See Section 5.1.5 [Special Texts processed], page 94, for further information.
- If the name of a text variable is trailed by the assignment operator character '`:`', all the text which appears at the right side of this assignment operator is assumed as an external command and executed by Gcal. The output which is created on the *standard output channel* by the command run is assigned to this text variable; and that in an uninterpreted manner. This means, Gcal does not interpret any special texts and references to other text variables that are contained in this output. See also Section 5.1.5 [Special Texts processed], page 94, for further details.

Indeed, Gcal executes external commands only if the '`--execute-command`' option is given at program start-up. See [Fixed date option '`--execute-command`'], page 40, for more details.

The text which appears at the right side of one of the valid assignment operator characters may reference names of already defined text variables. These references are converted to the according textual values just before the assignment is performed by Gcal. You should always remember that text variables can only be referenced in a line if they are already defined, because Gcal processes a resource file line by line. Gcal always expands text variable references recursively until all sub-references to other text variables are resolved.

Local text variables are set to an *empty value* and thus defined in a special mode in case no text appears at the right side of one of the valid assignment operator characters.

If a text variable is referenced, an optional format instruction may be specified between the '`$`' prefix and the single letter (here: the variable name), which makes it possible to modify the representation of the text (here: the contents) at which the text variable points to. Let us assume a resource file by the name of '`tvarf-1.rc`' exists with the following contents:

```
$ cat tvarf-1.rc
⊣ ; tvarf-1.rc
⊣ ;
⊣ $a=123
⊣ $b=$:010*a
⊣ $c=$b
⊣ $b=
⊣ 0 1. \$c=.$c.
⊣ 0 2. \$c=.$:20*c.
⊣ 0 3. \$b=.$>1w*b.
⊣ 0 4. \$a=.$>5#a.
```

So Gcal creates the following output:

```
$ gcal -Ux --text-variable='$b=XXX YY' --resource-file=./tvarf-1.rc
⊣
⊣ 1. $c=.0000000123.
⊣ 2. $c=.      0000000123       .
⊣ 3. $b=.Xxx Yy.
⊣ 4. $a=.  123.
```

See Appendix J [Format Instruction], page 231, for the detailed description of the format instruction and its components.

You may depreciate the special meaning of the '$' prefix character —in case this character itself is needed in the text— by placing a '\' (backslash) character before it, e.g. '\$'. If you need the '\$' characters themselves in the text, you have to protect the '\' (backslash) character by another '\' (backslash) character, e.g. '\\$'.

External commands are not directly executed by Gcal, they are executed by means of the command line interpreter of the operating system, the shell. As consequence to this, of course the commands have to be specified according to the syntax conventions of the used shell, concerning the grouping conventions, list processing conventions, redirection conventions et cetera! For example, it is possible that you have to write '$a?(cat text)' instead of the simple '$a?cat text'. See the pertinent literature for more details.

If the output of external commands, which appears on the *standard output channel*, is assigned to text variables in an interpreted manner, this output is internally directed by Gcal into a temporary file. Thereafter, the contents of the temporary file is transfered to the text variable and the temporary file is removed.

If the output of external commands, which appears on the *standard output channel*, is assigned to text variables in an uninterpreted manner, this output is internally processed by the Txt2gcal program first (see Section B.2 [Invoking txt2gcal], page 119), and then this output is directed by Gcal into a temporary file. Thereafter, the contents of the temporary file is transfered to the text variable and the temporary file is removed. In case a TXT2GCALPROG environment variable (see [Environment Variable TXT2GCALPROG], page 238) is defined and set with the file name of the executable Txt2gcal program, Gcal will use this file name for calling Txt2gcal. Otherwise, the file name txt2gcal —which is burned-in during the compilation step of the Gcal program— is used for calling the Txt2gcal program. The TXT2GCALPROG environment variable must always be set if the Txt2gcal program is installed under another name than the standard name txt2gcal, otherwise Gcal is unable to execute the Txt2gcal program automatically!

If the simple '--debug' respectively '--debug=internal' option is given at program start-up, informational messages about the executed command and its *exit code* will be shown on the *standard error channel* (see [Global option '--debug=internal'], page 10).

If the '--debug=abort' option is given, the Gcal program will be aborted with an error code in case an *exit code* not equal zero occurred during the execution of the command. See [Error Code 2], page 239, and [Global option '--debug=abort'], page 11, for further information.

Here is an example which explains how the output of external commands can be assigned to text variables. Supposing there is a resource file by the name of 'tvarc-1.rc' with the following contents:

```
$ cat tvarc-1.rc
⊣ %t  $a $>+06*b $>1u*c
```

This file is used in the resource file 'tvarc-1a.rc' and processed by Gcal as follows:

```
$ cat tvarc-1a.rc
⊣ ; tvarc-1a.rc
⊣ ;
⊣ $a=bonjour
⊣ $b=123
⊣ $c=bonsoir
⊣ $x?cat $f
⊣ 0 \$x=---$:30*x---
⊣ $y:cat $f
⊣ 0 \$y=---$y---
⊣ 0 \$y=:::$>1u*y:::
$ gcal -f tvarc-1a.rc -QUx --exe -r'$f=./tvarc-1.rc'
⊣ $x=---    18:54 bonjour +00123 BONSOIR     ---
⊣ $y=---%t  $a $>+06*b $>1u*c---
⊣ $y=:::%T  $A $>+06*B $>1U*C:::
```

Here are some more examples showing how Gcal processes text variables. Let us assume a resource file by the name of '`tvar-1.rc`' exists with the following contents:

```
$ cat tvar-1.rc
⊣ ; tvar-1.rc
⊣ ;
⊣ $a=foo
⊣ 0 \$a:1=$a
⊣ $c=$a
⊣ 0 \$c=$c
⊣ $b=$c bar $a
⊣ 0 \$b=$b
⊣ $a=bar $b baz $a booz
⊣ 0 \$a:2=$a
⊣ $a=
⊣ $b=0 $b $c frozz $a frozz
⊣ $b ->That's \$b
⊣ 0 \$x='$x' and \$d is undefined: $d...
⊣ $a= 0 ~ 1~2~\$3~%n~$c~\
⊣ now it's enough!
⊣ 0 \$a=---$a---
⊣ $a=0 \   \ \\ And this...
⊣ $a works too!
```

So Gcal creates the following output:

```
$ gcal %19960101 -H no -x -r '$A=FROBOZZ:$x=' -f ./tvar-1.rc
⊣
⊣ Mon, Jan   1st 1996:    \ \\ And this... works too!
⊣ Mon, Jan   1st 1996: $a:1=foo
⊣ Mon, Jan   1st 1996: $a:2=bar foo bar foo baz foo booz
⊣ Mon, Jan   1st 1996: $a=--- 0
⊣                       1
⊣                      2
⊣                      $3
⊣                      01-Jan-1996
⊣                      foo
⊣                      now it's enough!---
⊣ Mon, Jan   1st 1996: $b=foo bar foo
⊣ Mon, Jan   1st 1996: $c=foo
⊣ Mon, Jan   1st 1996: $x='' and $d is undefined: $d...
⊣ Mon, Jan   1st 1996: foo bar foo foo frozz FROBOZZ frozz ->That's $b
```

Or a resource file by the name of 'tvar-2.rc' exists with the following contents:

```
$ cat tvar-2.rc
⊣ ; tvar-2.rc
⊣ ;
⊣ $a=$b foo
⊣ $b=0@e
⊣ $a bar
⊣ $b \$b
⊣ 0 \$a:$a
⊣ 0 \$b:$b
```

So Gcal creates the following output:

```
$ gcal %19960101 -H no -x -f ./tvar-2.rc
⊣
⊣ Mon, Jan   1st 1996: $a:0@e foo
⊣ Mon, Jan   1st 1996: $b:0@e
$ gcal %19960101 -H no -x -f ./tvar-2.rc -y
⊣
⊣ Sun, Apr   7th 1996: $b
⊣ Sun, Apr   7th 1996: foo bar
```

As seen before, it is allowed to store complete (or partial) date parts (see Section 5.1.2 [Date part of a line], page 81), likewise special texts (see Section 5.1.5 [%?... Special Texts], page 94) into text variables or references to other text variables, which are processed by Gcal after their expansion. See Appendix G [Description of all %?... Special Texts], page 137, for limitations concerning the assignment of special texts to text variables.

Text variables which are defined in a resource file are always local and only respected in this specific file, and that file which is included by it. This means for the included file, that **all** local variables of the caller are visible. The included file itself may define its own local variables in its own name-space, which are not exported to caller on return. All other definitions of text variables[1] are always global. If a resource file line contains a reference to a text variable which is

[1] In the command line, environment variable GCAL or response file.

not locally defined in that file, Gcal will try to use the global values held by this variable. If this fails because no global definition of this text variable is given, Gcal keeps the name of this text variable untouched in this line, except this text variable held an *empty value*. In such a case, Gcal completely ignores this text variable while processing and displaying instead of keeping its name untouched in this line.

An example to this. Supposing a resource file by the name of '`tvar-3.rc`' exists with the following contents:

```
$ cat tvar-3.rc
⊣ ; tvar-3.rc
⊣ ;
⊣ $c=+00+000
⊣ $h=
⊣ $d=$c$h
⊣ 0 %o$d  %s$d
⊣ 0 \$c=$c  \$h=$h  \$d=$d  \$x=$x
```

So Gcal creates the following output:

```
$ gcal -QUx -f ./tvar-3.rc
⊣ $c=+00+000  $h=  $d=+00+000  $x=$x
⊣ 05:57 18:04
$ gcal -QUx -r '$h=+1000:$x=' -f ./tvar-3.rc
⊣ $c=+00+000  $h=+1000  $d=+00+000+1000  $x=
⊣ 05:52 18:09
```

Only advanced users should apply the '`--export-text-variables`' option which causes that the actual incarnation of a local text variable —which was previously defined in a resource file and not in an included file— is being exported to further resource files instead of using its global value, in case that text variable is not locally defined in that further resource file. See Section 5.1.7 [Include directives], page 96, for more details.

But be aware, the use of this option could create unwanted results, because the order of processing the resource files is an important entity in managing the '`--export-text-variables`' option, so it is not recommended to use it. You, the user, must exactly know what you are doing when applying this option; **you are expressively warned now!**

Some basic operations can be performed on text variables in case they contain integer values. These are:

| **Operation** | **Description** |
|---|---|
| `*tvar*++` | Simple increment by one. |
| `*tvar*--` | Simple decrement by one. |
| `*tvar*+=[+\|-]*n*` | Addition of a constant numerical factor [`+\|-`]*n*. |
| `*tvar*-=[+\|-]*n*` | Subtraction of a constant numerical factor [`+\|-`]*n*. |

Here is an example showing how Gcal processes text variable operations. Let us assume a resource file by the name of '`tvaro-1.rc`' exists with the following contents:

```
$ cat tvaro-1.rc
⊣ ; tvaro-1.rc
⊣ ;
⊣ $a=130
⊣ $b=2
⊣ 0 1. \$b=$b
⊣ $b++
⊣ 0 2. \$b=$b
⊣ $b+=7
⊣ 0 3. \$b=$b
⊣ $b--
⊣ 0 4. \$b=$b
⊣ $b-=-5
⊣ 0 5. \$b=$b
⊣ $b+=123
⊣ 0 6. \$b=$b
⊣ $b-=$a
⊣ 0 7. \$b=$b
⊣ $b-=10000
⊣ 0 8. \$b=$b
⊣ $b+=10000
⊣ 0 9. \$b=$b
⊣ $b=02
⊣ 0 a. \$b=$b
⊣ $b++
⊣ 0 b. \$b=$b
```

So Gcal creates the following output:

```
$ gcal -QUx -f ./tvaro-1.rc
⊣ 1. $b=2
⊣ 2. $b=3
⊣ 3. $b=10
⊣ 4. $b=09
⊣ 5. $b=14
⊣ 6. $b=137
⊣ 7. $b=007
⊣ 8. $b=-9993
⊣ 9. $b=00007
⊣ a. $b=02
⊣ b. $b=03
```

As you can see in the former example, if operations are made like these, the default behavior of Gcal is to fill with leading zeroes to preserve a former length of a text variable, in case a carry into one of the next decimal places has happened and is taken back by an operation at a later place in the resource file. But this default behavior can be disabled respectively changed by using the already above mentioned format instruction.

The scope of the definitions which are done on a local text variable, is that resource or include file, where the local text variable is defined.

If a global text variable is redefined in a resource file or that file which is included by it, these redefinitions are only valid in that specific file[2], not in further resource files processed.

[2] They are only done locally.

This means, the initial values of global text variables are always restored if the resource file changes.

Please finally note, that each text variable assignment/operation must be given separately on a single line in the resource file.

## 5.2 Resource file examples

Here are some examples of problem oriented resource files which might be useful to get a deeper impression about the attributes of Gcal's fixed date feature.

Let us begin with a common example, the resource file 'common.rc':

```
$ cat common.rc
⊣ ; common.rc, a common example of a resource file.
⊣ ;
⊣ 19940715 Local time: %t  --> Dentist at 10:00
⊣ 00000921 My %B1962  birthday%i1952#2062
⊣ 0000093 Gregorian~Reformation\~is %y1752  years ago
⊣ 0000Sep03 Gregorian~Reformation\~is %y1752 years ago
⊣ 0 Every day in every month in every year
⊣ 0000 Every day in every month in every year
⊣ 00000000 Every day in every month in every year
⊣ 199400fri        Every Friday in 1994
⊣ 000007mo3 Every third Monday~in July every year less 1980%i#1979
⊣ 0000@e-10 Easter Sunday-10 days every year
⊣ 0@e-10 Easter Sunday-10 days every year
⊣ 0000@e+10 Easter Sunday+10 days every year
⊣ a=1127
⊣ 0@a+20 20 days after date of date variable 'a'
⊣ a=*W10FR
⊣ 0@a-1 The Thursday of the 10th week every year
⊣ 0@a   The Friday of the 10th week every year
⊣ 0@a1  The Saturday of the 10th week every year
⊣ 1995*d10    The 10th absolute day in 1995
⊣ 1995*d99tue The last Tuesday in 1995
⊣ 0*w99su The Sunday of the last week every year
⊣ ;
⊣ ; Next entry is respected by ALL fixed date period modifiers.
⊣ ;
⊣ 1994*d1sun#99SUN.7 Each Sunday in 1994
⊣ ;
⊣ ; Day of Prayer & Repentance is Wednesday before the Sunday,
⊣ ;   which is before the 1st Advent's Sunday.
⊣ ;
⊣ a=1225
⊣ a=a-5sun
⊣ 0@a-4 Day of Prayer & Repentance
⊣ ; or
⊣ 0@a-1wed Day of Prayer & Repentance
```

And here is a resource file by the name of ‘demo-1.rc’, used to detect all Fridays the 13th of a year:

```
$ cat demo-1.rc
⊣ ; demo-1.rc
⊣ ;
⊣ $a=Friday the 13th%i0000
⊣ $b=#0000
⊣ $c=fri2
⊣ $d=fri3
⊣ 00000113 $a01$c$b01$c
⊣ 00000113 $a01$d$b01$d
⊣ 00000213 $a02$c$b02$c
⊣ 00000213 $a02$d$b02$d
⊣ 00000313 $a03$c$b03$c
⊣ 00000313 $a03$d$b03$d
⊣ 00000413 $a04$c$b04$c
⊣ 00000413 $a04$d$b04$d
⊣ 00000513 $a05$c$b05$c
⊣ 00000513 $a05$d$b05$d
⊣ 00000613 $a06$c$b06$c
⊣ 00000613 $a06$d$b06$d
⊣ 00000713 $a07$c$b07$c
⊣ 00000713 $a07$d$b07$d
⊣ 00000813 $a08$c$b08$c
⊣ 00000813 $a08$d$b08$d
⊣ 00000913 $a09$c$b09$c
⊣ 00000913 $a09$d$b09$d
⊣ 00001013 $a10$c$b10$c
⊣ 00001013 $a10$d$b10$d
⊣ 00001113 $a11$c$b11$c
⊣ 00001113 $a11$d$b11$d
⊣ 00001213 $a12$c$b12$c
⊣ 00001213 $a12$d$b12$d
```

The result for the years 1996. . .2002 is as follows:

```
$ gcal -H no -u -x -f ./demo-1.rc 1996+2002
⊣
⊣ Fri, Sep  13th 1996: Friday the 13th
⊣ Fri, Dec  13th 1996: Friday the 13th
⊣
⊣ Fri, Jun  13th 1997: Friday the 13th
⊣
⊣ Fri, Feb  13th 1998: Friday the 13th
⊣ Fri, Mar  13th 1998: Friday the 13th
⊣ Fri, Nov  13th 1998: Friday the 13th
⊣
⊣ Fri, Aug  13th 1999: Friday the 13th
⊣
⊣ Fri, Oct  13th 2000: Friday the 13th
⊣
⊣ Fri, Apr  13th 2001: Friday the 13th
⊣ Fri, Jul  13th 2001: Friday the 13th
⊣
⊣ Fri, Sep  13th 2002: Friday the 13th
⊣ Fri, Dec  13th 2002: Friday the 13th
```

Or a bit shorter version of the resource file ‘`demo-1.rc`’ by the name of ‘`demo-2.rc`’, likewise used to detect all Fridays the 13th of a year, and that by using the fact, that a Friday the 13th only occurs if the first day of a month is a Sunday:

```
$ cat demo-2.rc
⊣ ; demo-2.rc
⊣ ;
⊣ $a=%n+12 is a Friday%i0000
⊣ $b=#0000
⊣ $c=sun1
⊣ 00000101 $a01$c$b01$c
⊣ 00000201 $a02$c$b02$c
⊣ 00000301 $a03$c$b03$c
⊣ 00000401 $a04$c$b04$c
⊣ 00000501 $a05$c$b05$c
⊣ 00000601 $a06$c$b06$c
⊣ 00000701 $a07$c$b07$c
⊣ 00000801 $a08$c$b08$c
⊣ 00000901 $a09$c$b09$c
⊣ 00001001 $a10$c$b10$c
⊣ 00001101 $a11$c$b11$c
⊣ 00001201 $a12$c$b12$c
```

The result for the years 1996. . . 2002 is as follows:

```
$ gcal -H no -u -x -f ./demo-2.rc 1996+2002
⊣
⊣ Sun, Sep   1st 1996: 13-Sep-1996 is a Friday
⊣ Sun, Dec   1st 1996: 13-Dec-1996 is a Friday
⊣
⊣ Sun, Jun   1st 1997: 13-Jun-1997 is a Friday
⊣
⊣ Sun, Feb   1st 1998: 13-Feb-1998 is a Friday
⊣ Sun, Mar   1st 1998: 13-Mar-1998 is a Friday
⊣ Sun, Nov   1st 1998: 13-Nov-1998 is a Friday
⊣
⊣ Sun, Aug   1st 1999: 13-Aug-1999 is a Friday
⊣
⊣ Sun, Oct   1st 2000: 13-Oct-2000 is a Friday
⊣
⊣ Sun, Apr   1st 2001: 13-Apr-2001 is a Friday
⊣ Sun, Jul   1st 2001: 13-Jul-2001 is a Friday
⊣
⊣ Sun, Sep   1st 2002: 13-Sep-2002 is a Friday
⊣ Sun, Dec   1st 2002: 13-Dec-2002 is a Friday
```

Of course, preceding textual expense for the purpose of detecting all Fridays the 13th of a year can be dramatically reduced by simply using of

```
0*d1fri#99fri.7 Friday the 13th%i00000013#00000013
```

in the resource file '`demo-1.rc`', respectively,

```
0*d1sun#99sun.7 %n+12 is a Friday%i00000001#00000001
```

in the '`demo-2.rc`' resource file.

On the one hand, preceding *one-liners* reduce the coding expense quite considerably —from 28 respectively 15 lines to 1 line in each case—, but on the other hand, this partly increases the checking expense of these *one-liners* considerably and result in a longer program run-time therefore. Both *one-liners* produce about 52 internal productions respectively effective fixed date entries of themselves for each year checked by Gcal, at which the '`demo-1.rc`' and '`demo-2.rc`' resource files cause 24 respectively 12 effective fixed date entries only.

And here is a resource file by the name of ‘demo-3.rc’, used to detect all working-days of any month:

```
$ cat demo-3.rc
⊣ ; demo-3.rc
⊣ ;
⊣ $a=%VWork day
⊣ $c=%i#0000$b99
⊣ a=$bmon1
⊣ 0@a#+4 $a
⊣ a=$btue1
⊣ 0@a#+3 $a
⊣ a=$bwed1
⊣ 0@a#+2 $a
⊣ a=$bthu1
⊣ 0@a#+1 $a
⊣ a=$bfri1
⊣ 0@a $a
⊣ a=$bmon2
⊣ 0@a#+4 $a
⊣ a=$bmon3
⊣ 0@a#+4 $a
⊣ a=$bmon4
⊣ 0@a#+4 $a$c
⊣ a=$bmon9
⊣ 0@a#+4 $a$c
```

Respectively a bit shorter version of the resource file ‘demo-3.rc’:

```
; demo-3.rc
;
$a=%VWork day
$c=%i#0000$b99
0000$bmon1:5 $a
0000$btue1:4 $a
0000$bwed1:3 $a
0000$bthu1:2 $a
0000$bfri1:1 $a
0000$bmon2:5 $a
0000$bmon3:5 $a
0000$bmon4:5 $a$c
0000$bmon9:5 $a$c
```

And a resource file by the name of ‘`include.rc`’ exists which is only used to include one given resource file:

```
$ cat include.rc
⊣ #include "$f"
```

Then the result for the month August 1996 is as follows:

```
$ gcal -H no -u -x -r '$b=08:$f=demo-3.rc' -f ./include.rc Aug 1996
⊣
⊣ Thu, Aug   1st 1996: Work day
⊣ Fri, Aug   2nd 1996: Work day
⊣ Mon, Aug   5th 1996: Work day
⊣ Tue, Aug   6th 1996: Work day
⊣ Wed, Aug   7th 1996: Work day
⊣ Thu, Aug   8th 1996: Work day
⊣ Fri, Aug   9th 1996: Work day
⊣ Mon, Aug  12th 1996: Work day
⊣ Tue, Aug  13th 1996: Work day
⊣ Wed, Aug  14th 1996: Work day
⊣ Thu, Aug  15th 1996: Work day
⊣ Fri, Aug  16th 1996: Work day
⊣ Mon, Aug  19th 1996: Work day
⊣ Tue, Aug  20th 1996: Work day
⊣ Wed, Aug  21st 1996: Work day
⊣ Thu, Aug  22nd 1996: Work day
⊣ Fri, Aug  23rd 1996: Work day
⊣ Mon, Aug  26th 1996: Work day
⊣ Tue, Aug  27th 1996: Work day
⊣ Wed, Aug  28th 1996: Work day
⊣ Thu, Aug  29th 1996: Work day
⊣ Fri, Aug  30th 1996: Work day
```

And of course, the preceding textual expense for the purpose of detecting all work days of any month can be dramatically reduced by simply using of

```
00000001#0099 %V%rWork day
```

in the resource file ‘`demo-3.rc`’.

And next the resource file by the name of ‘`sun-1.rc`’ which detects the actual local time, the approximate time of sunrise and sunset, day and night length of two geographic locations:

```
$ cat sun-1.rc
⊣ ; sun-1.rc, comparison of sunrise / sunset, day length and night
⊣ ; length of two locations, namely:  Muenster.DE / Bangkok.TH
⊣ ; All references are based on Muenster.DE !
⊣ ;
⊣ ; Common definitions
⊣ $p=0*d1#999
⊣ $v=%2%4 for that places local time
⊣ $z=actual local time in
⊣ ; Definitions for winter time
⊣ $a=%e#1980 %e0@a#0@b-1
⊣ $x=Sun:~%o$c,$w rise, %s$c,$w set, %u$c,$w day, %z$c,$w night~$l
⊣ ; Winter time in Germany == CEWT/GMT-1 (+60 minutes)
⊣ $w=+60
⊣ ; Starting date of winter time in Germany
⊣ b=10sun9
⊣ ; Definitions for summer time (daylight-saving time)
⊣ $b=%e#1980 %i0@a#0@b-1
⊣ $y=Sun:~%o$c,$s rise, %s$c,$s set, %u$c,$s day, %z$c,$s night~$l
⊣ ; Summer time in Germany == CEST/GMT-2 (+120 minutes)
⊣ $s=+120
⊣ ; Starting date of summer time in Germany
⊣ a=03sun9
⊣ ; Geographic co-ordinate and name of the location Muenster, Germany
⊣ $c=+5158+00738+61
⊣ $l=%1Muenster
⊣ ; Muenster's local time sunrise etc. for that places local time
⊣ $p $a $x$v
⊣ $p $b $y$v
⊣ ; Muenster's actual local time
⊣ 0 $a %1%@$w %2 $z $l
⊣ 0 $b %1%@$s %2 $z $l
⊣ ; Geographic co-ordinate and name of the location Bangkok, Thailand
⊣ $c=+1345+10031+10
⊣ $l=%3Bangkok%4-time converted to $l's%2 local time
⊣ ; Bangkok's sunrise etc. based on Muenster's local time
⊣ $p $a $x
⊣ $p $b $y
⊣ ; Bangkok's local time == WAST/GMT-7 (+420 minutes)
⊣ $t=+420
⊣ $w=$t
⊣ ; Bangkok's actual local time
⊣ $l=%3Bangkok
⊣ 0 $a %3%@$t %4 $z $l
⊣ 0 $b %3%@$t %4 $z $l
⊣ ; Bangkok's sunrise etc. for that places local time
⊣ $l=$l$v
⊣ $p $x
```

The result for the 12th September 1999 is as follows:

```
$ gcal -fsun-1.rc -Ux -Hno
⊣
⊣ 18:53/12-Sep-1999 actual local time in Muenster
⊣ 23:53/12-Sep-1999 actual local time in Bangkok
⊣ Sun:
⊣ 01:06rise, 13:22set, 12h16'day, 11h44'night
⊣ Bangkok-time converted to Muenster's local time
⊣ Sun:
⊣ 06:06rise, 18:22set, 12h16'day, 11h44'night
⊣ Bangkok for that places local time
⊣ Sun:
⊣ 06:56rise, 19:54set, 12h58'day, 11h02'night
⊣ Muenster for that places local time
```

The next example is the resource file '`redir-1.rc`' which demonstrates the use of the shell command `%![`*argument*`]` special text:

```
$ cat redir-1.rc
⊣ ; redir.rc, run shell command $c and redirect it to $f
⊣ ;
⊣ $p=0*d1#999
⊣ $c=$p %!$c
⊣ $g=|txt2gcal - $p
⊣ ;
⊣ $c $g>$f
```

The result of succeeding call of Gcal, to which you can also add the '`--debug=all`' option for a better understanding of the execution methods, is as follows:

```
$ gcal --exe -QUx -f./redir-1.rc+XX -r'$f=XX:$c=echo $f' -#'0 %!rm $f'
⊣ XX
```

And finally a resource file by the name of 'swtim-1.rc' which produces fixed date messages reminding Daylight Saving that are valid since 1981 for the Federal Republic of Germany[1].

```
$ cat swtim-1.rc
⊣ ; swtim-1.rc, daylight-saving time BRD
⊣ ;
⊣ $a=daylight-saving time starts Sunday
⊣ ;
⊣ a=03sun9
⊣ 0@a-2#+2 Summer $a~+1 hrs. (02:00 --> 03:00 h.)%i1981
⊣ ;
⊣ a=09sun9
⊣ 0@a-2#+2 Winter $a~-1 hrs. (03:00 --> 02:00 h.)%i1981#1995
⊣ ;
⊣ a=10sun9
⊣ 0@a-2#+2 Winter $a~-1 hrs. (03:00 --> 02:00 h.)%i1996
```

The result for the year 1998 is as follows:

```
$ gcal %1998 -Hno -xy -f ./swtim-1.rc
⊣
⊣ Fri, Mar  27th 1998: Summer daylight-saving time starts Sunday
⊣                      +1 hrs. (02:00 --> 03:00 h.)
⊣ Sat, Mar  28th 1998: Summer daylight-saving time starts Sunday
⊣                      +1 hrs. (02:00 --> 03:00 h.)
⊣ Sun, Mar  29th 1998: Summer daylight-saving time starts Sunday
⊣                      +1 hrs. (02:00 --> 03:00 h.)
⊣ Mon, Mar  30th 1998: Summer daylight-saving time starts Sunday
⊣                      +1 hrs. (02:00 --> 03:00 h.)
⊣ Tue, Mar  31st 1998: Summer daylight-saving time starts Sunday
⊣                      +1 hrs. (02:00 --> 03:00 h.)
⊣ Fri, Oct  23rd 1998: Winter daylight-saving time starts Sunday
⊣                      -1 hrs. (03:00 --> 02:00 h.)
⊣ Sat, Oct  24th 1998: Winter daylight-saving time starts Sunday
⊣                      -1 hrs. (03:00 --> 02:00 h.)
⊣ Sun, Oct  25th 1998: Winter daylight-saving time starts Sunday
⊣                      -1 hrs. (03:00 --> 02:00 h.)
⊣ Mon, Oct  26th 1998: Winter daylight-saving time starts Sunday
⊣                      -1 hrs. (03:00 --> 02:00 h.)
⊣ Tue, Oct  27th 1998: Winter daylight-saving time starts Sunday
⊣                      -1 hrs. (03:00 --> 02:00 h.)
```

[1] Two days before adaptation till two days after adaptation.

# Appendix A Genesis of the Gregorian Calendar

The nowadays calendar was first formulated in several inaccurate variations by the Romans based on methods developed by the Babylonians and Egypts. The aim of all these calendars was to harmonize the cycles of the Moon and the Sun. During Julius Caesar's reign, January was falling in autumn so he ordered Sosigenes to make changes to the calendar. He added 90 days to the year 46 BC to make up for the seasonal drift and adjusted the lengths of the months similarly as we know them to be today. He introduced the leap year by adding one day to February every four years. For the present, the leap year regulation was made in the way that all four years after the 23rd day in February a leap day was laid in, so the 24th February occurred twice. The use of the leap year was an improvement but not entirely accurate.

But in the later years, the leap rule was used in the wrong way so that the errors are corrected by emperor Augustus in the year 8 BC. A curious sequel happened on this occasion. Because Augustus reacted with great jealousy to all things previously made and propagated by Julius Caesar, he did not like Caesar's name in the calendar, namely the today's month of **July**. Offhandedly he ordered to name another month to himself and so the month name **August** arose. Furthermore, Augustus did not tolerate the fact that his month of birth (the August) was shorter than Caesar's month in the sense of the periodical sequence of months with 30 and 31 days. Consequently, the month of August got 31 days, too. Due to this modification, the number of days in February were reduced to 28 and 29 days, respectively, so the 29th February was designed to be the leap day now.

This calendar is well known under the term *Julian calendar* and is based on a plain solar year. The nominal length of a solar year (respectively a so-called tropical year) is 365 days, 5 hours, 48 minutes, and 46 seconds. One 366-day year every four years equates to an average Julian year of 365 days and 6 hours, consequently to 365.25 days. This means, every four years, an error of 44 minutes, 56 seconds was accumulated by this kind of calendar calculation. Because of this counting method, the length of the years becomes a bit too long, by more than 11 minutes.

By the 16th century, the vernal equinox occurred around March 11, rather than March 21, by reason of an accumulated error of ten days. The feast of Easter shifted away more and more from the habitual vernal date, which must have always been celebrated on an earlier date. So Pope Gregory XIII introduced the new style calendar in 1582. Thursday, the 4th October 1582, was followed by Friday, the 15th October, by suppressing the ten days between both dates. Moreover, he ordained that years ending in hundreds should not be leap years unless they are divisible by 400. Incidentally, the Gregorian reform compensates by 72 hours (3 days) every 400 years. The actual excess accumulated is 74 hours, 53 minutes and 20 seconds. The error of 2 hours, 53 minutes and 20 seconds every 400 years accumulates to one day in about 3300 years. Thereby, the Gregorian year has an average length of 365.2425 days.

But this *Gregorian calendar* was accepted very slowly by others. Catholic countries adopted the Gregorian calendar almost immediately. Most Protestant countries on the Continent adopted the new calendar around 1700. England and the American colonies changed in 1752, by act of Parliament. Orthodox Christian countries adopted the Gregorian calendar later. Russia was the last European country to do so, after the communist revolution of 1917. As a result, the former U.S.S.R. celebrated the October Revolution (happened on October 25th, 1917) in the old style calendar on November 7th.

The era of a world wide uniform calendar is already part of history today. The Iran returned to the traditional Mohammedan lunar calendar in 1979 after removal of the Shah reign. There are some efforts to improve our currently valid Gregorian calendar. Its disadvantages are the reason why an appointed day is not always on the same week day. Besides, the month lengths are not equal and the holidays, which have relations to the feast of Easter, are moved within the calendar from one year to another. A very sophisticated suggestion was proposed by the United

Nations, but the international establishment of this suggestions has failed, since it was resisted by some countries as well as the churches.

# Appendix B Gcal Utilities

Three small utility programs are part of the Gcal 3.01 package. The Tcal program runs the Gcal program with the date set one day ahead. The Txt2gcal program creates a verbatim Gcal resource file from a text file, and in contrast to this, the Gcal2txt program creates a verbatim text file from a Gcal resource file.

## B.1 Invoking `tcal`

The Tcal program runs the Gcal program with the date of tomorrow's day[1]. Its arguments are as follows:

```
tcal [--help | --version] | [--shift=[+|-]number] [argument...]
```

All given *arguments* are passed unmodified to the Gcal program. If the Gcal program shall be called with another date than tomorrow's date, this desired date can be selected by using the '`--shift=[+|-]number`' option, in which '`[+|-]number`' is the distance of days the desired date is distant from the actual system date (≡today). (Works only for Gregorian years.) The '`--shift`' option must be given before all other *arguments* that are passed to the Gcal program.

In case a `GCALPROG` environment variable (see [Environment Variable `GCALPROG`], page 237) is defined and set with the file name of the executable Gcal program, Tcal will use this file name for calling Gcal. Otherwise, the file name `gcal` —which is burned-in during the compilation step of the Tcal program— is used for calling the Gcal program. The `GCALPROG` environment variable must always be set if the Gcal program is installed under another name than the standard name `gcal`, otherwise Tcal is unable to execute the Gcal program automatically! An exit status of 0 means all processing is successfully done, any other value means an error has occurred.

The program accepts the following options:

'`--help`' Print a usage message listing all available options, then exit successfully.

'`--version`'
Print the version number, then exit successfully.

'`--shift=[+|-]number`'
Define the displacement in '`[+|-]number`' days the desired date is distant from the actual date.

## B.2 Invoking `txt2gcal`

The Txt2gcal program creates a verbatim Gcal resource file from a text file. Its arguments are as follows:

```
txt2gcal [--help | --version] | [text-file|-] [date-part]
```

If no *text-file* argument is given or/but a single '`-`' character, the program reads and processes all input received from the *standard input channel*. If no *date-part* argument is given, Txt2gcal creates a '`0`' for the date part. All results are always shown on the *standard output channel*. An exit status of 0 means all processing is successfully done, any other value means an error has occurred.

[1] This is equivalent the '`--shift=1`' option.

The program accepts the following options:

‘`--help`’ Print a usage message listing all available options, then exit successfully.

‘`--version`’
Print the version number, then exit successfully.

Here comes an example how to use Txt2gcal. Let us suppose there is a text file by the name of ‘`tdemo-1.txt`’ with the following contents:

```
$ cat tdemo-1.txt
⊣         Hi friends,
⊣
⊣ I’m the demo text containing funny characters and character
⊣ sequences like ~~ \~ % %% $a $1 %%%\
⊣ %\ %s %%foo %bar \%Baz \\~ \~\ and so on...
⊣ I’m be anxious how I’ll be transformed by ‘txt2gcal’.
⊣
```

and Txt2gcal processes this file, redirecting the output to ‘`tdemo-1.rc`’:

```
$ txt2gcal tdemo-1.txt 0*d1#999 > tdemo-1.rc
$ cat tdemo-1.rc
⊣ 0*d1#999 \        Hi friends,~\
⊣ ~\
⊣ I’m the demo text containing funny characters and character~\
⊣ sequences like \~\~ \\~ % %% \$a $1 %%%\ ~\
⊣ %\ \%s %\%foo \%bar \\%Baz \\\~ \\~\ and so on...~\
⊣ I’m be anxious how I’ll be transformed by ‘txt2gcal’.~\
```

then Txt2gcal has now created a new Gcal resource file ‘`tdemo-1.rc`’ from the text file ‘`tdemo-1.txt`’. Let us finally see, how Gcal will interpret this new resource file:

```
$ gcal %19960101 -H no -Ax -f ./tdemo-1.rc
⊣
⊣ Mon, Jan   1st 1996:
⊣         Hi friends,
⊣
⊣ I’m the demo text containing funny characters and character
⊣ sequences like ~~ \~ % %% $a $1 %%%\
⊣ %\ %s %%foo %bar \%Baz \\~ \~\ and so on...
⊣ I’m be anxious how I’ll be transformed by ‘txt2gcal’.
⊣
```

## B.3 Invoking `gcal2txt`

The Gcal2txt program creates a verbatim text file from a Gcal resource file. Its arguments are as follows:

`gcal2txt` [`--help` | `--version`] | [*`resource-file`*|`-`]

If no *resource-file* argument is given or/but a single ‘`-`’ character, the program reads and processes all input received from the *standard input channel*. All results are always shown on the *standard output channel*. An exit status of 0 means all processing is successfully done, any other value means an error has occurred.

The sense and purpose of Gcal2txt is to retrieve all additional texts, which are put into the output by means of the Txt2gcal program.

The program accepts the following options:

‘--help’ Print a usage message listing all available options, then exit successfully.

‘--version’
Print the version number, then exit successfully.

Here comes an example how to use Gcal2txt. For that purpose, let us use the resource file ‘tdemo-1.rc’ of the previous section (see Section B.2 [Invoking txt2gcal], page 119), which was created by means of the Txt2gcal program from a text file and which looks like this:

```
$ txt2gcal tdemo-1.txt 0*d1#999 > tdemo-1.rc
$ cat tdemo-1.rc
⊣ 0*d1#999 \        Hi friends,~\
⊣ ~\
⊣ I’m the demo text containing funny characters and character~\
⊣ sequences like \~\~ \\~ % %% \$a $1 %%%\ ~\
⊣ %\ \%s %\%foo \%bar \\%Baz \\\~ \\~\ and so on...~\
⊣ I’m be anxious how I’ll be transformed by ‘txt2gcal’.~\
```

Now let us see, how this resource file will be processed by Gcal2txt:

```
$ gcal2txt tdemo-1.rc
⊣         Hi friends,
⊣
⊣ I’m the demo text containing funny characters and character
⊣ sequences like ~~ \~ % %% $a $1 %%%\
⊣ %\ %s %%foo %bar \%Baz \\~ \~\ and so on...
⊣ I’m be anxious how I’ll be transformed by ‘txt2gcal’.
```

# Appendix C Aspects in Internationalization

Starting with version 2.00, Gcal is able to display message texts using any native languages instead of using the English language only, because parts of the GNU `gettext` package are integrated into the Software. See section "Introduction" in *GNU gettext tools*, for more details.

By default, Gcal displays all message texts using the English native language in case no other native language is wanted. A so-called *message catalog* is read by Gcal at run-time if message texts from another native language are required. Gcal 3.01 supports the following native languages:

| **Native Language** | **Language Code** |
|---|---|
| English | `en` |
| German | `de` |
| French | `fr` |
| Dutch | `nl` |
| Polish | `pl` |
| Russian | `ru` |
| Swedish | `sv` |

It is only necessary to set one of the environment variables[1]:

1. `LANGUAGE`
2. `LC_ALL`
3. `LC_MESSAGES`
4. `LANG`

with a language code to select another native language instead of the English native language.

Normally, users only have to set the `LANG` environment variable to tell Gcal the native language to use at run-time level. Presuming users want to run Gcal using the German native language for displaying message texts, they merely have to execute '`setenv LANG de`' (in `csh`) or '`export LANG; LANG=de`' (in `sh`) at the shell prompt. Of course they could even do this from their '`.login`' or '`.profile`' file. See section "The User's View" in *GNU gettext tools*, for more details.

As shown above, a simple setting of `de` in the environment variable `LANG` is sufficient to enable German message texts. `de` is the two-letter language code for the German language defined in the ISO-639:1988, and is called *simple language code information* in the further context. Other language codes can be taken from this ISO-document[2].

Because Gcal as calendar program must also comply the specifics of a used native language concerning the ordering of day, month and year (and further things) of a displayed date, the period of Gregorian Reformation, the type of week number and the representation of calendar sheets, these criteria are likewise bound to the language code[3].

A `en` language code causes the following internal defaults of above criteria:

- U.S. American date format
- Gregorian Reformation has occurred 1752

[1] Listed in decreasing priority as they are respected.

[2] For example `fr` for French, `es` for Spanish. . .

[3] Strictly speaking, an extended language code information.

- Week starts on Sunday
- Standard week numbers
- Representation of calendar sheet

```
      September 1994
   Su Mo Tu We Th Fr Sa
                1  2  3
    4  5  6  7  8  9 10
   11 12 13 14 15 16 17
   18 19 20 21 22 23 24
   25 26 27 28 29 30
```

And a `de` language code[4] causes the following internal defaults:

- German date format
- Gregorian Reformation has occurred 1582
- Week starts on Monday
- ISO-8601:1988 week numbers
- Representation of calendar sheet

```
   September 1994

   Monday          5 12 19 26
   Tuesday         6 13 20 27
   Wednesday       7 14 21 28
   Thursday     1  8 15 22 29
   Friday       2  9 16 23 30
   Saturday     3 10 17 24
   Sunday       4 11 18 25
```

Remember, all these internal defaults are modifiable by the options '`--date-format`', '`--gregorian-reform`', '`--starting-day`', '`--iso-week-number`' and '`--type`'.

If no language code is detected, Gcal takes the internal defaults of the `en` language code[5].

If a language code is specified for which no *message catalog* is installed, Gcal takes the internal defaults of the `de` language code, but displays the message texts using the English native language. Actually, this behavior seems to me the most proper solution in such a case. The English native language is spoken all over the world unlike the German or other native languages, so it is wise to use it here. But the other criteria bound to the English native language are so special for users of other native languages, that it is wise to use the criteria taken for internal defaults of the `de` language code, because most European countries (taken as standard) essentially use them.

Now British users will certainly ask whether they could use their date format as an internal default[6]. The answer to this is a simple *yes*, nevertheless, these users have to set the environment variable `LANG` with an *extended language code information* instead of a simple language code information.

The usual template of an extended language code information is as follows:

- For XPG syntax (X/Open Portability Guide):

[4] Or other language codes, for which a *message catalog* will be created and distributed in future.

[5] Or to be more precise, of the extended language code information `en_US`.

[6] All other internal defaults of the simple `en` language code information just meet their criteria.

`language[_territory][.codeset][@modifier]`

- For CEN syntax (Comité Europén de Normalisation):

  `language[_territory][+audience][+special][,sponsor][_revision]`

Both syntaxes contain the *language* and *territory* components, which are used by Gcal to select the native language and the other criteria. The *language* component is equivalent to the simple language code information and the *territory* component is a two-letter territory or country code as defined by the ISO-3166 like ‘`GB`’ for Great Britain or ‘`US`’ for the U.S.A. See the pertinent literature for more details. So British users only have to set the `LANG` environment variable with a `en_GB` contents, and after that, they can use the British date format as an internal default.

# Appendix D Metasymbols

Within this document, the following *metasyntactic* conventions will be used when discussing command line options, commands, arguments and templates:

`n` Argument is not optional and *n* must be a valid number.

`e|t|dvar` Argument is not optional and must be a single alphabetic character.

`$tvar` Argument is not optional and must be a single alphabetic character, that is lead by a '`$`' character.

`[text]` Argument is optional and *text* is a valid string of text.

`[ab]` Arguments are optional, any number may be used in any order, e.g.:

`NOTHING, a, b, ab, ba ...`

`{ab}` Arguments are not optional, minimum one up to any number may be used in any order, e.g.:

`a, b, ab, ba ...`

`[a|b]` Arguments are optional, select either *a* or *b*, but not both.

`a|b` Arguments are not optional, select either *a* or *b*, but not both.

`[a[b]]` Arguments are optional, select either *a* or *ab*, but not *b*.

`[a|b[c]]` Arguments are optional, select *a* or *b* or *ac* or *bc*, but not *c*.

`[a|bc]` Arguments are optional, select either *ac* or *bc*, but not *a* or *b* or *c*.

`[a]|[b]` Argument *lists* are optional, select either list *a* or list *b*, but not both.

`[...]` Additional arguments/files may be specified and are optional.

# Appendix E Regular Expressions

This appendix is derived from the `gawk-3.0.0` and `regex-0.12` manuals.

A *regular expression*, or *regexp*, is a way of describing a set of strings. The simplest regular expression is a sequence of letters, numbers, or both. Such a regexp matches any string that contains that sequence. Thus, the regexp '`foo`' matches any string containing '`foo`'. Other kinds of regular expressions let you specify more complicated classes of strings.

Gcal uses exactly one of the following regular expression programming libraries respectively methods, and that in the decreasing priority how it is chosen at configuration time, or better, compile time of the software:

| No. | Function | Symbol |
|---|---|---|
| 1. | GNU `re_compile_pattern()` and '`regex.h`' | `GNU-REGEX` |
| 2. | POSIX `regcomp()` and '`regex.h`' | `POSIX-REGEX` |
| 3. | BSD `re_comp()` | `BSD-REGEX` |
| 4. | System V `regcmp()` | `SysV-REGEX` |
| 5. | Henry Spencer V8 `regcomp()` and '`regexp.h`' | `V8-REGEX` |
| 6. | Pattern matching is supported, but without metacharacters | `NO-REGEX` |

Try '`gcal --version`' for detecting the kind of regular expression programming library respectively method that is burned-in in your Gcal program!

## E.1 How to Use Regular Expressions

A regular expression can be used as a pattern if Gcal's '`--filter-text=`*`pattern`*' option is specified at program start-up.

See [Fixed date option '`--filter-text=`*`pattern`*'], page 41, for more details.

Then the regular expression is tested against the entire, completely expanded text of each **valid** fixed date, and the fixed date is only displayed in case the pattern matches the text[1].

## E.2 Regular Expression Operators

You can combine regular expressions with the following characters, called *regular expression operators*, or *metacharacters*, to increase the power and versatility of regular expressions.

[1] Normally, it only needs to match some part of the text in order to succeed.

Here is a table of these metacharacters. All characters that are not listed in the table stand for themselves.

`\` This is used to suppress the special meaning of a character when matching. For example:

```
\$
```

matches the character ‘`$`’.

`^` This matches the beginning of a string. For example:

```
^@chapter
```

matches the ‘`@chapter`’ at the beginning of a string, and can be used to identify chapter beginnings in Texinfo source files. The ‘`^`’ is known as an *anchor*, since it anchors the pattern to matching only at the beginning of the string.

`$` This is similar to ‘`^`’, but it matches only at the end of a string. For example:

```
p$
```

matches a string that ends with a ‘`p`’. The ‘`$`’ is also an anchor.

`.` The period, or dot, matches any single character. For example:

```
.P
```

matches any single character followed by a ‘`P`’ in a string. Using concatenation we can make a regular expression like ‘`U.A`’, which matches any three-character sequence that begins with ‘`U`’ and ends with ‘`A`’.

`[...]` This is called a *character list*. It matches any *one* of the characters that are enclosed in the square brackets. For example:

```
[MVX]
```

matches any one of the characters ‘`M`’, ‘`V`’, or ‘`X`’ in a string.

Ranges of characters are indicated by using a hyphen between the beginning and ending characters, and enclosing the whole thing in brackets. For example:

```
[0-9]
```

matches any digit. Multiple ranges are allowed. E.g., the list `[A-Za-z0-9]` is a common way to express the idea of “all alphanumeric characters.”

To include one of the characters ‘`\`’, ‘`]`’, ‘`-`’ or ‘`^`’ in a character list, put a ‘`\`’ in front of it. For example:

```
[d\]]
```

matches either ‘`d`’, or ‘`]`’.

*Character classes* are a new feature introduced in the POSIX standard. A character class is a special notation for describing lists of characters that have a specific attribute, but where the actual characters themselves can vary from country to country and/or from character set to character set. For example, the notion of what is an alphabetic character differs in the U.S.A. and in France.

A character class is only valid in a regexp *inside* the brackets of a character list. Character classes consist of ‘`[:`’, a keyword denoting the class, and ‘`:]`’. Here are the character classes defined by the POSIX standard:

`[:alnum:]`
Alphanumeric characters.

`[:alpha:]`
Alphabetic characters.

`[:blank:]`
Space and tab characters.

`[:cntrl:]`
Control characters.

`[:digit:]`
Numeric characters.

`[:graph:]`
Characters that are printable and are also visible[1].

`[:lower:]`
Lower-case alphabetic characters.

`[:print:]`
Printable characters[2].

`[:punct:]`
Punctuation characters[3].

`[:space:]`
Space characters[4].

`[:upper:]`
Upper-case alphabetic characters.

`[:xdigit:]`
Characters that are hexadecimal digits.

For example, before the POSIX standard, to match alphanumeric characters, you had to write `[A-Za-z0-9]`. If your character set had other alphabetic characters in it, this would not match them. With the POSIX character classes, you can write `[[:alnum:]]`, and this will match *all* the alphabetic and numeric characters in your character set.

Two additional special sequences can appear in character lists. These apply to non-ASCII character sets, which can have single symbols (called *collating elements*) that are represented with more than one character, as well as several characters that are equivalent for collating, or sorting, purposes. (E.g., in French, a plain '`e`' and a grave-accented '`è`' are equivalent.)

[1] A space is printable, but not visible, while an '`a`' is both.

[2] Characters that are not control characters.

[3] Characters that are not letter, digits, control characters, or space characters.

[4] Such as Space, Tab, and Form-feed, to name a few.

Collating Symbols

A *collating symbol* is a multi-character collating element enclosed in '`[.`' and '`.]`'. For example, if '`ch`' is a collating element, then `[[.ch.]]` is a regexp that matches this collating element, while `[ch]` is a regexp that matches either '`c`' or '`h`'.

Equivalence Classes

An *equivalence class* is a list of equivalent characters enclosed in '`[=`' and '`=]`'. Thus, `[[=eè=]]` is a regexp that matches either '`e`' or '`è`'.

These features are very valuable in non-English speaking locales.

**Caution:**
The library functions that Gcal uses for regular expression matching currently only recognize POSIX character classes (possibly); they do not recognize collating symbols or equivalence classes.

`[^ ...]` This is a *negated character list* respectively *complemented character list*. The first character after the '`[`' *must* be a '`^`'. It matches any characters *except* those in the square brackets. For example:

```
[^0-9]
```

matches any character that is not a digit.

`|` This is the *alternation operator*, and it is used to specify alternatives. For example:

```
^P|[0-9]
```

matches any string that matches either '`^P`' or '`[0-9]`'. This means it matches any string that starts with '`P`' or contains a digit.

The alternation applies to the largest possible regexps on either side. In other words, '`|`' has the lowest precedence of all the regular expression operators.

`(...)` Parentheses are used for grouping in regular expressions as in arithmetic. They can be used to concatenate regular expressions containing the alternation operator, '`|`'. For example, '`@(samp|code)\{[^}]+\}`' matches both '`@code{foo}`' and '`@samp{bar}`'. (These are Texinfo formatting control sequences.)

`*` This symbol means that the preceding regular expression is to be repeated as many times as necessary to find a match. For example:

```
ph*
```

applies the '`*`' symbol to the preceding '`h`' and looks for matches of one '`p`' followed by any number of '`h`'s. This will also match just '`p`' if no '`h`'s are present.

The '`*`' repeats the *smallest* possible preceding expression. (Use parentheses if you wish to repeat a larger expression.) It finds as many repetitions as possible. For example:

```
gcal --filter-text='\(c[ad][ad]*r x\)' -f sample.rc -y
```

prints every fixed date in '`sample.rc`' containing a fixed date text of the form '`(car x)`', '`(cdr x)`', '`(cadr x)`', and so on. Notice the escaping of the parentheses by preceding them with backslashes.

`+` This symbol is similar to '`*`', but the preceding expression must be matched at least once. This means that:

```
wh+y
```

would match '`why`' and '`whhy`' but not '`wy`', whereas '`wh*y`' would match all three of these strings. This is a simpler way of writing the last '`*`' example:

```
gcal --filter-text='\(c[ad]+r x\)' -f sample.rc -y
```

`?` This symbol is similar to '`*`', but the preceding expression can be matched either once or not at all. For example:

```
fe?d
```

will match '`fed`' and '`fd`', but nothing else.

`{`*n*`}`<br>
`{`*n*`,}`<br>
`{`*n*`,`*m*`}` One or two numbers inside braces denote an *interval expression* which is available in the POSIX standard. If there is one number in the braces, the preceding regexp is repeated *n* times. If there are two numbers separated by a comma, the preceding regexp is repeated *n* to *m* times. If there is one number followed by a comma, then the preceding regexp is repeated at least *n* times.

`wh{3}y` matches '`whhhy`' but not '`why`' or '`whhhhy`'.

`wh{3,5}y` matches '`whhhy`' or '`whhhhy`' or '`whhhhhy`', only.

`wh{2,}y` matches '`whhy`' or '`whhhy`', and so on.

GNU software that deals with regular expressions provides a number of additional regexp operators. These operators are described here.

Most of the additional operators are for dealing with word matching. For our purposes, a *word* is a sequence of one or more letters, digits, or underscores ('`_`').

`\w` This operator matches any word-constituent character, i.e. any letter, digit, or underscore. Think of it as a short-hand for `[A-Za-z0-9_]` or `[[:alnum:]_]`.

`\W` This operator matches any character that is not word-constituent. Think of it as a short-hand for `[^A-Za-z0-9_]` or `[^[:alnum:]_]`.

`\<` This operator matches the empty string at the beginning of a word. For example, `\<away` matches '`away`', but not '`stowaway`'.

`\>` This operator matches the empty string at the end of a word. For example, `stow\>` matches '`stow`', but not '`stowaway`'.

`\b` This operator matches the empty string at either the beginning or the end of a word (the word **b**oundary). For example, '`\bballs?\b`' matches either '`ball`' or '`balls`' as a separate word.

`\B` This operator matches the empty string within a word. In other words, '`\B`' matches the empty string that occurs between two word-constituent characters. For example, `\Brat\B` matches '`crate`', but it does not match '`dirty rat`'. '`\B`' is essentially the opposite of '`\b`'.

There are two other operators that work on buffers. In Emacs, a *buffer* is, naturally, an Emacs buffer. For other programs, the regexp library routines that Gcal uses consider the entire string to be matched as the buffer[5].

For Gcal, since '`^`' and '`$`' always work in terms of the beginning and end of strings, these operators do not add any new capabilities. They are provided for compatibility with other GNU software.

`` \` `` This operator matches the empty string at the beginning of the buffer.

`\'` This operator matches the empty string at the end of the buffer.

[5] That is the *text part* of a Gcal resource file line.

In regular expressions, the ‘*’, ‘+’, and ‘?’ operators, as well as the braces ‘{’ and ‘}’, have the highest precedence, followed by concatenation, and finally by ‘|’. As in arithmetic, parentheses can change how operators are grouped.

Case is normally significant in regular expressions, both when matching ordinary characters (i.e. not metacharacters), and inside character sets. Thus a ‘w’ in a regular expression matches only a lower-case ‘w’ and not an upper-case ‘W’.

The simplest way to do a case-independent match is to use a character list: ‘[Ww]’. However, this can be cumbersome if you need to use it often; and unfortunately, it can make the regular expressions harder to read. Supplying a want, Gcal offers the ‘--ignore-case’ option which ignores all case distinctions in both the regular expression and the completely expanded text of each **valid** fixed date. See [Fixed date option ‘--ignore-case’], page 41.

# Appendix F Summary of all Regular Expressions

Regular expressions are based on POSIX EREs (Extended Regular Expressions). Regexps are composed of characters. Here is a short-list of them all:

`c` matches the character *c* (assuming *c* is none of the characters listed below).

`\c` matches the literal character *c*.

`.` matches any character.

`^` matches the beginning of a string.

`$` matches the end of a string.

`[abc...]` matches any of the characters *abc*. . . (character list).

`[[:keyword:]]`
matches any character in the character class *keyword*. Allowable classes are `alnum`, `alpha`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, and `xdigit`.

`[[.element.]]`
matches the multi-character collating *element*. Gcal does not currently support collating symbols.

`[[=list=]]`
matches any of the equivalent characters in *list*. Gcal does not currently support equivalence classes.

`[^abc...]`
matches any character except *abc*. . . (negated resp. complemented character list).

`r1|r2` matches either *r1* or *r2* (alternation).

`r1r2` matches *r1*, and then *r2* (concatenation).

`r+` matches one or more *r*'s.

`r*` matches zero or more *r*'s.

`r?` matches zero or one *r*'s.

`(r)` matches *r* (grouping).

`r{n}`
`r{n,}`
`r{n,m}` matches at least *n*, *n* to any number, or *n* to *m* occurrences of *r* (interval expressions).

`\b` matches the empty string at either the beginning or the end of a word.

`\B` matches the empty string within a word.

`\<` matches the empty string at the beginning of a word.

`\>` matches the empty string at the end of a word.

`\w` matches any word-constituent character (alphanumeric characters and the underscore).

`\W` matches any character that is not word-constituent.

`` \` `` matches the empty string at the beginning of a buffer[1].

`\'` matches the empty string at the end of a buffer.

[1] The same as a *text part* of a resource file line in Gcal.

# Appendix G %?... Special Texts

The %?... special texts which can be used in the text part of a Gcal resource file line can coarsely be separated into three categories. So there are special texts used for suppression output of fixed dates in definite cases. Others are replaced by particular texts or cause the shell to start external commands. See Section 5.1.5 [%?... Special Texts], page 94, for a general description of the special texts which may be used in the text part of a Gcal resource file line.

Some of these special texts may have an optional format instruction (called *format* in the further context), which affects the representation of an expanded special text. See Appendix J [Format Instruction], page 231, for the detailed description of the format instruction and its components. A possibly specified *format* within special texts which may not have a format instruction is ignored by Gcal and not displayed in output.

But Gcal is also able to represent such special texts in a special way by using a format instruction, and that by using an indirect method. For displaying, you simply assign the special text exclusively to a text variable which contains a format instruction. Nevertheless, it is important to note that the field width component of the format instruction is chosen in the way that it refers to the assigned contents of the text variable, if the format instruction has a fixed format component. The contents assigned to the text variable is only the special text, and not its expanded result. Gcal has to proceed like that, because the special text which is hidden in the text variable could not be used (here: displayed) only for a definite date, but possibly also for several dates, and that adjusted to each date it is referenced. Thus exists a special text in the text variable which has a definite length, and a format instruction also exists, whose field width component has to affect the expanded result of the special text. Indeed, Gcal does not know at the moment when the assignment of the special text to the text variable is made, of which length the expanded result will be, because this expansion is made at a much later phase in the program execution. So, in such a case, the field width component of a format instruction should have the length of the special text minimum, otherwise it is crippled and expanded respectively displayed incorrectly by the internal methods which effect the expansion.

Of course, not all possibilities concerning the representation —which are offered by the format instruction— are listed here for those special texts those expanded representation may be affected by an optional *format* instruction, but only some few exemplary.

And some of these special texts may also have an optional simple or special date argument (called *date* in the further context) or another argument (called *argument* in the further context). The *date* argument must be given in one of the following date formats in case a special text is directly trailed by it:

- *yyyy*[*mm*[*dd*|*wwwn*]]
- *yyyy**d|*wn*[*www*]
- *yyyy*@e|t|*dvar*[[+|-]*n*[*www*]]

Special texts that may have a *date* argument **must** always be trailed by a *whitespace* character which is removed in output, no matter whether *date* was given or not!

See Appendix H [Summary of all %?... Special Texts], page 197, for a short-list of all applicable special texts.

## G.1 %?... special texts for date exclusion

%?... special texts used for exclusion of points of time or periods of dates are very useful if they are used together with ranges of days (see Section 5.1.3.2 [Ranges of days], page 85). The complete special text is always suppressed in output in case the fixed date text must be displayed.

### G.1.1 Exclusions with date argument %?[*date*] special texts

First of all, an *inclusive date* period %?[*date*] special text can be specified with the effect, that only those fixed dates are respected and displayed which are part of this period. And the specification of an *exclusive date* period %?[*date*] special text has the effect, that only those fixed dates are respected and displayed which are not part of this period.

Cleverly combined, these special texts offer very effective filtration capabilities used for the definition of special periods in a highly flexible manner, for example:

```
0*d1#999 Every day of year: from January...August except April\
         %i00001#00008 %e00004#00004
;
0*d1#999 Every day of year: from 1991...1993 and from 1996\
         except February 1992 and all September\
         %i1991#1993 %i1996# %e19922#19922 %e00009#00009
```

#### G.1.1.1 Inclusive date period %i[*date*][#[*date*]] special text

%i[*date*][#[*date*]] references an inclusive date period, i.e. all dates which are part of the specified period are valid and are displayed. Of course this means, that all fixed dates which are not part of this period (not covered by it) are invalid and remain undisplayed therefore.

The first specified date defines the *starting date* of the fixed date, i.e. the first date the fixed date may occur. For the starting date, the following rules concerning definite omission values are valid, elucidated by using the *yyyy*[*mm*[*dd*|*wwwn*]] date format:

- If *yyyy* is set to 0000, the current year is assumed for the year.
- If *mm* is not given, then 01 (≡January) is assumed for the month.
- If *mm* is set to 00, the current month is assumed for the month, and the December is assumed in case *mm* is set to 99.
- If *dd*|*www* is not given, then 01 is assumed for the day.
- If *dd* is set to 00, the current day is assumed for the day, and the last day of the month *mm* is assumed in case *dd* is set to 99.
- If the *n* field is given (range 1...5 or 9), then a *www* field must also be specified.

The second specified date defines the *ending date* of the fixed date, i.e. the last date the fixed date may occur. For the ending date, the following rules concerning definite omission values are valid, likewise elucidated by using the *yyyy*[*mm*[*dd*|*wwwn*]] date format:

- If *yyyy* is set to 0000, the current year is assumed for the year.
- If *mm* is not given, then 12 (≡December) is assumed for the month.
- If *mm* is set to 00, the current month is assumed for the month, and the December is assumed in case *mm* is set to 99.
- If *dd*|*www* is not given, the last day of the month *mm* is assumed for the day.
- If *dd* is set to 00, the current day is assumed for the day, and the last day of the month *mm* is assumed in case *dd* is set to 99.
- If the *n* field is given (range 1...5 or 9), then a *www* field must also be specified.

The preceding rules are analogously valid for the other permitted date formats.

If the starting date, which is encoded in the %i... special text, is later the ending date, the %i... special text will be ignored.

In case neither the starting date nor the ending date is specified, means only '%i' is given, the implicit period 1st January 1...31st December 9999 (00010101...99991231) will be used.

If only the starting date is specified, means either '%i*date*' or '%i*date*#' is given, the implicit ending date 31st December 9999 (99991231) will be used.

In case only the ending date is specified, means '%i#*date*' is given, the implicit starting date 1st January 1 (00010101) will be used.

For example:

```
0d*1#999 Every day of year: only April and July\
         %i00004#00004 %i00007#00007
;
0*d1#999 Every day of year: from January...October\
         %i#000010
;
0*d1#999 Every day of year: from August 1990\
         %i19908
;
0*d1#999 Every day of year: from August...December\
         %i00008
;
0*d1#999 Every day of year: only today%i0@t#0@t
```

#### G.1.1.2 Exclusive date period %e[*date*][#[*date*]] special text

%e[*date*][#[*date*]] references an exclusive date period, i.e. all dates which are part of the specified period are invalid and remain undisplayed therefore. Of course this means, that all fixed dates which are not part of this period (not covered by it) are valid and are displayed.

The first specified date defines the starting date of the fixed date, i.e. the first date the fixed date may occur. For the starting date, the following rules concerning definite omission values are valid, elucidated by using the *yyyy*[*mm*[*dd*|*wwwn*]] date format:

- If *yyyy* is set to 0000, the current year is assumed for the year.
- If *mm* is not given, then 01 (≡January) is assumed for the month.
- If *mm* is set to 00, the current month is assumed for the month, and the December is assumed in case *mm* is set to 99.
- If *dd*|*www* is not given, then 01 is assumed for the day.
- If *dd* is set to 00, the current day is assumed for the day, and the last day of the month *mm* is assumed in case *dd* is set to 99.
- If the *n* field is given (range 1...5 or 9), then a *www* field must also be specified.

The second specified date defines the ending date of the fixed date, i.e. the last date the fixed date may occur. For the ending date, the following rules concerning definite omission values are valid, likewise elucidated by using the *yyyy*[*mm*[*dd*|*wwwn*]] date format:

- If *yyyy* is set to 0000, the current year is assumed for the year.
- If *mm* is not given, then 12 (≡December) is assumed for the month.
- If *mm* is set to 00, the current month is assumed for the month, and the December is assumed in case *mm* is set to 99.
- If *dd*|*www* is not given, the last day of the month *mm* is assumed for the day.
- If *dd* is set to 00, the current day is assumed for the day, and the last day of the month *mm* is assumed in case *dd* is set to 99.
- If the *n* field is given (range 1...5 or 9), then a *www* field must also be specified.

The preceding rules are analogously valid for the other permitted date formats.

If the starting date, which is encoded in the %e... special text, is later the ending date, the %e... special text will be ignored.

In case neither the starting date nor the ending date is specified, means only '%e' is given, the implicit period 1st January 1...31st December 9999 (00010101...99991231) will be used.

If only the starting date is specified, means either '%e*date*' or '%e*date*#' is given, the implicit ending date 31st December 9999 (99991231) will be used.

In case only the ending date is specified, means '%e#*date*' is given, the implicit starting date 1st January 1 (00010101) will be used.

For example:

```
0d*1#999 Every day of year: except April and July\
        %e00004#00004 %e00007#00007
;
0*d1#999 Every day of year: except January...October\
         %e#000010
;
0*d1#999 Every day of year: until July 1990\
         %e19908
;
0*d1#999 Every day of year: except August...December\
         %e00008
;
0*d1#999 Every day of year: except today%e0@t#0@t
```

### G.1.2 Exclusions without any argument %? special texts

First of all, an *inclusive day* period %? special text can be specified with the effect, that only those fixed dates are respected and displayed which are part of this period. And the specification of an *exclusive day* period %? special text has the effect, that only those fixed dates are respected and displayed which are not part of this period.

Cleverly combined, these special texts also offer very effective filtration capabilities used for the definition of very special periods.

For example:

```
0*d1#999 Every day of year: all Mondays...Fridays\
         except Wednesdays and all legal holidays\
         %r%F%V
;
00001001#9999 Every day in last quarter of year: all holidays\
              which are not on a Sunday%x %P
```

#### G.1.2.1 Inclusive day period %? special texts

The following inclusive day periods %? special texts are respected, at which *inclusive* is meant for the period which is not excluded.

%v Excludes fixed date in case it is not listed as legal holiday in the *eternal holiday list*.

%x Excludes fixed date in case it is neither listed as legal holiday nor memorial day in the *eternal holiday list*.

%a Excludes fixed date in case it is no Monday.

%c Excludes fixed date in case it is no Tuesday.

%f Excludes fixed date in case it is no Wednesday.

%g Excludes fixed date in case it is no Thursday.

%h Excludes fixed date in case it is no Friday.

%l Excludes fixed date in case it is no Saturday.

`%p` Excludes fixed date in case it is no Sunday.

`%q` Excludes fixed date in case it is no Monday...Thursday.

`%r` Excludes fixed date in case it is no Monday...Friday.

#### G.1.2.2 Exclusive day period `%?` special texts

The following exclusive day periods `%?` special texts are respected, at which *exclusive* is meant for the period which is excluded.

`%V` Excludes fixed date in case it is listed as legal holiday in the *eternal holiday list*.

`%X` Excludes fixed date in case it is either listed as legal holiday or memorial day in the *eternal holiday list*.

`%A` Excludes fixed date in case it is a Monday.

`%C` Excludes fixed date in case it is a Tuesday.

`%F` Excludes fixed date in case it is a Wednesday.

`%G` Excludes fixed date in case it is a Thursday.

`%H` Excludes fixed date in case it is a Friday.

`%L` Excludes fixed date in case it is a Saturday.

`%P` Excludes fixed date in case it is a Sunday.

`%Q` Excludes fixed date in case it is a Monday...Thursday.

`%R` Excludes fixed date in case it is a Monday...Friday.

## G.2 `%?`... special texts for text replacement

`%?`... special texts which are replaced by particular texts at program run-time causes the interspersing of data into the fixed date text, which quality is essentially depending on the command line arguments given for running the program. The complete special text is always replaced accordingly in output if the fixed date text must be displayed.

Cleverly combined, these special texts also offer very effective mechanisms used for the definition of particular texts in a highly flexible manner, for example:

```
    0*d1#999 Every day of year: Today is %K, the %n (%N)
```

### G.2.1 Replacements with date argument `%?`[*date*] special texts

Apart from different representations of a weekday name, it is possible to create its weekday number. Likewise, the current day-of-year number, the day number, the week number, the month name, the month number, the year number, the Moon phase, the biorhythm, and some useful difference values can be produced.

For the *date* argument these `%?`[*date*] special text may have, the following rules concerning definite omission values are valid, elucidated by using the *yyyy*[*mm*[*dd*|*wwwn*]] date format[1]:

- If *yyyy*... is not given (≡**no** date encoded), the current year is assumed for the date.
- If *yyyy* is set to `0000`, the current year is assumed for the year.
- If *mm* is not given or set to `00`, the current month is assumed for the month, and the December is assumed in case *mm* is set to `99`.
- If *dd*|*www* is not given or set to `00`, the current day is assumed for the day, and the last day of the month is assumed in case *dd* is set to `99`.
- If the *n* field is given (range `1...5` or `9`), then a *www* field must also be specified.

[1] The succeeding rules are analogously valid for the other permitted date formats.

### G.2.1.1 Weekday name `%[format]K[date]` special text

`%K[date]` Is replaced by the complete **weekday name** of the current day, e.g.:

> The text '`Today is %K , the %n`' will be expanded to
> ↦ '`Today is Monday, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%>3#K[date]`
Is replaced by the 3-letter **weekday name** of the current day, e.g.:

> The text '`Today is %>3#K0@t+1 , the %n+1`' will be expanded to
> ↦ '`Today is Tue, the 11-Jun-1996`', in case the actual system date is the 10th June 1996.

`%>2#K[date]`
Is replaced by the 2-letter **weekday name** of the current day, e.g.:

> The text '`Today is %>2#K , the %n`' will be expanded to
> ↦ '`Today is Mo, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

### G.2.1.2 Weekday number `%[format]?[date]` special texts

The following weekday number `%[format]?[date]` special texts are respected, e.g.:

`%W[date]` Is replaced by the **weekday number** of the current day, and which is determined by the *Monday≡1. . . Sunday≡7* counting method, e.g.:

> The text '`Today is weekday %W , the %n`' will be expanded to
> ↦ '`Today is weekday 1, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%>1&*W[date]`
Is replaced by the **weekday number** with trailing Ordinal Number suffix of the current day, and which is determined by the *Monday≡1. . . Sunday≡7* counting method, e.g.:

> The text '`Today is weekday %>1&*W0@t+1 , the %n+1`' will be expanded to
> ↦ '`Today is weekday 2nd, the 11-Jun-1996`', in case the actual system date is the 10th June 1996.

`%E[date]` Is replaced by the **weekday number** of the current day, and which is determined by the *Monday≡0. . . Sunday≡6* counting method, e.g.:

> The text '`Today is weekday %E , the %n`' will be expanded to
> ↦ '`Today is weekday 0, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%>1&*E[date]`
Is replaced by the **weekday number** with trailing Ordinal Number suffix of the current day, and which is determined by the *Monday≡0. . . Sunday≡6* counting method, e.g.:

> The text '`Today is weekday %>1&*E , the %n`' will be expanded to
> ↦ '`Today is weekday 0th, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%I[date]` Is replaced by the **weekday number** of the current day, and which is determined by the *Sunday≡1. . . Saturday≡7* counting method, e.g.:

> The text '`Today is weekday %I , the %n`' will be expanded to
> ↦ '`Today is weekday 2, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%>1&*I`[*date*]
Is replaced by the **weekday number** with trailing Ordinal Number suffix of the current day, and which is determined by the *Sunday*≡*1*. . .*Saturday*≡*7* counting method, e.g.:

> The text '`Today is weekday %>1&*I , the %n`' will be expanded to ↦ '`Today is weekday 2nd, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%J`[*date*] Is replaced by the **weekday number** of the current day, and which is determined by the *Sunday*≡*0*. . .*Saturday*≡*6* counting method, e.g.:

> The text '`Today is weekday %J , the %n`' will be expanded to ↦ '`Today is weekday 1, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%>1&*J`[*date*]
Is replaced by the **weekday number** with trailing Ordinal Number suffix of the current day, and which is determined by the *Sunday*≡*0*. . .*Saturday*≡*6* counting method, e.g.:

> The text '`Today is weekday %>1&*J , the %n`' will be expanded to ↦ '`Today is weekday 1st, the 10-Jun-1996`', in case the actual system date is the 10th June 1996.

`%S`[*date*] Is replaced by the **weekday number** of the current day, and which is determined by the *starting day of week*≡*1*. . .*ending day of week*≡*7* counting method (see [Calendar option '`--starting-day=`*`argument`*'], page 30), e.g.:

> The text '`Today is weekday %S , the %n`' will be expanded to ↦ '`Today is weekday 4, the 10-Jun-1996`', in case the actual system date is the 10th June 1996 and Friday (≡5) is the starting day of week.

`%>1&*S`[*date*]
Is replaced by the **weekday number** with trailing Ordinal Number suffix of the current day, and which is determined by the *starting day of week*≡*1*. . .*ending day of week*≡*7* counting method (see [Calendar option '`--starting-day=`*`argument`*'], page 30), e.g.:

> The text '`Today is weekday %>1&*S , the %n`' will be expanded to ↦ '`Today is weekday 4th, the 10-Jun-1996`', in case the actual system date is the 10th June 1996 and Friday (≡5) is the starting day of week.

`%T`[*date*] Is replaced by the **weekday number** of the current day, and which is determined by the *starting day of week*≡*0*. . .*ending day of week*≡*6* counting method (see [Calendar option '`--starting-day=`*`argument`*'], page 30), e.g.:

> The text '`Today is weekday %T , the %n`' will be expanded to ↦ '`Today is weekday 3, the 10-Jun-1996`', in case the actual system date is the 10th June 1996 and Friday (≡5) is the starting day of week.

`%>1&*T`[*date*]
Is replaced by the **weekday number** with trailing Ordinal Number suffix of the current day, and which is determined by the *starting day of week* ≡ *0*. . . *ending day of week* ≡ *6* counting method (see [Calendar option '`--starting-day=`*argument*'], page 30), e.g.:

> The text '`Today is weekday %>1&*T , the %n`' will be expanded to
> ↦ '`Today is weekday 3rd, the 10-Jun-1996`', in case the actual system date is the 10th June 1996 and Friday (≡5) is the starting day of week.

#### G.2.1.3 Day-of-year number `%`[*format*]`N`[*date*] special text

`%N`[*date*] Is replaced by the **day-of-year number** of the current day of year, e.g.:

> The text '`Day %N  of year, the %n`' will be expanded to
> ↦ '`Day 53 of year, the 22-Feb-1996`', in case the actual system date is the 22nd February 1996.

`%>03*N`[*date*]
Is replaced by the 3-digit **day-of-year number** with leading zeroes of the current day of year, e.g.:

> The text '`Day %>03*N0@t-1  of year, the %n-1`' will be expanded to
> ↦ '`Day 052 of year, the 21-Feb-1996`', in case the actual system date is the 22nd February 1996.

`%>1&*N`[*date*]
Is replaced by the **day-of-year number** with trailing Ordinal Number suffix of the current day of year, e.g.:

> The text '`Day %>1&*N  of year, the %n`' will be expanded to
> ↦ '`Day 53rd of year, the 22-Feb-1996`', in case the actual system date is the 22nd February 1996.

`%>03&*N`[*date*]
Is replaced by the 3-digit **day-of-year number** with leading zeroes and trailing Ordinal Number suffix of the current day of year, e.g.:

> The text '`Day %>03&*N  of year, the %n`' will be expanded to
> ↦ '`Day 053rd of year, the 22-Feb-1996`', in case the actual system date is the 22nd February 1996.

#### G.2.1.4 Day number `%`[*format*]`D`[*date*] special text

`%D`[*date*] Is replaced by the **day number** of the current day of month, e.g.:

> The text '`Day %D , %n`' will be expanded to
> ↦ '`Day 2, 02-Feb-1996`', in case the actual system date is the 2nd February 1996.

`%>02*D`[*date*]

Is replaced by the 2-digit **day number** with leading zero of the current day of month, e.g.:

> The text ‘`Day %>02*D0@t-1 , %n-1`’ will be expanded to
> ↦ ‘`Day 01, 01-Feb-1996`’, in case the actual system date is the 2nd February 1996.

`%>1&*D`[*date*]

Is replaced by the **day number** with trailing Ordinal Number suffix of the current day of month, e.g.:

> The text ‘`Day %>1&*D , %n`’ will be expanded to
> ↦ ‘`Day 2nd, 02-Feb-1996`’, in case the actual system date is the 2nd February 1996.

`%>02&*D`[*date*]

Is replaced by the 2-digit **day number** with leading zero and trailing Ordinal Number suffix of the current day of month, e.g.:

> The text ‘`Day %>02&*D , %n`’ will be expanded to
> ↦ ‘`Day 02nd, 02-Feb-1996`’, in case the actual system date is the 2nd February 1996.

### G.2.1.5 Week number `%k`[*date*] special text

`%k`[*date*] Is replaced by either the 2-digit **ISO week number** or the **standard week number** with leading zero, or a 4-alphanumeric character text of the current week of year (see [Calendar option ‘`--starting-day=`*argument*’], page 30, [Calendar option ‘`--iso-week-number=yes|no`’], page 29, and Appendix C [Aspects in Internationalization], page 123, for further details), e.g.:

> The text ‘`Today is %K  of week %k`’ will be expanded to
> ↦ ‘`Today is Monday of week 24`’, in case the actual system date is the 10th June 1996.

### G.2.1.6 Month name `%`[*format*]`U`[*date*] special text

`%U`[*date*] Is replaced by the complete **month name** of the current month of year, e.g.:

> The text ‘`Today is %U , %n`’ will be expanded to
> ↦ ‘`Today is June, 10-Jun-1996`’, in case the actual system date is the 10th June 1996.

`%>3#U`[*date*]

Is replaced by the 3-letter **month name** of the current month of year, e.g.:

> The text ‘`Today is %>3#U19961010 , %n`’ will be expanded to
> ↦ ‘`Today is Oct, 10-Jun-1996`’, in case the actual system date is the 10th June 1996.

### G.2.1.7 Month number `%`[*format*]`M`[*date*] special text

`%M`[*date*] Is replaced by the **month number** of the current month of year, e.g.:

> The text ‘`Month %M , %n`’ will be expanded to
> ↦ ‘`Month 2, 22-Feb-1996`’, in case the actual system date is the 22nd February 1996.

`%>02*M`[*date*]

Is replaced by the 2-digit **month number** with leading zero of the current month of year, e.g.:

> The text '`Month %>02*M000001 , %n`' will be expanded to
> ↦ '`Month 01, 22-Feb-1996`', in case the actual system date is the 22nd February 1996.

`%>1&*M`[*date*]
: Is replaced by the **month number** with trailing Ordinal Number suffix of the current month of year, e.g.:

> The text '`Month %>1&*M , %n`' will be expanded to
> ↦ '`Month 2nd, 22-Feb-1996`', in case the actual system date is the 22nd February 1996.

`%>02&*M`[*date*]
: Is replaced by the 2-digit **month number** with leading zero and trailing Ordinal Number suffix of the current month of year, e.g.:

> The text '`Month %>02&*M , %n`' will be expanded to
> ↦ '`Month 02nd, 22-Feb-1996`', in case the actual system date is the 22nd February 1996.

#### G.2.1.8 Year number %[*format*]Y[*date*] special text

`%Y`[*date*] Is replaced by the **year number** of the current year, e.g.:

> The text '`Year %Y , %n`' will be expanded to
> ↦ '`Year 933, 22-Feb-0933`', in case the actual system date is the 22nd February 933.

`%>04*Y`[*date*]
: Is replaced by the 4-digit **complete year number** with leading zeroes of the current year, e.g.:

> The text '`Year %>04*Y0015 , %n`' will be expanded to
> ↦ '`Year 0015, 22-Feb-0933`', in case the actual system date is the 22nd February 933.

#### G.2.1.9 Moon phase %[*format*]?[*date*] special texts

Gcal uses a very simple algorithm for detecting the Moon phase, which computes approximate values only. The Moon phase is always calculated for 0 o'clock Universal time (UTC/GMT). See [Calendar option '`--time-offset=`*argument*'], page 31, how to change the base time for which the astronomical functions are calculated. The following Moon phase `%`[*format*]*?*[*date*] special texts are respected:

`%O`[*date*] Is replaced by the **Moon phase text** of the current day of year, e.g.:

> The text '`Moon phase %O , %n`' will be expanded to
> ↦ '`Moon phase 94%+, 22-Dec-1996`', in case the actual system date is the 22nd December 1996.

The constructed Moon phase text consists of a percent value, which informs about the illuminated fraction of the Moon's disk, and a trailing sign that tells something about the state of the Moon. A trailing '`+`' sign indicates a waxing Moon, a '`-`' sign a waning Moon, a '`@`' sign indicates the Full Moon phase, and a '`!`' sign the New Moon phase.

`%>03*O`[*date*]
: Is replaced by the 3-digit **Moon phase text** with leading zeroes of the current day of year, e.g.:

> The text '`Moon phase %>03*O0@t+1 , %n+1`' will be expanded to
> ↦ '`Moon phase 098%+, 23-Dec-1996`', in case the actual system date is the 22nd December 1996.

The constructed Moon phase text consists of a percent value, which informs about the illuminated fraction of the Moon's disk, and a trailing sign that tells something about the state of the Moon. A trailing '`+`' sign indicates a waxing Moon, a '`-`' sign a waning Moon, a '`@`' sign indicates the Full Moon phase, and a '`!`' sign the New Moon phase.

`%Z[`*`date`*`]` Is replaced by the **Moon phase text graphics** of the current day of year, e.g.:

The text '`Date %n %Z  --- Moon phase %>03*O ~Text`' will be expanded to

```
↦ ‘Date 08-Mar-1995 ’
↦ ‘           (       @@@@@@’
↦ ‘      (               @@@@@@@@@@’
↦ ‘    (                 @@@@@@@@@@@@’
↦ ‘  (                    @@@@@@@@@@@@@’
↦ ‘ (                     @@@@@@@@@@@@@@’
↦ ‘(                       @@@@@@@@@@@@@@’
↦ ‘(                       @@@@@@@@@@@@@@’
↦ ‘ (                     @@@@@@@@@@@@@@’
↦ ‘  (                    @@@@@@@@@@@@@’
↦ ‘   (                   @@@@@@@@@@@@’
↦ ‘      (              @@@@@@@@@@’
↦ ‘           (         @@@@@@@ --- Moon phase 041%+’
```

↦ '`Text`', in case the actual system date is the 8th March 1995.

See [Fixed date option '`--moonimage-lines=`*`number`*'], page 45, how to change the size of a Moon phase text graphics.

A possibly specified *format* within this special text is ignored by Gcal.

#### G.2.1.10 Biorhythm `%?[`*`date`*`]` special texts

Gcal is able to create a *biorhythm* for any specified date of birth. The created text shows the three standard biorhythm cycles, and that are the 28 day emotional cycle, the 33 day intellectual cycle, and the 23 day physical cycle. The emotional cycle governs sensibility, nerves, moodiness, and creative ability. The intellectual cycle reflects intelligence, memory, mental alertness, and reasoning power. The physical cycle represents physical strength, endurance, energy and resistance. All cycles start in the zero point at the date of birth and swing like sine curves between their positive and negative maximum values. The periods above the zero point show the days of full vitality and efficiency while the periods below the zero point indicate days of reduced efficiency. The biorhythm is implemented for entertaining purposes only! It is up to the user to interpret the biorhythm texts.

Gcal counts *critical* days, *positive* likewise *negative* days. Critical days are those days in which one or more of the biological cycles crosses the zero point. At that time, one's system is said to be in a state of flux and it may be desirable to exhibit caution. Positive days are those days in which one or more of the biological cycles have a positive maximum value. At that time, one's system is said to be in a raised state. Negative days are those days in which one or more of the biological cycles have a negative maximum value. At that time, one's system is said to be in a lessen state.

The following biorhythm `%?[`*`date`*`]` special texts are respected:

`%,[`*`date`*`]` Is replaced by the **biorhythm text** of the current day of year, e.g.:

The text '`%,19620921`' will be expanded to
↦ '`1! 0+ 0- , Emo=+022%- Int=-091%+ Phy=-014%-`', in case the actual system date is the 12th December 1996.

The preceding example shows the calculated biorhythm expressed as a series of values for a person born in 1962, September 21st. The constructed text consists of two parts, and that is on the one hand a triplet of total values which indicate the critical, positive and negative day. And on the other hand a triplet of values which indicate the emotional, intellectual and physical cycle.

In the first triplet of total values, a ‘`!`’ suffix means a critical day, a ‘`+`’ indicates a positive day, and a ‘`-`’ suffix marks a negative day.

In the second triplet of values, each single cycle percent value consists of a leading positive or negative sign that tells something about the distance of this value to the zero point, and a trailing character that tells something about the state of each cycle. A trailing ‘`+`’ character marks a waxing phase, a ‘`-`’ suffix means a waning phase, and a ‘`@`’ indicates the maximum value of a phase.

`%;`[*`date`*] Is replaced by the **biorhythm text graphics** of the current day of year, e.g.:

> The text ‘`%;19620921`’ will be expanded to
> ↦ ‘`0-  I                P 1   E                 +0`’, in case the actual system date is the 12th December 1996.

The preceding example shows the calculated biorhythm expressed as a text graphics line for a person born in 1962, September 21st. The constructed line is a bar with a negative and positive axis of adjustable length (see [Fixed date option ‘`--biorhythm-axis=`*`number`*’], page 45), which left margin is represented by the total value of the negative days (the ‘`0-`’ in this case), and which right margin is represented by the total value of the positive days (the ‘`+0`’ in this case). The zero point of the co-ordinate is represented by the total value of the critical days, which is the ‘`1`’ in this case. The emotional, intellectual and the physical phase value is accordingly placed in scaled manner on this bar, and that by using the initial letter of the cycle in each case (‘`E`’, ‘`I`’ and ‘`P`’). Coincidental phase values are marked by a ‘`@`’ character.

### G.2.1.11 Difference value `%`[*`format`*]`?`[*`date`*] special texts

The following difference value `%`[*`format`*]`?`[*`date`*] special texts are respected:

`%y`*`date`* Is replaced by a **year difference value**, e.g.:

> The text ‘`Sylvester 1912 is %y1912 years ago`’ will be expanded to
> ↦ ‘`Sylvester 1912 is -82 years ago`’, in case the year of the actual system date is 1994.

`%B`[*`date`*] Is replaced by an **age value**, e.g.:

> The text ‘`My %B1962 birthday%i1952#2062`’ will be expanded to
> ↦ ‘`My 32 birthday`’, in case the year of the actual system date is 1994.

Those age values are only displayed, if the computation of an age value is greater zero. The fixed date warning in preceding example is displayed only in case the current year is greater than 1952 and less than 2062.

`%>1&*B`[*`date`*]

Is replaced by an **age value** with trailing Ordinal Number suffix, e.g.:

> The text ‘`My %>1&*B1962 birthday%i1952#2062`’ will be expanded to
> ↦ ‘`My 32nd birthday`’, in case the year of the actual system date is 1994.

Those age values are only displayed, if the computation of an age value is greater zero. The fixed date warning in preceding example is displayed only in case the current year is greater than 1952 and less than 2062.

### G.2.2 Replacements with other argument %[*format*]?[*argument*] special texts

Apart from different representations of the actual system time, a fixed format date text can be created. Moreover, it is possible to produce a day number, which bears the Julian date as base date and which can be deferred if needed. Furthermore, the approximate distance and the course angle between two geographic point locations, and different Sun and Moon oriented data and times for at pleasure any geographical location can be created. It is also possible to display the contents of environment variables in the fixed date text.

For the *argument* these %?[*argument*] special texts may have, no special rules concerning definite omission values are valid.

#### G.2.2.1 Actual clocktime %[*format*]?[*argument*] special texts

Gcal is able to represent the actual clocktime as local/zone time and as Universal time (UTC/GMT), and additionally the RFC-822[2] style numerical Universal time timezone offset value and the actual numerical local time timezone offset value. See [Actual local time in *hh*:*mm* format %t[*argument*] special text], page 149, for the detailed description of all components of the *argument*, which may trail the actual clocktime %[*format*]?[*argument*] special texts. All actual clocktime %[*format*]?[*argument*] special texts **must** always be trailed by a *whitespace* character which is removed in output! The following actual clocktime %[*format*]?[*argument*] special texts are respected:

%t[*argument*]

%t[*][[+|-]*mmmm*|*hh*:[*mm*]] references the **actual local time**, which is displayed by using the *hh*:*mm* output format, e.g.:

> The text ‘`Local time: %t  --> Dentist at 10:00`’ will be expanded to ↦ ‘`Local time: 07:32 --> Dentist at 10:00`’, in case the actual local time value is 07:32 o’clock (*hh*:*mm*).

If only %t is specified, the local time value will implicitly be displayed by using the *24-hours* format. For displaying using the *12-hours* format, add a ‘*’ character directly behind %t, e.g. ‘%t*’.

A displacement value may trail the %t special text, which has to be specified either by using the [+|-]*mmmm* format or the [+|-]*hh*:[*mm*] format. [+|-]*mmmm* adds respectively subtracts the specified amount of minutes *mmmm* from the local time value (range 0...9999), while [+|-]*hh*:[*mm*] adds respectively subtracts the given amount of hours *hh* (range 0...99) and minutes *mm* (range 0...59) from the local time value. The displacement value is always added to the local time value in case it is specified without a +|- sign. In case Gcal is unable to compute the actual local time in *hh*:*mm* format by reason of a misspecified *argument*, a ??:?? text will be created instead of the actual local time in *hh*:*mm* format.

> The text ‘`Local time: %t-3:  --> Dentist at 10:00`’ will be expanded to ↦ ‘`Local time: 04:32 --> Dentist at 10:00`’, in case the actual local time value is 07:32 o’clock (*hh*:*mm*).

%[*format*]’[*argument*]

%[*format*]’[[+|-]*mmmm*|*hh*:[*mm*]] references the **actual local time in minutes since midnight**, which is displayed by using the *m*[*m*...] output format, e.g.:

> The text ‘`Local time: %’  --> Dentist at 10:00`’ will be expanded to ↦ ‘`Local time: 452 --> Dentist at 10:00`’, in case the actual local time value is 07:32 o’clock (*hh*:*mm*).

[2] RFC is the abbreviation of *Request for Comments*.

In case Gcal is unable to compute the actual local time in *m*[*m*. . .] format by reason of a misspecified *argument*, a `??` text will be created instead of the actual local time in *m*[*m*. . .] format.

`%[`*`format`*`]_[`*`argument`*`]`

`%[`*`format`*`]_[*][[+|-]`*mmmm*`|`*`hh`*`:[`*mm*`]]` references the **actual local time hour**, which is displayed by using the *h*[*h*. . .] output format, e.g.:

> The text '`Local time hour: %_-1:`' will be expanded to
> ↦ '`Local time hour: 6`' in case the actual local time value is 07:32 o'clock (*`hh`*`:`*`mm`*).

In case Gcal is unable to compute the actual local time hour by reason of a misspecified *argument*, a `??` text will be created instead of the actual local time hour *h*[*h*. . .].

`%[`*`format`*`]?[`*`argument`*`]`

`%[`*`format`*`]?[[+|-]`*mmmm*`|`*`hh`*`:[`*mm*`]]` references the **actual local time minute**, which is displayed by using the *m*[*m*. . .] output format, e.g.:

> The text '`Local time minute: %>04*?+:2`' will be expanded to
> ↦ '`Local time minute: 0034`' in case the actual local time value is 07:32 o'clock (*`hh`*`:`*`mm`*).

In case Gcal is unable to compute the actual local time minute by reason of a misspecified *argument*, a `??` text will be created instead of the actual local time minute *m*[*m*. . .].

`%[`*`format`*`]{[`*`argument`*`]`

`%[`*`format`*`]{[[+|-]`*mmmm*`|`*`hh`*`:[`*mm*`]]` references the **actual local time 12-hour format suffix**, which is displayed as `am|pm`, e.g.:

> The text '`Local time suffix: %{`' will be expanded to
> ↦ '`Local time suffix: am`' in case the actual local time value is 07:32 o'clock (*`hh`*`:`*`mm`*).

In case Gcal is unable to compute the actual local time 12-hour format suffix by reason of a misspecified *argument*, a `??` text will be created instead of the actual local time 12-hour format suffix `am|pm`.

`%@[`*`argument`*`]`

`%@[*][[+|-]`*mmmm*`|`*`hh`*`:[`*mm*`]]` references the **actual Universal time** (UTC/GMT), which is displayed by using the *`hh`*`:`*`mm`*`/`*`date`* output format, e.g.:

> The text '`Universal time: %@*-3.`' will be expanded to
> ↦ '`Universal time: 03:32am/15-Feb-1999`', in case the actual date is the 15th February 1999 and the actual Universal time value is 06:32 o'clock (*`hh`*`:`*`mm`*).

In case Gcal is unable to compute the actual Universal time in *`hh`*`:`*`mm`*`/`*`date`* format by reason of a misspecified *argument*, a `??:??/??-???-????` text will be created instead of the actual Universal time in *`hh`*`:`*`mm`*`/`*`date`* format.

%[*format*]`[*argument*]

%[*format*]`[[+|-]*mmmm*|*hh*:[*mm*]] references the **actual Universal time in minutes since midnight**, which is displayed by using the *m*[*m*...]/*date* output format, e.g.:

> The text 'Universal time: %>06*`-3:' will be expanded to
> ↦ 'Universal time: 000212/15-Feb-1999' in case the actual date is the 15th February 1999 and the actual Universal time value is 06:32 o'clock (*hh*:*mm*).

In case Gcal is unable to compute the actual Universal time in *m*[*m*...]/*date* format by reason of a misspecified *argument*, a ??/??-???-???? text will be created instead of the actual Universal time in *m*[*m*...]/*date* format.

%[*format*].[*argument*]

%[*format*].[*][[+|-]*mmmm*|*hh*:[*mm*]] references the **actual Universal time hour**, which is displayed by using the *h*[*h*...] output format, e.g.:

> The text 'Universal time hour: %.-1:' will be expanded to
> ↦ 'Universal time hour: 5' in case the actual Universal time value is 06:32 o'clock (*hh*:*mm*).

In case Gcal is unable to compute the actual Universal time hour by reason of a misspecified *argument*, a ?? text will be created instead of the actual Universal time hour *h*[*h*...].

%[*format*]/[*argument*]

%[*format*]/[[+|-]*mmmm*|*hh*:[*mm*]] references the **actual Universal time minute**, which is displayed by using the *m*[*m*...] output format, e.g.:

> The text 'Universal time minute: %>04*/+:2' will be expanded to
> ↦ 'Universal time minute: 0034' in case the actual Universal time value is 06:32 o'clock (*hh*:*mm*).

In case Gcal is unable to compute the actual Universal time minute by reason of a misspecified *argument*, a ?? text will be created instead of the actual Universal time minute *m*[*m*...].

%[*format*]}[*argument*]

%[*format*]}[[+|-]*mmmm*|*hh*:[*mm*]] references the **actual Universal time 12-hour format suffix**, which is displayed as am|pm, e.g.:

> The text 'Universal time suffix: %>1w*}+10:' will be expanded to
> ↦ 'Universal time suffix: Pm' in case the actual Universal time value is 06:32 o'clock (*hh*:*mm*).

In case Gcal is unable to compute the actual Universal time 12-hour format suffix by reason of a misspecified *argument*, a ?? text will be created instead of the actual Universal time 12-hour format suffix am|pm.

%"[*argument*]

%"[[+|-]*mmmm*|*hh*:[*mm*]] references the RFC-822 style **numerical Universal time timezone offset value**, which is displayed by using the +|-*hhmm* output format, e.g.:

> The text 'Universal time timezone offset value: %"-90' will be expanded to
> ↦ 'Universal time timezone offset value: -0130', at which calculations are always based on the timezone of the Universal time UTC/GMT.

In case Gcal is unable to compute the numerical Universal time timezone offset value by reason of a misspecified *argument*, a +???? text will be created instead of the numerical Universal time timezone offset value +|-*hhmm*.

`%=`[*argument*]

> `%=`[[`+`|`-`]*mmmm*|*hh*`:`[*mm*]] references the RFC-822 style **actual numerical local time timezone offset value**, which is displayed by using the `+`|`-`*hhmm* output format, e.g.:
>
> > The text '`Local time timezone offset value CET: %=-10`' will be expanded to
> > ↦ '`Local time timezone offset value CET: +0050`', in case the actual timezone is equal the Central European (winter)time CEWT/CET (`+`*hhmm*). Calculations are always based on the timezone of the local time.
>
> In case Gcal is unable to compute the actual numerical local time timezone offset value by reason of a misspecified *argument*, a `+????` text will be created instead of the actual numerical local time timezone offset value `+`|`-`*hhmm*.

#### G.2.2.2 Textual date `%n`[*argument*] special text

`%n`[[`+`|`-`]*n*] references the current respectively queried **day number** relative to the current date ± *n* days and is replaced by a **date text** by using the fixed '`%>02*D-%>3#U-%>04*Y`' format, e.g.:

> The resource file line '`1962Sep21 10000 days old: %n+10000`' will be expanded to
> ↦ '`10000 days old: 06-Feb-1990`', in case you call Gcal with the simple '`-c`' option and the command '`sep 1962`' (see [Single command '*mm yyyy*'], page 68).

#### G.2.2.3 Julian day number `%`[*format*]`j`[*argument*] special text

`%`[*format*]`j`[[`+`|`-`]*n*] references the current respectively queried **Julian day number** relative to the actual system date (≡today). This day number is based on the date 1st January 4713 BCE — which is the starting day zero of a consecutive day counting used in astronomical computations— and is known as the Julian Date (J.D.). The real zero of this date is at 12 o'clock Universal time (UTC/GMT); the day does not change at midnight, but at noon Universal time. Here, Gcal does not evaluate the timezone returned by the systems date function. For that reason, this day number is represented without a time fraction on the supposition that the day has already changed at noon. If you do not like the feature that the day displayed has already changed at noon, you can decrease the resulting Julian day number of that special text always by one, e.g. '`%j-1`'.

For example:

> The resource file line '`0 Julian day %j since 01-Jan-4713 BCE`' will be expanded to
> ↦ '`Julian day 2437929 since 01-Jan-4713 BCE`', in case you call Gcal with the '`-c %19620921`' option and no command.

If the `%`[*format*]`j` text is directly trailed by an unsigned number, this number is always subtracted from the real Julian day number. So you are able to work with any quantities referenced, e.g.:

> The resource file line '`0 Julian day %j2415021 since 01-Jan-1900`' will be expanded to
> ↦ '`Julian day 4 since 01-Jan-1900`', in case you call Gcal with the '`-c %19000105`' option and no command.

#### G.2.2.4 Geographical distance and course angle `%`[*format*]`b`*argument* special text

`%`[*format*]`b`[`*`][*mode*]*ISO-6709:1983-co-ordinate-1*`/`*ISO-6709:1983-co-ordinate-2* references either the approximate **air line distance** or the approximate **course angle (true**

**track)**[3] between any of two geographic point locations. The selection, which value has to be calculated by using this special text is done by specifying the *mode* part of the preceding argument. Actually, exactly three different modes can be used that are represented by the ‘`0`’...‘`2`’ characters:

**Mode** **Description**

`0` Calculates the air line distance between *ISO-6709:1983-co-ordinate-1* and *ISO-6709:1983-co-ordinate-2*. The calculated air line distance value is displayed in kilometers by default. A ‘`*`’ character directly before this mode character causes Gcal to display the distance value for another quantity, and that in *statute miles*; where one statute mile is equal to 1.609344 kilometer. If Gcal is unable to compute the approximate distance between the two geographic point locations by reason of a misspecified *argument*, a `??` text will be created instead of the distance value.

`1` Calculates the course angle (true track) between *ISO-6709:1983-co-ordinate-1* and *ISO-6709:1983-co-ordinate-2*. The calculated course angle value is displayed in degrees and arcminutes by default. The course angle is measured clockwise relative to the geographic, true North (not the magnetic North as shown by a compass), where angle values for the North direction are both denoted as 0 degree and 360 degree. A ‘`*`’ character directly before this mode character causes Gcal to display the course angle value using another style; and that in decimal degrees. If Gcal is unable to compute the approximate course angle between the two geographic point locations by reason of a misspecified *argument*, a `???d??’` text will be created instead of the course angle value.

`2` Like mode `1`, but the course angle (true track) between *ISO-6709:1983-co-ordinate-2* and *ISO-6709:1983-co-ordinate-1* is calculated.

If no *mode* is given, Gcal automatically uses that mode, which is enabled by the mode character ‘`0`’. If a mode character is given that is not according to one of the ‘`0`’...‘`2`’ characters, Gcal also automatically uses that mode, which is enabled by the mode character ‘`0`’.

After the optional style and mode characters, the latitude and longitude of the geographic co-ordinates follows, for which the calculations must be made. They must be conform the ISO-6709:1983 standard representation of latitude and longitude for geographic point locations. The two co-ordinates have to be separated by a ‘`/`’ termination character from each other.

See [Arguments of the Sun oriented special texts], page 171, for the detailed description of the components of the ISO-6709:1983 standard representation of latitude and longitude for geographic point locations.

For example:

[3] The course angle gives the direction, which is the vertex of a great circles arc (*Orthodrome*) that is casted between two points on a surface of a sphere, at a meridian (Gcal uses a geoid (rotation ellipsoid) that is calculated taking pattern from the *World Geodetic System*, short W.G.S, that was designed in 1961). Such an arc is the shortest geodetic connecting line between two points on a sphere. Indeed, the course angles on such an arc are changing continually, because the circular line that is casted intersects each meridian with another angle — except the case, that both co-ordinates are referring to exactly the same *meridian* (longitude), or to exactly the same latitude.

> The text '`Distance Paris-Tokyo: %b+4852+00220/+3542+13946 km`' will be expanded to
> ↦ '`Distance Paris-Tokyo: 9746km`'.
>
> The text '`Distance Paris-Tokyo: %b*0+4852+00220/+3542+13946 ms`' will be expanded to
> ↦ '`Distance Paris-Tokyo: 6056ms`'.
>
> The text '`Course angle Paris-Tokyo: %b1+4852+00220/+3542+13946`' will be expanded to
> ↦ '`Course angle Paris-Tokyo: 033d22''`'.
>
> The text '`Course angle Tokyo-Paris: %b*2+4852+00220/+3542+13946`' will be expanded to
> ↦ '`Course angle Tokyo-Paris: 333.548`'.

While praying, people of Islamic faith always turn their heads into the direction of Makkah, Saudi-Arabia. Now by means of Gcal, these people can easily find out for their respective location, where they have to turn to, and that by:

> `%b1`*`ISO-6709:1983-co-ordinate-1`*`/+212516+0394929`

where *ISO-6709:1983-co-ordinate-1* is simply replaced by the co-ordinate of the respective location.

See [Fixed date option '`--precise`'], page 40, how to obtain a more precise representation of the values that are caused by this special text.

This special text **must** always be trailed by a *whitespace* character which is removed in output!

### G.2.2.5 Sun data %[*format*]?*argument* special texts

%[*format*]o[*][*mode*]*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]] references the approximate time of **sunrise** by default,
%[*format*]s[*][*mode*]*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]] references the approximate time of **sunset** by default,
%[*format*]u[*][*mode*]*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]] references the approximate period of **visibility** of the Sun (solar day length) by default,
%[*format*]z[*][*mode*]*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]] references the approximate period of **non-visibility** of the Sun (solar night length) by default.

All these special texts can be used for at pleasure any geographic point location, i.e. it is possible to determine different astronomical values for any location on the globe, and that for at pleasure any clocktime with a resolution of a single minute within the period of the years AD 1 until AD 9999, that is respected by Gcal.

The selection which value has to be calculated by these special texts is done by specifying the *mode* part of the preceding argument. Actually, exactly 54 different modes can be used that are represented by the '0'...'9', 'a'...'z' and 'A'...'R' characters, and which create different kind of results that are depending on the special text used. First of all, here is a table in which all usable modes are described and explained sufficiently. You can also see from this table, which Sun oriented special text or texts are corresponding to which mode, i.e. cause the determination of an astronomical value as it is described in the table:

| **Mode** | **Special text** | **Description** |
|---|---|---|
| 0 | o, s | Calculates the approximate midnight time of the Sun. The astronomical midnight time of the Sun is at that clocktime, when the Sun holds an *azimuth* (horizontal angular distance between the vertical circle, that passes the Sun, and the North point) of either precisely 0 degrees of precisely 180 degrees (*noon line*), which depends on the season and the geographical location. At that clocktime, the Sun is close its *lowest culmination point*, i.e. close the lowest point below or above the horizontal plane the Sun transits during this day. |

| | | |
|---|---|---|
| 1 | `o, s` | Calculates the approximate noon time of the Sun. The astronomical noon time of the Sun is at that clocktime, when the Sun holds an azimuth of either precisely 180 degrees of precisely 0 degrees (noon line), which depends on the season and the geographical location. At that clocktime, the Sun is close its *highest culmination point*, i.e. close the highest point above or below the horizontal plane the Sun transits during this day. People of Islamic faith normally pray for the second time on the day during the period, which is between the astronomical noon time of the Sun (or some minutes later) and the Islamic prayer time by the name of *Asr*. These people commonly use the term *Zuhr* for this prayer time. The timing of Asr depends on the length of the shadow cast by a vertical pole (*gnomon*). According to the Shafi school of jurisprudence, Asr begins when the length of the shadow of a vertical pole exceeds the length of the pole. According to the Hanafi school of jurisprudence, Asr begins when the length of the shadow exceeds twice the length of the vertical pole. In both cases, the minimum length of the shadow at astronomical noon time of the Sun is subtracted from the length of the shadow before comparing it with the length of the pole. See [Islamic Asr-1 prayer time], page 168, and [Islamic Asr-2 prayer time], page 169, for further details. |
| 2 | `o` | Calculates the approximate time when the center of the Sun passes a reference altitude of 0 degrees on a *mathematical-geocentric* horizon in the morning; thus rising. A mathematical horizon is a purely geometrically-built horizon which disregards the phenomenon of refraction as it arises in reality by the influence of the Earth's atmosphere. A geocentrical horizon is the horizontal plane that passes through the Earth's center, orthogonal to the observer's local vertical. In the further context, the shorter term *mathematical horizon* is used which actually means the mathematical-geocentric horizon. |
| 2 | `s` | Calculates the approximate time when the center of the Sun passes a reference altitude of 0 degrees on a mathematical horizon in the evening; thus setting. |
| 2 | `u` | Calculates the approximate period while the center of the Sun is above a reference altitude of 0 degrees on a mathematical horizon; thus is visible. |
| 2 | `z` | Calculates the approximate period while the center of the Sun is below a reference altitude of 0 degrees on a mathematical horizon; thus is non-visible. |

| | | |
|---|---|---|
| 3 | `o` | Calculates the approximate time when the upper limb of the Sun passes a reference altitude of 0 degrees on a mathematical horizon in the morning; thus rising. The above mentioned reference altitude is computed from the value of the Sun's semidiameter as it appear at that clocktime. If the reference altitude that is referring to the Sun's upper limb is converted to a reference altitude that is referring to the Sun's center, this results in a value of about 16 arcminutes below the geocentric horizon. |
| 3 | `s` | Calculates the approximate time when the upper limb of the Sun passes a reference altitude of 0 degrees on a mathematical horizon in the evening; thus setting. The above mentioned reference altitude is computed from the value of the Sun's semidiameter as it appear at that clocktime. |
| 3 | `u` | Calculates the approximate period while the upper limb of the Sun is above a reference altitude of 0 degrees on a mathematical horizon; thus is visible. |
| 3 | `z` | Calculates the approximate period while the upper limb of the Sun is below a reference altitude of 0 degrees on a mathematical horizon; thus is non-visible. |
| 4 | `o` | Calculates the approximate time when the center of the Sun passes a reference altitude of 34 arcminutes below the geocentric horizon in the morning; thus rising. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the '`--atmosphere`' option. [Fixed dates option '`--atmosphere=`*`air-pressure`*[,*`temperature`*]'], page 37, how to change the base data of the atmosphere, so that the atmospheric conditions as defined by it are used to calculate the amount of refraction. |
| 4 | `s` | Calculates the approximate time when the center of the Sun passes a reference altitude of 34 arcminutes below the geocentric horizon in the evening; thus setting. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the '`--atmosphere`' option. |
| 4 | `u` | Calculates the approximate period while the center of the Sun is above a reference altitude of 34 arcminutes below the geocentric horizon; thus is visible. |
| 4 | `z` | Calculates the approximate period while the center of the Sun is below a reference altitude of 34 arcminutes below the geocentric horizon; thus is non-visible. |

| | | |
|---|---|---|
| `5` | `o` | Calculates the approximate time when the upper limb of the Sun passes a reference altitude of 34 arcminutes below the geocentric horizon in the morning; thus rising. This kind of rise time calculation is done according to the standard calculation method as it is commonly used internationally. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the ‘`--atmosphere`’ option. [Fixed dates option ‘`--atmosphere=`*`air-pressure`*[,*`temperature`*]’], page 37, how to change the base data of the atmosphere, so that the atmospheric conditions as defined by it are used to calculate the amount of refraction. The above mentioned reference altitude is computed from the respective values of the Sun's semidiameter and (standard) refraction as they appear at that clocktime. If the reference altitude that is referring to the Sun's upper limb is converted to a reference altitude that is referring to the Sun's center, this results in a value of about 50 arcminutes below the geocentric horizon. |
| `5` | `s` | Calculates the approximate time when the upper limb of the Sun passes a reference altitude of 34 arcminutes below the geocentric horizon in the evening; thus setting. This kind of set time calculation is done according to the standard calculation method as it is commonly used internationally. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the ‘`--atmosphere`’ option. The above mentioned reference altitude is computed from the respective values of the Sun's semidiameter and (standard) refraction as they appear at that clocktime. People of Islamic faith normally pray for the second-last time on the day at this clocktime, or some minutes later. These people commonly use the term *Maghrib* for this prayer time. |
| `5` | `u` | Calculates the approximate period while the upper limb of the Sun is above a reference altitude of 34 arcminutes below the geocentric horizon; thus is visible. This kind of visibility period calculation is done according to the standard calculation method as it is commonly used internationally. |
| `5` | `z` | Calculates the approximate period while the upper limb of the Sun is above a reference altitude of 34 arcminutes below the geocentric horizon; thus is non-visible. This kind of non-visibility period calculation is done according to the standard calculation method as it is commonly used internationally. |

6 `o` Calculates the approximate time when the center of the Sun passes a reference altitude of 6 degrees below a mathematical horizon in the morning; thus the beginning of *civil twilight*. The scattered light of the Sun that is remaining at the beginning of the civil twilight phase is in general not yet sufficient for reading outside without artificial illumination.

6 `s` Calculates the approximate time when the center of the Sun passes a reference altitude of 6 degrees below a mathematical horizon in the evening; thus the ending of civil twilight.

6 `u` Calculates the approximate period while the center of the Sun is above a reference altitude of 6 degrees below a mathematical horizon; thus the period, while the center of the Sun is always above -6 degrees.

6 `z` Calculates the approximate period while the center of the Sun is below a reference altitude of 6 degrees below a mathematical horizon; thus the period, while the center of the Sun is always below -6 degrees.

7 `o` Calculates the approximate time when the center of the Sun passes a reference altitude of 12 degrees below a mathematical horizon in the morning; thus the beginning of *nautical twilight*. The scattered light of the Sun that is remaining at the beginning of the nautical twilight phase is in general not yet sufficient for navigation using a sea horizon.

7 `s` Calculates the approximate time when the center of the Sun passes a reference altitude of 12 degrees below a mathematical horizon in the evening; thus the ending of nautical twilight.

7 `u` Calculates the approximate period while the center of the Sun is above a reference altitude of 12 degrees below a mathematical horizon; thus the period, while the center of the Sun is always above -12 degrees.

7 `z` Calculates the approximate period while the center of the Sun is below a reference altitude of 12 degrees below a mathematical horizon; thus the period, while the center of the Sun is always below -12 degrees.

8 `o` Calculates the approximate time when the center of the Sun passes a reference altitude of 15 degrees below a mathematical horizon in the morning; thus the beginning of *amateur-astronomical twilight*. The scattered light of the Sun that is remaining at the beginning of the amateur-astronomical twilight phase is in general yet so faint that most astronomical observations can be made.

8 `s` Calculates the approximate time when the center of the Sun passes a reference altitude of 15 degrees below a mathematical horizon in the evening; thus the ending of amateur-astronomical twilight.

8 `u` Calculates the approximate period while the center of the Sun is above a reference altitude of 15 degrees below a mathematical horizon; thus the period, while the center of the Sun is always above -15 degrees.

8 `z` Calculates the approximate period while the center of the Sun is below a reference altitude of 15 degrees below a mathematical horizon; thus the period, while the center of the Sun is always below -15 degrees.

8 `o` Calculates the approximate time when the center of the Sun passes a reference altitude of 18 degrees below a mathematical horizon in the morning; thus the beginning of *astronomical twilight*. No appreciable scattered sunlight is remaining at the beginning of the astronomical twilight phase, the sky is completely dark yet. People of Islamic faith normally pray for the first time on the day during the period, which is between this clocktime and the time of standard sunrise. These people commonly use the term *Fajr* for this prayer time. See [Standard rise time of the Sun], page 158, for further details.

9 `s` Calculates the approximate time when the center of the Sun passes a reference altitude of 18 degrees below a mathematical horizon in the evening; thus the ending of astronomical twilight. People of Islamic faith normally pray for the last time on the day at this clocktime, or some minutes later. These people commonly use the term *Isha* for this prayer time.

9 `u` Calculates the approximate period while the center of the Sun is above a reference altitude of 18 degrees below a mathematical horizon; thus the period, while the center of the Sun is always above -18 degrees.

9 `z` Calculates the approximate period while the center of the Sun is below a reference altitude of 18 degrees below a mathematical horizon; thus the period, while the center of the Sun is always below -18 degrees.

a o, s Calculates the approximate *topocentric*, apparent *elevation* of the Sun, thus the vertical angular distance between the Sun's center and the horizon, in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Sun's center is below the horizon at the moment, and results with a positive sign mean that the momentary center of the Sun is above the horizon. Observations of celestial objects that are done from the surface of the Earth yield in topocentrically based data. The locations of the celestial bodies are often at another place if the data is topocentrically determined instead of determine it *geocentrically*, i.e. at the fictitious center of the Earth. This is mainly caused by the refraction, which raises a celestial body to another location as it is been in reality. Because the terrestrial globe flattens towards the pole caps and therefore cannot be taken as an ideally shaped sphere, the individual Earth radius between the observer's location and the center of the Earth also affects the computation of topocentrically based data.

b o, s Calculates the approximate topocentric, apparent azimuth of the Sun in degrees and arcminutes as it happen at civil midnight time.

c o, s Calculates the approximate topocentric, apparent *declination* of the Sun, thus the vertical angular distance between the Sun's center and the celestial equator, in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Sun's center is below the celestial equator at the moment, and results with a positive sign mean that the momentary center of the Sun is below the celestial equator.

d o, s Calculates the approximate topocentric, apparent *ecliptic longitude* of the Sun, thus the horizontal angular distance between the Sun's center and the vernal equinox point on the *ecliptic* (the zodiacal line or Sun's orbit), in degrees and arcminutes as it happen at civil midnight time.

e o, s Calculates the approximate topocentric, apparent *right ascension* of the Sun, thus the horizontal angular distance between the Sun's center and the hour circle that passes through the vernal equinox point on the ecliptic, as time value in hours and minutes as it happen at civil midnight time.

f o, s Calculates the approximate topocentric, apparent distance of the Sun from the Earth in *astronomical units* as it happen at civil midnight time. An astronomical unit, abbreviated by ae, is equal to the mean distance of the Sun from the Earth, which is about 149,597,870.691 kilometers.

g o, s Calculates the approximate topocentric, apparent *horizontal parallax* of the Sun in degrees and arcminutes as it happen at civil midnight time. The horizontal parallax of the Sun specifies the diameter of the Earth as it is seen from the surface of the Sun.

`h` `o, s` Calculates the approximate topocentric, apparent semidiameter of the Sun in degrees and arcminutes as it happen at civil midnight time.

`i` `o, s` Calculates the approximate refraction of the Earth's atmosphere in degrees and arcminutes as it happen at civil midnight time.

`j` `o, s` Calculates the approximate geocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Sun's center is below the horizon at the moment, and results with a positive sign mean that the momentary center of the Sun is above the horizon.

`k` `o, s` Calculates the approximate geocentric, apparent azimuth of the Sun in degrees and arcminutes as it happen at civil midnight time.

`l` `o, s` Calculates the approximate geocentric, apparent declination of the Sun in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Sun's center is below the celestial equator at the moment, and results with a positive sign mean that the momentary center of the Sun is above the celestial equator.

`m` `o, s` Calculates the approximate geocentric, apparent ecliptic longitude of the Sun in degrees and arcminutes as it happen at civil midnight time.

`n` `o, s` Calculates the approximate geocentric, apparent right ascension of the Sun as time value in hours and minutes as it happen at civil midnight time.

`o` `o, s` Calculates the approximate geocentric, apparent distance of the Sun from the Earth in astronomical units as it happen at civil midnight time.

`p` `o, s` Calculates the approximate geocentric, apparent horizontal parallax of the Sun in degrees and arcminutes as it happen at civil midnight time.

`q` `o, s` Calculates the approximate geocentric, apparent semidiameter of the Sun in degrees and arcminutes as it happen at civil midnight time.

`r` `o, s` Calculates the approximate *delta-t* in seconds as it happen at civil midnight time. Delta-t is the difference between the Terrestrial Dynamical time (abbreviated by TDT), that was formerly known as *Ephemeris time* (abbreviated by ET), and the Universal time (UT). Thus, '`delta-t ≡ TDT - UT`'.

| | | |
|---|---|---|
| s | o, s | Calculates the approximate, apparent location oriented *sidereal time* (local sidereal time (LAST), also known as *local star time*) in hours and minutes as it happen at civil midnight time. A *star day* is the period between two consecutive upper culminations of the vernal equinox point on the ecliptic in the meridian of the observer's location. Therefore, the local *star time* is the momentary period, which is past between the last upper culmination of the vernal equinox point in the meridian of the observer's location (the momentary hour angle of the vernal equinox point), thus the right ascension of the stars in the observer's meridian at the moment. |
| t | o, s | Outputs the base time as time value in hours and minutes, for which the dynamical, i.e. depending on the respective clocktime, astronomical data and times of the Sun are calculated. Without a given ‘--time-offset=*argument*’ option, the astronomical data and times of the Sun are always calculated for 0 o'clock Universal time (UTC/GMT). See [Calendar option ‘--time-offset=*argument*’], page 31, for further details. |
| u | o, s | Calculates the approximate Julian date in days as it happen at civil midnight time. See Section G.2.2.3 [Julian day number], page 152, for further information about the Julian date. |
| v | o, s | Calculates the approximate *Julian Ephemeris date*, thus a Julian date that is corrected by *delta-t*, in days as it happen at civil midnight time. |
| w | o, s | Calculates the approximate difference between *true solar time* and *mean solar time* as time value in hours and minutes as it happen at civil midnight time. This so-called *equation of time* is a correction to be added to the true solar time —as read on a sundial— to obtain the mean solar time. A *true solar day* is the period between two consecutive lower culminations of the Sun. This entity is taken as the base for deriving the true solar time (as it is also shown by a sundial during the day). A star day is also known as a *mean solar day*. Because the Sun apparently shifts with respect to the vernal equinox point on the ecliptic due to the Earth's orbit around the Sun, the star day and the true solar day have a different length. As the true Sun namely moves irregularly through the ecliptic, a fictitious *mean Sun* with a symmetrical motion through the celestial equator is used for deriving the mean solar time. So, this difference in time is a consequence of the ellipticity and tilt of the Earth's orbit, causing the irregular apparent movement of the Sun across the sky. |

x o, s Calculates the difference of the approximate topocentric, apparent elevation of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the momentary center of the Sun is at an elevation that is below the momentary elevation of the Moon's center; thus the Sun is lower than the Moon. Results with a positive sign signify that the momentary center of the Sun is at an elevation that is above the momentary elevation of the Moon's center; thus the Sun is higher than the Moon.

y o, s Calculates the difference of the approximate topocentric, apparent azimuth of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at civil midnight time. The result specifies the horizontal angular distance, by which the momentary center of the Sun is distant from the momentary Moon's center, and that measured at the vertical circles that pass the Sun and the North point and the Moon and the North point. Results with a negative sign signify that the Moon is to the right (clockwise) of the Sun if one looks to the Sun — or alternatively expressed, that the Sun is to the left (anti-clockwise) of the Moon. Results with a positive sign signify that the Moon is to the left (anti-clockwise) of the Sun if one looks to the Sun — or alternatively expressed, that the Sun is to the right (clockwise) of the Moon.

z o, s Calculates the difference of the approximate geocentric, apparent elevation of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at civil midnight time.

A o, s Calculates the difference of the approximate geocentric, apparent azimuth of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at civil midnight time.

B o Calculates the difference of the approximate topocentric, apparent elevation of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Sun. See [Standard rise time of the Sun], page 158, for further details.

B s Calculates the difference of the approximate topocentric, apparent elevation of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Sun. See [Standard set time of the Sun], page 158, for further details.

C o

Calculates the difference of the approximate topocentric, apparent azimuth of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Sun. See [Standard rise time of the Sun], page 158, for further details.

C s

Calculates the difference of the approximate topocentric, apparent azimuth of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Sun. See [Standard set time of the Sun], page 158, for further details.

D o

Calculates the difference of the approximate geocentric, apparent elevation of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Sun. See [Standard rise time of the Sun], page 158, for further details.

D s

Calculates the difference of the approximate geocentric, apparent elevation of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Sun. See [Standard set time of the Sun], page 158, for further details.

E o

Calculates the difference of the approximate geocentric, apparent azimuth of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Sun. See [Standard rise time of the Sun], page 158, for further details.

E s

Calculates the difference of the approximate geocentric, apparent azimuth of Sun and Moon (delta), at which the Sun is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Sun. See [Standard set time of the Sun], page 158, for further details.

F o, s

Calculates the difference of the approximate astronomical midnight times of Sun and Moon (delta), at which the Sun is used as the reference point, as time value in hours and minutes as it happen at astronomical midnight time of the Sun. Results with a negative sign signify that the astronomical midnight time of the Sun is earlier than the astronomical midnight time of the Moon; thus the solar midnight is before the lunar midnight. Results with a positive sign signify that the astronomical midnight time of the Sun is later than the astronomical midnight time of the Moon; thus the solar midnight is after the lunar midnight. See [Astronomical midnight time of the Sun], page 155, and [Astronomical midnight time of the Moon], page 176, for further details.

G o, s

Calculates the difference of the approximate astronomical noon times of Sun and Moon (delta), at which the Sun is used as the reference point, as time value in hours and minutes as it happen at astronomical noon time of the Sun. Results with a negative sign signify that the astronomical noon time of the Sun is earlier than the astronomical noon time of the Moon; thus the solar noon is before the lunar noon. Results with a positive sign signify that the astronomical noon time of the Sun is later than the astronomical noon time of the Moon; thus the solar noon is after the lunar noon. See [Astronomical noon time of the Sun], page 156, and [Astronomical noon time of the Moon], page 176, for further details.

H o

Calculates the difference of the approximate standard rise times of Sun and Moon (delta), at which the Sun is used as the reference point, as time value in hours and minutes as it happen at standard rise time of the Sun. Results with a negative sign signify that the standard rise time of the Sun is earlier than the standard rise time of the Moon; thus the sunrise is before the moonrise. Results with a positive sign signify that the standard rise time of the Sun is later than the standard rise time of the Moon; thus the sunrise is after the moonrise. See [Standard rise time of the Sun], page 158, and [Standard rise time of the Moon], page 179, for further details.

H s

Calculates the difference of the approximate standard set times of Sun and Moon (delta), at which the Sun is used as the reference point, as time value in hours and minutes as it happen at standard set time of the Sun. Results with a negative sign signify that the standard set time of the Sun is earlier than the standard set time of the Moon; thus the sunset is before the moonset. Results with a positive sign signify that the standard set time of the Sun is later than the standard set time of the Moon; thus the sunset is after the moonset. See [Standard set time of the Sun], page 158, and [Standard set time of the Moon], page 179, for further details.

I o, s

Calculates the approximate topocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at astronomical midnight time of the Sun (topocentric *midnight height*). See [Astronomical midnight time of the Sun], page 155, for further details.

J o, s

Calculates the approximate topocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at astronomical midnight time of the Sun (topocentric *midnight height*). See [Astronomical noon time of the Sun], page 156, for further details.

K o

Calculates the approximate topocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at standard rise time of the Sun (topocentric *rise height*). See [Standard rise time of the Sun], page 158, for further details.

K s

Calculates the approximate topocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at standard set time of the Sun (topocentric *set height*). See [Standard set time of the Sun], page 158, for further details.

L o

Calculates the approximate topocentric, apparent azimuth of the Sun in degrees and arcminutes as it happen at standard rise time of the Sun (topocentric *rise azimuth*). The horizontal angular distance between the topocentric rise azimuth and the East direction is also known as the topocentric *morning width* of the Sun. See [Standard rise time of the Sun], page 158, for further details.

L s

Calculates the approximate topocentric, apparent azimuth of the Sun in degrees and arcminutes as it happen at standard set time of the Sun (topocentric *set azimuth*). The horizontal angular distance between the topocentric set azimuth and the West direction is also known as the topocentric *evening width* of the Sun. See [Standard set time of the Sun], page 158, for further details.

M o, s

Calculates the approximate geocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at astronomical midnight time of the Sun (geocentric *midnight height*). See [Astronomical midnight time of the Sun], page 155, for further details.

N o, s

Calculates the approximate geocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at astronomical midnight time of the Sun (geocentric *midnight height*). See [Astronomical noon time of the Sun], page 156, for further details.

O o

Calculates the approximate geocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at standard rise time of the Sun (geocentric *rise height*). See [Standard rise time of the Sun], page 158, for further details.

O s

Calculates the approximate geocentric, apparent elevation of the Sun in degrees and arcminutes as it happen at standard set time of the Sun (geocentric *set height*). See [Standard set time of the Sun], page 158, for further details.

P o

Calculates the approximate geocentric, apparent azimuth of the Sun in degrees and arcminutes as it happen at standard rise time of the Sun (geocentric *rise azimuth*). The horizontal angular distance between the geocentric rise azimuth and the East direction is also known as the geocentric *morning width* of the Sun. See [Standard rise time of the Sun], page 158, for further details.

`P` `s` Calculates the approximate geocentric, apparent azimuth of the Sun in degrees and arcminutes as it happen at standard set time of the Sun (geocentric *set azimuth*). The horizontal angular distance between the geocentric set azimuth and the West direction is also known as the geocentric *evening width* of the Sun. See [Standard set time of the Sun], page 158, for further details.

`Q` `o` Calculates the approximate time when the length of the shadow cast by a vertical pole in the forenoon is **equal** the length of the pole. Nevertheless, the minimum length of the shadow is subtracted from the length of the shadow before comparing it with the length of the pole. See [Fixed dates option ‘`--adjust-value=`*`argument`*’], page 36, how to change the shadow length factor.

`Q` `s` Calculates the approximate time when the length of the shadow cast by a vertical pole in the afternoon is **equal** the length of the pole. Nevertheless, the minimum length of the shadow is subtracted from the length of the shadow before comparing it with the length of the pole. People of Islamic faith, and that the people holding the *Shafi* school of jurisprudence, normally pray for the third time on the day at this clocktime, or some minutes later. These people commonly use the term *Asr* for this prayer time. See [Astronomical noon time of the Sun], page 156, for more information. And note [Fixed dates option ‘`--adjust-value=`*`argument`*’], page 36, how to change the shadow length factor.

`Q` `u` Calculates the approximate period while the center of the Sun is above a reference altitude, at which the length of the shadow cast by a vertical pole is of single or shorter length than the pole itself.

`Q` `z` Calculates the approximate period while the center of the Sun is below a reference altitude, at which a vertical pole either casts no shadow anymore, or casts a shadow that is longer than the single length of the pole itself.

`R` `o` Calculates the approximate time when the length of the shadow cast by a vertical pole at forenoon is **twice** the length of the pole. Nevertheless, the minimum length of the shadow is subtracted from the length of the shadow before comparing it with the length of the pole. See [Fixed dates option ‘`--adjust-value=`*`argument`*’], page 36, how to change the shadow length factor.

`R` `s` Calculates the approximate time when the length of the shadow cast by a vertical pole in the afternoon is **twice** the length of the pole. Nevertheless, the minimum length of the shadow is subtracted from the length of the shadow before comparing it with the length of the pole. People of Islamic faith, and that the people holding the *Hanafi* school of jurisprudence, normally pray for the third time on the day at this clocktime, or some minutes later. These people commonly use the term *Asr* for this prayer time. See [Astronomical noon time of the Sun], page 156, for more information. And note [Fixed dates option '`--adjust-value=`*`argument`*'], page 36, how to change the shadow length factor.

`R` `u` Calculates the approximate period while the center of the Sun is above a reference altitude, at which the length of the shadow cast by a vertical pole is of double or shorter length than the pole itself.

`R` `z` Calculates the approximate period while the center of the Sun is below a reference altitude, at which a vertical pole either casts no shadow anymore, or casts a shadow that is longer than twice the length of the pole itself.

If no *mode* is given, Gcal automatically uses that mode, which is enabled by the mode character '`5`'. If a mode character is given that is not according to one of the '`0`'...'`9`', '`a`'...'`z`' and '`A`'...'`R`' characters, Gcal also automatically uses that mode, which is enabled by the mode character '`5`'.

Gcal represents the Sun oriented special texts depending on the selected mode using the following types and styles:

1. Unsigned decimal based rational number value
   Unsigned decimal based rational number values are represented using the *`n`*`.`*`n`*`...` format by default. A '`*`' character that is directly given before some mode characters causes Gcal to represent the value for another quantity. For the mode characters, which
   a. cause the calculation of Earth/Sun or Earth/Moon distances, the calculated distance is represented in kilometers.
   b. cause the calculation of phase angles of the Moon, the calculated phase angle is represented as a phase value in percents.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations. See [Event texts of the Sun oriented special texts], page 170, where you can find the event oriented texts that are created for clocktime values, which are schematically and analogously used for the type of representation as it is described here.
2. Signed decimal based rational number value
   Signed decimal based rational number values are represented using the `+|-`*`n`*`.`*`n`*`...` format by default. A '`*`' character that is directly given before a mode character causes Gcal not to represent such values using another style.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations. See [Event texts of the Sun oriented special texts], page 170, where you can find the event oriented texts that are created for clocktime values, which are schematically and analogously used for the type of representation as it is described here.

3. Clocktime value
   Clocktime values are represented in hours and minutes, and that in the *hh*:*mm* *24-hour* format by default. A '*' character that is directly given before a mode character causes Gcal to represent the clocktime value using the *12-hour* format, thus to provide it with a time suffix. See [Actual localtime in *hh*:*mm* format %t[*argument*] special text], page 149, for more details about the above mentioned time value template.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations:

   - In case Gcal is unable to compute the approximate clocktime by reason of a misspecified *argument*, a ??:?? text will be created instead of the clocktime text.
   - In case Gcal calculates an approximate clocktime that refers to a previous or next day, or Gcal is unable to calculate a clocktime because the event calculated does not occur during this day, a **:** text will be created instead of the clocktime text.
   - In case Gcal is unable to calculate an approximate clocktime because the reference point of the astronomical object, thus either the center or the upper limb of the Sun's or the Moon's disk, is always above the respective reference altitude, a ++:++ text will be created instead of the clocktime text.
   - In case Gcal is unable to calculate an approximate clocktime because the reference point of the astronomical object, thus either the center or the upper limb of the Sun's or the Moon's disk, is always below the respective reference altitude, a --:-- text will be created instead of the clocktime text.

4. Unsigned time value
   Unsigned time values, which mostly denote a period or interval of time, are represented in hours and minutes using the *hh*h*mm*' format by default. A '*' character that is directly given before a mode character causes Gcal to represent the time value using another style, and that in decimal hours, i.e. in the *hh*.*h*... format.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations. See [Event texts of the Sun oriented special texts], page 170, where you can find the event oriented texts that are created for clocktime values, which are schematically and analogously used for the type of representation as it is described here.

5. Signed time value
   Signed time values, which mostly denote a period or interval of time, are represented in hours and minutes using the +|-*hh*h*mm*' format by default. A '*' character that is directly given before a mode character causes Gcal to represent the time value using another style, and that in decimal hours, i.e. in the +|-*hh*.*h*... format.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations. See [Event texts of the Sun oriented special texts], page 170, where you can find the event oriented texts that are created for clocktime values, which are schematically and analogously used for the type of representation as it is described here.

6. Unsigned angular value
   Unsigned angular values are represented in degrees and arcminutes using the *ddd*d*mm*' format by default. A '*' character that is directly given before a mode character causes Gcal to represent the angular value using another style, and that in decimal degrees, i.e. in the *ddd*.*d*... format.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations. See [Event texts of the Sun oriented special texts], page 170, where you can find the event oriented texts that are created for clocktime values, which are schematically and analogously used for the type of representation as it is described here.

7. Signed angular value
   Signed angular values are represented in degrees and arcminutes using the `+|-dddmm'` format by default. A '`*`' character that is directly given before a mode character causes Gcal to represent the angular value using another style, and that in decimal degrees, i.e. in the `+|-ddd.d...` format.

   If definite events happen, Gcal displays special event oriented texts instead of using the previously described representations. See [Event texts of the Sun oriented special texts], page 170, where you can find the event oriented texts that are created for clocktime values, which are schematically and analogously used for the type of representation as it is described here.

After the optional style and mode characters, the latitude and longitude of the geographic co-ordinates follows, for which the calculations must be made. They must be conform the ISO-6709:1983 standard representation of latitude and longitude for geographic point locations, so that the co-ordinate has to be declared like this:

- Latitude (`+` for North and on the equator, `-` for South of the equator.)

  `+|-dd[.dd]`
  Degrees (2 digits), and optionally decimal degrees

  `+|-ddmm[.mm]`
  Degrees (2 digits), arcminutes (2 digits), and optionally decimal minutes

  `+|-ddmmss[.ss]`
  Degrees (2 digits), arcminutes (2 digits), arcseconds (2 digits), and optionally decimal seconds

- Longitude (`+` for East and on the prime meridian (Greenwich), `-` for West of Greenwich and up to the 180th meridian.)

  `+|-ddd[.dd]`
  Degrees (3 digits), and optionally decimal degrees

  `+|-dddmm[.mm]`
  Degrees (3 digits), arcminutes (2 digits), and optionally decimal minutes

  `+|-dddmmss[.ss]`
  Degrees (3 digits), arcminutes (2 digits), arcseconds (2 digits), and optionally decimal seconds

- Optionally, the relative height in meters above the sea level (`+` for heights above and on the sea level, `-` for heights below the sea level.)

  `[+|-n[n[n[n]]]]`
  Height in meters (integer number) in range `-9999...+9999` (1...4 digits)

All components of the co-ordinates must have leadings zeroes in case they have less digits than the templates shown above. Declared decimal seconds are not respected by Gcal. Heights which have a negative sign remain unrespected if Gcal determinates Sun and Moon data and times, respectively. In such a case, Gcal always uses the height `+0`. Latitude and longitude co-ordinates, and the height of the observer's location are connected without any separating characters, like '`+40-075+61`', '`+401213.1-0750015.1`' or '`+40.20361-075.00417+0061`'. See the pertinent literature for more details.

A time value `[+|-]mmmm|hh:[mm]`, which is separated by a '`,`' character, may trail the co-ordinate. Such a time value informs Gcal, about how many minutes *mmmm* respectively hours *hh* and minutes *mm* the geographic location is displaced from Universal time (UTC/GMT).

This time displacement value defines the timezone, which is actually valid for this location. If summer- and wintertimes are respected for the location, you should include that change in time into the timezone value for the period in which the summertime is valid, by which the clock is put on during the summertime period — such a change is either subtracted from the timezone value for locations West of the prime meridian (Greenwich), or it is added for locations East of the prime meridian, because Gcal is actually unable to perform such operations automatically! See [Actual local time in *hh*:*mm* format %t[*argument*] special text], page 149, for more details about the above mentioned time value template. If no time displacement value is specified for a given co-ordinate, Gcal assumes a time displacement value of `0`, which is equal to the actual Universal time (UTC/GMT).

The following table informs you about which type of representation is caused by a mode. The previously defined numbering scheme, as it has been used for the introduction of the types of representations, is used as key value in the column that holds the type of representation. The table also contains a column that shows whether a mode enables dynamical values, i.e. values that are depending on the respective clocktime (if you use the '`--time-offset=`*argument*' option, you can change the respective clocktime that is used for calculating such values). In a next table column, it is listed whether the given co-ordinate of the location influences the determination of a value, and the last column of the table gives you the information whether a given timezone value affects the values determination:

| **Mode** | **Representation Type** | **Dynamical** | **Co-ordinate** | **Timezone** |
|---|---|---|---|---|
| `0` | 3 | No | Yes | Yes |
| `1` | 3 | No | Yes | Yes |
| `2` | 3 or 4 | No | Yes | Yes |
| `3` | 3 or 4 | No | Yes | Yes |
| `4` | 3 or 4 | No | Yes | Yes |
| `5` | 3 or 4 | No | Yes | Yes |
| `6` | 3 or 4 | No | Yes | Yes |
| `7` | 3 or 4 | No | Yes | Yes |
| `8` | 3 or 4 | No | Yes | Yes |
| `9` | 3 or 4 | No | Yes | Yes |
| `a` | 7 | Yes | Yes | Yes |
| `b` | 6 | Yes | Yes | Yes |
| `c` | 7 | Yes | Yes | Yes |
| `d` | 6 | Yes | Yes | Yes |
| `e` | 4 | Yes | Yes | Yes |
| `f` | 1 or 1a | Yes | Yes | Yes |
| `g` | 6 | Yes | Yes | Yes |
| `h` | 6 | Yes | Yes | Yes |
| `i` | 6 | Yes | No | No |
| `j` | 7 | Yes | Yes | Yes |
| `k` | 6 | Yes | Yes | Yes |
| `l` | 7 | Yes | Yes | Yes |
| `m` | 6 | Yes | Yes | Yes |
| `n` | 4 | Yes | Yes | Yes |
| `o` | 1 or 1a | Yes | Yes | Yes |
| `p` | 6 | Yes | Yes | Yes |
| `q` | 6 | Yes | Yes | Yes |
| `r` | 2 | Yes | No | No |

| | | | | |
|---|---|---|---|---|
| s | 3 | Yes | Yes | Yes |
| t | 3 | Yes | No | No |
| u | 1 | Yes | No | No |
| v | 1 | Yes | No | No |
| w | 5 | Yes | No | Yes |
| x | 7 | Yes | Yes | Yes |
| y | 7 | Yes | Yes | Yes |
| z | 7 | Yes | Yes | Yes |
| A | 7 | Yes | Yes | Yes |
| B | 7 | No | Yes | Yes |
| C | 7 | No | Yes | Yes |
| D | 7 | No | Yes | Yes |
| E | 7 | No | Yes | Yes |
| F | 5 | No | Yes | Yes |
| G | 5 | No | Yes | Yes |
| H | 5 | No | Yes | Yes |
| I | 7 | No | Yes | Yes |
| J | 7 | No | Yes | Yes |
| K | 7 | No | Yes | Yes |
| L | 6 | No | Yes | Yes |
| M | 7 | No | Yes | Yes |
| N | 7 | No | Yes | Yes |
| O | 7 | No | Yes | Yes |
| P | 6 | No | Yes | Yes |
| Q | 3 or 4 | No | Yes | Yes |
| R | 3 or 4 | No | Yes | Yes |

And now some examples to these special texts:

> The text ‘Sunrise at %o+5158+00738,120  in MS, BRD’ will be expanded to
> ↦ ‘Sunrise at 05:16 in MS, BRD’, in case the actual system date is the 1st June 1998.
>
> The text ‘Sunset at %s*5+5158+00738,120  in MS, BRD’ will be expanded to
> ↦ ‘Sunset at 09:39pm in MS, BRD’, in case the actual system date is the 1st June 1998.
>
> The text ‘Sun visible %u5+5158+00738,120  in MS, BRD’ will be expanded to
> ↦ ‘Sun visible 16h24’ in MS, BRD’, in case the actual system date is the 1st June 1998.
>
> The text ‘Sun non-visible %z*+5158+00738,120  in MS, BRD’ will be expanded to
> ↦ ‘Sun non-visible 7.607 in MS, BRD’, in case the actual system date is the 1st June 1998.
>
> The text ‘Sun azimuth 0 o’clock=%s*a+5158+00738,120  in MS, BRD’ will be expanded to
> ↦ ‘Sun azimuth 0 o’clock=339d16’ in MS, BRD’, in case the actual system date is the 1st June 1998.
>
> The text ‘Equation of time %ot+00+000=%o*w+00+000,120  BRD’ will be expanded to
> ↦ ‘Equation of time +16h00’=+00h02’13.201" BRD’, in case you call Gcal with the ‘--time-offset=16:’ and ‘--precise’ options and the actual system date is the 1st June 1998.

> The text '`Julian date at %ot+00+000 =%ou+00+000`' will be expanded to
> ↦ '`Julian date at +10h15'=2450965.927`', in case you call Gcal with the '`--time-offset=10:15`' option and the actual system date is the 1st June 1998.

Here is a list that reports about the used reference systems in a short manner, describes other aspects that are unmentioned now, and informs about the lacks and limitations that are existing for the Sun oriented special texts:

- All astronomical calculations are based on methods that use —astronomically seen— low-precision elements, which of course is shown in the precision of the results that are caused by them. Nevertheless in most cases, these results can be treated as more as sufficiently precise if one measures their practical use.
- All azimuth and elevation values are always referring to the center of the respective astronomical object, except the case, it is stated otherwise.
- The base point of azimuth values, right ascension values and other ecliptic co-ordinate values is always measured clockwise relative to the geographical, true North (not the magnetic North as shown by a compass), where angular values for the North direction are both denoted as 0 degree and 360 degree.
- All topocentrically based azimuth and elevation values are always corrected by the amount of atmospheric refraction. Such a kind of correction is always done if the center of the respective astronomical object is above 2 degrees below the horizon.
- Generally, the rise and set times of the Sun cannot be precisely predicted, because depending on unpredictable atmospheric conditions, i.e. the local weather conditions as they appear during these times change the amount of atmospheric refraction at the horizon, so that the precalculated rise and set times may throughout be in error by a minute or more.
- The accuracy of rise and set computations decreases again at high latitudes, i.e. within in the solar Arctic Circle. There, small variations in atmospheric refraction can change the time of sunrise or sunset by many minutes, since the Sun intersects the horizon at a very shallow angle. For the same reason, at high latitudes, the effects of observer height and local topography are magnified and can substantially change the times of the phenomena actually observed, or even whether the phenomena are observed to occur at all.
- The short explanations that are given to the different twilight phases all assume the model of ideal atmospheric conditions by default. It is obvious that these explanations all become meaningless if other conditions occur in reality, like a cloudy sky or a badly polluted atmosphere.

**Please also note the following references:**

- See [Fixed dates option '`--precise`'], page 40, how to obtain a more precise representation of the values which are cause by these special texts.
- And note [Calendar option '`--time-offset=`*`argument`*'], page 31, how to change the clocktime, for which the dynamical, i.e. depending on the respective clocktime, Sun oriented special texts are calculated.
- Likewise, see [Fixed dates option '`--cycle-end=`*`argument`*'], page 38, how the dynamical, i.e. depending on the respective clocktime, Sun oriented special texts are calculated for a series of clocktimes for the current day.
- And also note [Fixed dates option '`--adjust-value=`*`argument`*'], page 36, how to change the reference altitude that is used to calculate the rise and set times, as well as the shadow length factor.
- Furthermore, see [Fixed dates option '`--atmosphere=`*`air-pressure`*[,*`temperature`*]'], page 37, how to change the base data of the atmosphere that influences the calculation of the Sun oriented special texts.

- And last, note [Fixed dates option ‘`--limit`’], page 37, how to limit the rise and set times of the Sun to the actual day.

All Sun oriented special texts **must** always be trailed by a *whitespace* character which is removed in output!

### G.2.2.6 Moon data %[format]?argument special texts

%[format]([*][mode]ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]] references the approximate time of **moonrise** by default,
%[format])[*][mode]ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]] references the approximate time of **moonset** by default,
%[format][[*][mode]ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]] references the approximate period of **visibility** of the Moon (lunar day length) by default,
%[format]][*][mode]ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]] references the approximate period of **non-visibility** of the Moon (lunar night length) by default.

All these special texts can be used for at pleasure any geographic point location, i.e. it is possible to determine different astronomical values for any location on the globe, and that for at pleasure any clocktime with a resolution of a single minute within the period of the years AD 1 until AD 9999, that is respected by Gcal.

The selection which value has to be calculated by these special texts is done by specifying the *mode* part of the preceding argument. Actually, exactly 61 different modes can be used that are represented by the '0'...'9', 'a'...'z' and 'A'...'Y' characters, and which create different kind of results that are depending on the special text used. First of all, here is a table in which all usable modes are described and explained sufficiently. You can also see from this table, which Moon oriented special text or texts are corresponding to which mode, i.e. cause the determination of an astronomical value as it is described in the table:

| **Mode** | **Special text** | **Description** |
|---|---|---|
| 0 | (, ) | Calculates the approximate midnight time of the Moon. The astronomical midnight time of the Moon is at that clocktime, when the Moon holds an *azimuth* (horizontal angular distance between the vertical circle, that passes the Moon, and the North point) of either precisely 0 degrees of precisely 180 degrees, which depends on the season and the geographical location. At that clocktime, the Moon is close its *lowest culmination point*, i.e. close the lowest point below or above the horizontal plane the Moon transits during this day. Nevertheless, there is exactly one day during a *synodic month* (or lunation) —i.e. the mean time between two consecutive *conjunctions* (or New Moon phases)— at which no lunar midnight happens, because the Moon revolves the Earth within 24 hours and 50 minutes on the average — which also means, that the Moon rises on the average 50 minutes later each day. |
| 1 | (, ) | Calculates the approximate noon time of the Moon. The astronomical noon time of the Moon is at that clocktime, when the Moon holds an azimuth of either precisely 180 degrees of precisely 0 degrees, which depends on the season and the geographical location. At that clocktime, the Moon is close its *highest culmination point*, i.e. close the highest point above or below the horizontal plane the Moon transits during this day. Nevertheless, there is exactly one day during a synodic month at which no lunar noon happens. |

2 ( Calculates the approximate time when the center of the Moon passes a reference altitude which is between about 54 and 61 arcminutes above a *mathematical-geocentric* horizon before lunar noon time; thus rising. A mathematical horizon is a purely geometrically-built horizon which disregards the phenomenon of refraction as it arises in reality by the influence of the Earth's atmosphere. A geocentrical horizon is the horizontal plane that passes through the Earth's center, orthogonal to the observer's local vertical. In the further context, the shorter term *mathematical horizon* is used which actually means the mathematical-geocentric horizon. The above mentioned reference altitude is computed from the value of the Moon's parallax as it appear at that clocktime. Nevertheless, there is exactly one day during a synodic month at which no such moonrise happens.

2 ) Calculates the approximate time when the center of the Moon passes a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon after lunar noon time; thus setting. The above mentioned reference altitude is computed from the value of the Moon's parallax as it appear at that clocktime. Nevertheless, there is exactly one day during a synodic month at which no such moonset happens.

2 [ Calculates the approximate period while the center of the Moon is above a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon; thus is visible.

2 ] Calculates the approximate period while the center of the Moon is below a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon; thus is non-visible.

3 ( Calculates the approximate time when the upper limb of the Moon passes a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon before lunar noon time; thus rising. The above mentioned reference altitude is computed from the respective values of the Moon's semidiameter and Moon's parallax as they appear at that clocktime. If the reference altitude that is referring to the Moon's upper limb is converted to a reference altitude that is referring to the Moon's center, this results in a value which is between about 39 and 44 arcminutes above the geocentric horizon. Nevertheless, there is exactly one day during a synodic month at which no such moonrise happens.

3 ) Calculates the approximate time when the upper limb of the Moon passes a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon after lunar noon time; thus setting. The above mentioned reference altitude is computed from the respective values of the Moon's semidiameter and Moon's parallax as they appear at that clocktime. Nevertheless, there is exactly one day during a synodic month at which no such moonset happens.

`3` `[` Calculates the approximate period while the upper limb of the Moon is above a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon; thus is visible.

`3` `]` Calculates the approximate period while the upper limb of the Moon is below a reference altitude which is between about 54 and 61 arcminutes above a mathematical horizon; thus is non-visible.

`4` `(` Calculates the approximate time when the center of the Moon passes a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon before lunar noon time; thus rising. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the '`--atmosphere`' option. [Fixed dates option '`--atmosphere=`*`air-pressure`*[,*`temperature`*]'], page 37, how to change the base data of the atmosphere, so that the atmospheric conditions as defined by it are used to calculate the amount of refraction. The above mentioned reference altitude is computed from the respective values of the Moon's parallax and (standard) refraction as they appear at that clocktime. Nevertheless, there is exactly one day during a synodic month at which no such moonrise happens.

`4` `)` Calculates the approximate time when the center of the Moon passes a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon after lunar noon time; thus setting. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the '`--atmosphere`' option. The above mentioned reference altitude is computed from the respective values of the Moon's parallax and (standard) refraction as they appear at that clocktime. Nevertheless, there is exactly one day during a synodic month at which no such moonset happens.

`4` `[` Calculates the approximate period while the center of the Moon is above a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon; thus is visible.

`4` `]` Calculates the approximate period while the center of the Moon is below a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon; thus is non-visible.

5 ( Calculates the approximate time when the upper limb of the Moon passes a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon before lunar noon time; thus rising. This kind of rise time calculation is done according to the standard calculation method as it is commonly used internationally. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the '--atmosphere' option. [Fixed dates option '--atmosphere=*air-pressure*[,*temperature*]'], page 37, how to change the base data of the atmosphere, so that the atmospheric conditions as defined by it are used to calculate the amount of refraction. The above mentioned reference altitude is computed from the respective values of the Moon's semidiameter, Moon's parallax and (standard) refraction as they appear at that clocktime. If the reference altitude that is referring to the Moon's upper limb is converted to a reference altitude that is referring to the Moon's center, this results in a value which is between about 5 and 10 arcminutes above the geocentric horizon. Nevertheless, there is exactly one day during a synodic month at which no such moonrise happens.

5 ) Calculates the approximate time at which the upper limb of the Moon passes a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon after lunar noon time; thus setting. This kind of set time calculation is done according to the standard calculation method as it is commonly used internationally. The phenomenon of refraction is already respected in this as it arises in reality by the influence of the Earth's atmosphere, and that with the standard value of 34 arcminutes, which can indirectly be changed by using the '--atmosphere' option. The above mentioned reference altitude is computed from the respective values of the Moon's semidiameter, Moon's parallax and (standard) refraction as they appear at that clocktime. Nevertheless, there is exactly one day during a synodic month at which no such moonset happens.

5 [ Calculates the approximate period while the upper limb of the Moon is above a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon; thus is visible. This kind of visibility period calculation is done according to the standard calculation method as it is commonly used internationally.

5 ] Calculates the approximate period while the upper limb of the Moon is below a reference altitude which is between about 20 and 27 arcminutes above the geocentric horizon; thus is nonvisible. This kind of non-visibility period calculation is done according to the standard calculation method as it is commonly used internationally.

6 (, ) Calculates the approximate *topocentric*, apparent *horizontal parallax* of the Moon in degrees and arcminutes as it happen at civil midnight time. The Moon's parallax states the diameter of the Earth as it is seen from the surface of the Moon. Observations of celestial objects that are done from the surface of the Earth yield in topocentrically based data. The locations of the celestial bodies are often at another place if the data is topocentrically determined instead of determine it *geocentrically*, i.e. at the fictitious center of the Earth. This is mainly caused by the refraction, which raises a celestial body to another location as it is been in reality. Because the terrestrial globe flattens towards the pole caps and therefore cannot be taken as an ideally shaped sphere, the individual Earth radius between the observer's location and the center of the Earth also affects the computation of topocentrically based data.

7 (, ) Calculates the approximate topocentric, apparent semidiameter of the Moon in degrees and arcminutes as it happen at civil midnight time.

8 (, ) Calculates the approximate topocentric, apparent brightness of the Moon in *magnitude units* as it happen at civil midnight time. The magnitude (Latin term *magnitudo*, abbreviated m) is used to define the brightness of a star, and is a non-metrical value. The difference between two consecutive magnitudes is 1 to 2.512. Therefore, a star with the brightness of 1m is 2.512 times brighter than a star of 2m. A negative magnitude denotes a very bright star, for example almost -27m for the Sun, whereas the hardly visible planet Pluto has a magnitude of a bit more than +14m. The Full Moon has a visual brightness of about -12m.55.

9 (, ) Calculates the approximate topocentric, apparent phase angle of the Moon in range 0.0. . .1.0 as it happen at civil midnight time.

a (, ) Calculates the approximate topocentric, apparent *elevation* of the Moon, thus the vertical angular distance between the Moon's center and the horizon, in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Moon's center is below the horizon at the moment, and results with a positive sign mean that the momentary center of the Moon is above the horizon.

b (, ) Calculates the approximate topocentric, apparent azimuth of the Moon in degrees and arcminutes as it happen at civil midnight time.

c (, ) Calculates the approximate topocentric, apparent *declination* of the Moon, thus the vertical angular distance between the Moon's center and the celestial equator, in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Moon's center is below the celestial equator at the moment, and results with a positive sign mean that the momentary center of the Moon is above the celestial equator.

d (, ) Calculates the approximate topocentric, apparent *ecliptic longitude* of the Moon, thus the horizontal angular distance between the Moon's center and the vernal equinox point on the *ecliptic* (the zodiacal line or Sun's orbit), in degrees and arcminutes as it happen at civil midnight time.

e (, ) Calculates the approximate topocentric, apparent *ecliptic latitude* of the Moon, thus the vertical angular distance between the Moon's center and the ecliptic (zodiacal line/Sun's orbit), in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Moon's center is North of the ecliptic at the moment, and results with a positive sign mean that the momentary center of the Moon is South of the ecliptic.

f (, ) Calculates the approximate topocentric, apparent *right ascension* of the Moon, thus the horizontal angular distance between the Moon's center and the hour circle that passes through the vernal equinox point on the ecliptic, as time value in hours and minutes as it happen at civil midnight time.

g (, ) Calculates the approximate topocentric, apparent distance of the Moon from the Earth in mean *Earth equator radii* as it happen at civil midnight time. The mean radius of Earth at the equator is about 6,378.137 kilometer.

h (, ) Calculates the approximate topocentric, apparent *elongation* of the Moon, thus the horizontal angular distance between the Moon's center and the Sun's center, in degrees and arcminutes as it happen at civil midnight time.

i (, ) Calculates the approximate refraction of the Earth's atmosphere in degrees and arcminutes as it happen at civil midnight time.

j (, ) Calculates the approximate geocentric, apparent horizontal parallax of the Moon in degrees and arcminutes as it happen at civil midnight time.

k (, ) Calculates the approximate geocentric, apparent semidiameter of the Moon in degrees and arcminutes as it happen at civil midnight time.

l (, ) Calculates the approximate geocentric, apparent brightness of the Moon in magnitude units as it happen at civil midnight time.

`m` `(, )` Calculates the approximate geocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at civil midnight time.

`n` `(, )` Calculates the approximate geocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Moon's center is below the horizon at the moment, and results with a positive sign mean that the momentary center of the Moon is above the horizon.

`o` `(, )` Calculates the approximate geocentric, apparent azimuth of the Moon in degrees and arcminutes as it happen at civil midnight time.

`p` `(, )` Calculates the approximate geocentric, apparent declination of the Moon in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Moon's center is below the celestial equator at the moment, and results with a positive sign mean that the momentary center of the Moon is above the celestial equator.

`q` `(, )` Calculates the approximate geocentric, apparent ecliptic longitude of the Moon in degrees and arcminutes as it happen at civil midnight time.

`r` `(, )` Calculates the approximate geocentric, apparent ecliptic latitude of the Moon in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the Moon's center is North of the ecliptic at the moment, and results with a positive sign mean that the momentary center of the Moon is South of the ecliptic.

`s` `(, )` Calculates the approximate geocentric, apparent right ascension of the Moon as time value in hours and minutes as it happen at civil midnight time.

`t` `(, )` Calculates the approximate geocentric, apparent distance of the Moon from the Earth in mean Earth equator radii as it happen at civil midnight time.

`u` `(, )` Calculates the approximate geocentric, apparent elongation of the Moon in degrees and arcminutes as it happen at civil midnight time.

`v` `(, )` Calculates the approximate *delta-t* in seconds as it happen at civil midnight time. Delta-t is the difference between the Terrestrial Dynamical time (abbreviated by TDT), that was formerly known as *Ephemeris time* (abbreviated by ET), and the Universal time (UT). Thus, '`delta-t ≡ TDT - UT`'.

w (, ) Calculates the approximate, apparent location oriented *sidereal time* (local sidereal time (LAST), also known as *local star time*) in hours and minutes as it happen at civil midnight time. A *star day* is the period between two consecutive upper culminations of the vernal equinox point on the ecliptic in the meridian of the observer's location. Therefore, the local *star time* is the momentary period, which is past between the last upper culmination of the vernal equinox point in the meridian of the observer's location (the momentary hour angle of the vernal equinox point), thus the right ascension of the stars in the observer's meridian at the moment.

x (, ) Outputs the base time as time value in hours and minutes, for which the dynamical, i.e. depending on the respective clock-time, astronomical data and times of the Moon are calculated. Without a given '`--time-offset=`*`argument`*' option, the astronomical data and times of the Moon are always calculated for 0 o'clock Universal time (UTC/GMT). See [Calendar option '`--time-offset=`*`argument`*'], page 31, for further details.

y (, ) Calculates the approximate Julian date in days as it happen at civil midnight time. See Section G.2.2.3 [Julian day number], page 152, for further information about the Julian date.

z (, ) Calculates the approximate *Julian Ephemeris date*, thus a Julian date that is corrected by *delta-t*, in days as it happen at civil midnight time.

A (, ) Calculates the difference of the approximate topocentric, apparent elevation of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at civil midnight time. Results with a negative sign signify that the momentary center of the Moon is at an elevation that is below the momentary elevation of the Sun's center; thus the Moon is lower than the Sun. Results with a positive sign signify that the momentary center of the Moon is at an elevation that is above the momentary elevation of the Sun's center; thus the Moon is higher than the Sun.

`B` `(, )` Calculates the difference of the approximate topocentric, apparent azimuth of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at civil midnight time. The result specifies the horizontal angular distance, by which the momentary center of the Moon is distant from the momentary Sun's center, and that measured at the vertical circles that pass the Moon and the North point and the Sun and the North point. Results with a negative sign signify that the Sun is to the right (clockwise) of the Moon if one looks to the Moon — or alternatively expressed, that the Moon is to the left (anti-clockwise) of the Sun. Results with a positive sign signify that the Sun is to the left (anti-clockwise) of the Moon if one looks to the Moon — or alternatively expressed, that the Moon is to the right (clockwise) of the Sun.

`C` `(, )` Calculates the difference of the approximate geocentric, apparent elevation of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at civil midnight time.

`D` `(, )` Calculates the difference of the approximate geocentric, apparent azimuth of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at civil midnight time.

`E` `(` Calculates the difference of the approximate topocentric, apparent elevation of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Moon. See [Standard rise time of the Moon], page 179, for further details.

`E` `)` Calculates the difference of the approximate topocentric, apparent elevation of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Moon. See [Standard set time of the Moon], page 179, for further details.

`F` `(` Calculates the difference of the approximate topocentric, apparent azimuth of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Moon. See [Standard rise time of the Moon], page 179, for further details.

`F` `)` Calculates the difference of the approximate topocentric, apparent azimuth of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Moon. See [Standard set time of the Moon], page 179, for further details.

| | | |
|---|---|---|
| `G` | `(` | Calculates the difference of the approximate geocentric, apparent elevation of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Moon. See [Standard rise time of the Moon], page 179, for further details. |
| `G` | `)` | Calculates the difference of the approximate geocentric, apparent elevation of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Moon. See [Standard set time of the Moon], page 179, for further details. |
| `H` | `(` | Calculates the difference of the approximate geocentric, apparent azimuth of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard rise time of the Moon. See [Standard rise time of the Moon], page 179, for further details. |
| `H` | `)` | Calculates the difference of the approximate geocentric, apparent azimuth of Moon and Sun (delta), at which the Moon is used as the reference point, in degrees and arcminutes as it happen at standard set time of the Moon. See [Standard set time of the Moon], page 179, for further details. |
| `I` | `(, )` | Calculates the difference of the approximate astronomical midnight times of Moon and Sun (delta), at which the Moon is used as the reference point, as time value in hours and minutes as it happen at astronomical midnight time of the Moon. Results with a negative sign signify that the astronomical midnight time of the Moon is earlier than the astronomical midnight time of the Sun; thus the lunar midnight is before the solar midnight. Results with a positive sign signify that the astronomical midnight time of the Moon is later than the astronomical midnight time of the Sun; thus the lunar midnight is after the solar midnight. See [Astronomical midnight time of the Moon], page 176, and [Astronomical midnight time of the Sun], page 155, for further details. |
| `J` | `(, )` | Calculates the difference of the approximate astronomical noon times of Moon and Sun (delta), at which the Moon is used as the reference point, as time value in hours and minutes as it happen at astronomical noon time of the Moon. Results with a negative sign signify that the astronomical noon time of the Moon is earlier than the astronomical noon time of the Sun; thus the lunar noon is before the solar noon. Results with a positive sign signify that the astronomical noon time of the Moon is later than the astronomical noon time of the Sun; thus the lunar noon is after the solar noon. See [Astronomical noon time of the Moon], page 176, and [Astronomical noon time of the Sun], page 156, for further details. |

`K` `(` Calculates the difference of the approximate standard rise times of Moon and Sun (delta), at which the Moon is used as the reference point, as time value in hours and minutes as it happen at standard rise time of the Moon. Results with a negative sign signify that the standard rise time of the Moon is earlier than the standard rise time of the Sun; thus the moonrise is before the sunrise. Results with a positive sign signify that the standard rise time of the Moon is later than the standard rise time of the Sun; thus the moonrise is after the sunrise. See [Standard rise time of the Moon], page 179, and [Standard rise time of the Sun], page 158, for further details.

`K` `)` Calculates the difference of the approximate standard set times of Moon and Sun (delta), at which the Moon is used as the reference point, as time value in hours and minutes as it happen at standard set time of the Moon. Results with a negative sign signify that the standard set time of the Moon is earlier than the standard set time of the Sun; thus the moonset is before the sunset. Results with a positive sign signify that the standard set time of the Moon is later than the standard set time of the Sun; thus the moonset is after the sunset. See [Standard set time of the Moon], page 179, and [Standard set time of the Sun], page 158, for further details.

`L` `(, )` Calculates the approximate topocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at astronomical midnight time of the Moon (topocentric *midnight height*). See [Astronomical midnight time of the Moon], page 176, for further details.

`M` `(, )` Calculates the approximate topocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at astronomical midnight time of the Moon (topocentric *midnight phase angle*). See [Astronomical midnight time of the Moon], page 176, for further details.

`N` `(, )` Calculates the approximate topocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at astronomical noon time of the Moon (topocentric *noon height*). See [Astronomical noon time of the Moon], page 176, for further details.

`O` `(, )` Calculates the approximate topocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at astronomical noon time of the Moon (topocentric *noon phase angle*). See [Astronomical noon time of the Moon], page 176, for further details.

`P` `(` Calculates the approximate topocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at standard rise time of the Moon (topocentric *rise height*). See [Standard rise time of the Moon], page 179, for further details.

`P` `)`

Calculates the approximate topocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at standard set time of the Moon (topocentric *set height*). See [Standard set time of the Moon], page 179, for further details.

`Q` `(`

Calculates the approximate topocentric, apparent azimuth of the Moon in degrees and arcminutes as it happen at standard rise time of the Moon (topocentric *rise azimuth*). The horizontal angular distance between the topocentric rise azimuth and the East direction is also known as the topocentric *rise width* of the Moon. See [Standard rise time of the Moon], page 179, for further details.

`Q` `)`

Calculates the approximate topocentric, apparent azimuth of the Moon in degrees and arcminutes as it happen at standard set time of the Moon (topocentric *set azimuth*). The horizontal angular distance between the topocentric set azimuth and the West direction is also known as the topocentric *set width* of the Moon. See [Standard set time of the Moon], page 179, for further details.

`R` `(`

Calculates the approximate topocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at standard rise time of the Moon (topocentric *rise phase angle*). See [Standard rise time of the Moon], page 179, for further details.

`R` `)`

Calculates the approximate topocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at standard set time of the Moon (topocentric *set phase angle*). See [Standard set time of the Moon], page 179, for further details.

`S` `(, )`

Calculates the approximate geocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at astronomical midnight time of the Moon (geocentric *midnight height*). See [Astronomical midnight time of the Moon], page 176, for further details.

`T` `(, )`

Calculates the approximate geocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at astronomical midnight time of the Moon (geocentric *midnight phase angle*). See [Astronomical midnight time of the Moon], page 176, for further details.

`U` `(, )`

Calculates the approximate geocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at astronomical noon time of the Moon (geocentric *noon height*). See [Astronomical noon time of the Moon], page 176, for further details.

`V` `(, )`

Calculates the approximate geocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at astronomical noon time of the Moon (geocentric *noon phase angle*). See [Astronomical noon time of the Moon], page 176, for further details.

`W` `(` Calculates the approximate geocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at standard rise time of the Moon (geocentric *rise height*). See [Standard rise time of the Moon], page 179, for further details.

`W` `)` Calculates the approximate geocentric, apparent elevation of the Moon in degrees and arcminutes as it happen at standard set time of the Moon (geocentric *set height*). See [Standard set time of the Moon], page 179, for further details.

`X` `(` Calculates the approximate geocentric, apparent azimuth of the Moon in degrees and arcminutes as it happen at standard rise time of the Moon (geocentric *rise azimuth*). The horizontal angular distance between the geocentric rise azimuth and the East direction is also known as the geocentric *rise width* of the Moon. See [Standard rise time of the Moon], page 179, for further details.

`X` `)` Calculates the approximate geocentric, apparent azimuth of the Moon in degrees and arcminutes as it happen at standard set time of the Moon (geocentric *set azimuth*). The horizontal angular distance between the geocentric set azimuth and the West direction is also known as the geocentric *set width* of the Moon. See [Standard set time of the Moon], page 179, for further details.

`Y` `(` Calculates the approximate geocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at standard rise time of the Moon (geocentric *rise phase angle*). See [Standard rise time of the Moon], page 179, for further details.

`Y` `)` Calculates the approximate geocentric, apparent phase angle of the Moon in range 0.0. . . 1.0 as it happen at standard set time of the Moon (geocentric *set phase angle*). See [Standard set time of the Moon], page 179, for further details.

If no *mode* is given, Gcal automatically uses that mode, which is enabled by the mode character ‘`5`’. If a mode character is given that is not according to one of the ‘`0`’. . . ‘`9`’, ‘`a`’. . . ‘`z`’ and ‘`A`’. . . ‘`Y`’ characters, Gcal also automatically uses that mode, which is enabled by the mode character ‘`5`’.

Depending on the selected mode, Gcal represents the Moon oriented special texts using the same types and styles as they are used by the Sun oriented special texts, these are analogously valid! See [x-Sun-representation], page 169, for the detailed description of the different types of representation used by the Sun oriented special texts, which are likewise valid for the Moon oriented special texts.

The *argument* the Moon oriented special texts must have is exactly equivalent the argument the Sun oriented special texts must have! See [Arguments of the Sun oriented special texts], page 171, for the detailed description of the components of the *argument* which also has to be given to the Moon oriented special texts.

The following table informs you about which type of representation is caused by a mode. The previously defined numbering scheme, as it has been used for the introduction of the types of representation, is used as key value in the column that holds the type of representation. The table also contains a column that shows whether a mode enables dynamical values, i.e. values that are depending on the respective clocktime (if you use the '`--time-offset=argument`' option, you can change the respective clocktime that is used for calculating such values). In a next table column, it is listed whether the given co-ordinate of the location influences the determination of a value, and the last column of the table gives you the information whether a given timezone value affects the values determination:

| **Mode** | **Representation Type** | **Dynamical** | **Co-ordinate** | **Timezone** |
|---|---|---|---|---|
| `0` | 3 | No | Yes | Yes |
| `1` | 3 | No | Yes | Yes |
| `2` | 3 or 4 | No | Yes | Yes |
| `3` | 3 or 4 | No | Yes | Yes |
| `4` | 3 or 4 | No | Yes | Yes |
| `5` | 3 or 4 | No | Yes | Yes |
| `6` | 6 | Yes | Yes | Yes |
| `7` | 6 | Yes | Yes | Yes |
| `8` | 2 | Yes | Yes | Yes |
| `9` | 1 or 1b | Yes | Yes | Yes |
| `a` | 7 | Yes | Yes | Yes |
| `b` | 6 | Yes | Yes | Yes |
| `c` | 7 | Yes | Yes | Yes |
| `d` | 6 | Yes | Yes | Yes |
| `e` | 7 | Yes | Yes | Yes |
| `f` | 4 | Yes | Yes | Yes |
| `g` | 1 or 1a | Yes | Yes | Yes |
| `h` | 6 | Yes | Yes | Yes |
| `i` | 6 | Yes | No | No |
| `j` | 6 | Yes | Yes | Yes |
| `k` | 6 | Yes | Yes | Yes |
| `l` | 2 | Yes | Yes | Yes |
| `m` | 1 or 1b | Yes | Yes | Yes |
| `n` | 7 | Yes | Yes | Yes |
| `o` | 6 | Yes | Yes | Yes |
| `p` | 7 | Yes | Yes | Yes |
| `q` | 6 | Yes | Yes | Yes |
| `r` | 7 | Yes | Yes | Yes |
| `s` | 4 | Yes | Yes | Yes |
| `t` | 1 or 1a | Yes | Yes | Yes |
| `u` | 6 | Yes | Yes | Yes |
| `v` | 2 | Yes | No | No |
| `w` | 3 | Yes | Yes | Yes |
| `x` | 3 | Yes | No | No |
| `y` | 1 | Yes | No | No |
| `z` | 1 | Yes | No | No |
| `A` | 7 | Yes | Yes | Yes |
| `B` | 7 | Yes | Yes | Yes |
| `C` | 7 | Yes | Yes | Yes |
| `D` | 7 | Yes | Yes | Yes |
| `E` | 7 | No | Yes | Yes |
| `F` | 7 | No | Yes | Yes |

| | | | | |
|---|---|---|---|---|
| G | 7 | No | Yes | Yes |
| H | 7 | No | Yes | Yes |
| I | 5 | No | Yes | Yes |
| J | 5 | No | Yes | Yes |
| K | 5 | No | Yes | Yes |
| L | 7 | No | Yes | Yes |
| M | 1 or 1b | No | Yes | Yes |
| N | 7 | No | Yes | Yes |
| O | 1 or 1b | No | Yes | Yes |
| P | 7 | No | Yes | Yes |
| Q | 6 | No | Yes | Yes |
| R | 1 or 1b | No | Yes | Yes |
| S | 7 | No | Yes | Yes |
| T | 1 or 1b | No | Yes | Yes |
| U | 7 | No | Yes | Yes |
| V | 1 or 1b | No | Yes | Yes |
| W | 7 | No | Yes | Yes |
| X | 6 | No | Yes | Yes |
| Y | 1 or 1b | No | Yes | Yes |

And now some examples to these special texts:

The text ‘`Moonrise at %(+5158+00738,120  in MS, BRD`’ will be expanded to
↦ ‘`Moonrise at 12:21 in MS, BRD`’, in case the actual system date is the 1st June 1998.

The text ‘`Moonset at %)*5+5158+00738,120  in MS, BRD`’ will be expanded to
↦ ‘`Moonset at 01:53am in MS, BRD`’, in case the actual system date is the 1st June 1998.

The text ‘`Moon visible %[5+5158+00738,120  in MS, BRD`’ will be expanded to
↦ ‘`Moon visible 13h32' in MS, BRD`’, in case the actual system date is the 1st June 1998.

The text ‘`Moon non-visible %]*+5158+00738,120  in MS, BRD`’ will be expanded to
↦ ‘`Moon non-visible 10.469 in MS, BRD`’, in case the actual system date is the 1st June 1998.

The text ‘`Moon azimuth 0 o'clock=%(*a+5158+00738,120  in MS, BRD`’ will be expanded to
↦ ‘`Moon azimuth 0 o'clock=267d37' in MS, BRD`’, in case the actual system date is the 1st June 1998.

The text ‘`Moonphase %(x+00+000 =%(*m+5158+00738,120 % in MS, BRD`’ will be expanded to
↦ ‘`Moonphase +16h00'=45.248% in MS, BRD`’, in case you call Gcal with the ‘`--time-offset=16:`’ option and the actual system date is the 1st June 1998.

The text ‘`Julian date at %(x+00+000 =%(y+00+000`’ will be expanded to
↦ ‘`Julian date at +10h15'=2450965.927`’, in case you call Gcal with the ‘`--time-offset=10:15`’ option and the actual system date is the 1st June 1998.

Here is a list that reports about the used reference systems in a short manner, describes other aspects that are unmentioned now, and informs about the lacks and limitations that are existing for the Moon oriented special texts:

- All astronomical calculations are based on methods that use —astronomically seen— low-precision elements, which of course is shown in the precision of the results that are caused by them. Nevertheless in most cases, these results can be treated as more as sufficiently precise if one measures their practical use.
- All azimuth and elevation values are always referring to the center of the respective astronomical object, except the case, it is stated otherwise.
- The base point of azimuth values, right ascension values and other ecliptic co-ordinate values is always measured clockwise relative to the geographical, true North (not the magnetic North as shown by a compass), where angular values for the North direction are both denoted as 0 degree and 360 degree.
- All topocentrically based azimuth and elevation values are always corrected by the amount of atmospheric refraction. Such a kind of correction is always done if the center of the respective astronomical object is above 2 degrees below the horizon.
- Generally, the rise and set times of the Moon cannot be precisely predicted, because depending on unpredictable atmospheric conditions, i.e. the local weather conditions as they appear during these times change the amount of atmospheric refraction at the horizon, so that the precalculated rise and set times may throughout be in error by a minute or more.
- The accuracy of rise and set computations decreases again at high latitudes, i.e. within in the lunar Arctic Circle. There, small variations in atmospheric refraction can change the time of moonrise or moonset by many minutes, since the Moon intersects the horizon at a very shallow angle. For the same reason, at high latitudes, the effects of observer height and local topography are magnified and can substantially change the times of the phenomena actually observed, or even whether the phenomena are observed to occur at all.
- It is possible that calculations, which are done for locations within the lunar Arctic Circle, sometimes result in the wrongly **.** text instead of the more precisely ++:++ or –:– texts that are used for marking special events. Unfortunately, therefore, then the reason why the Moon does not rise or set is no longer clearly evident, respectively, it is possible to get the false suggestion that the reason for non-rising or non-setting of the Moon is caused by just the one day within a synodic month, at which the Moon does not rise or set anyhow. Unfortunately, it is just only obvious **that** the Moon does not rise or set during this day.

**Please also note the following references:**

- See [Fixed dates option ‘`--precise`’], page 40, how to obtain a more precise representation of the values which are cause by these special texts.
- And note [Calendar option ‘`--time-offset=`*`argument`*’], page 31, how to change the clocktime, for which the dynamical, i.e. depending on the respective clocktime, Moon oriented special texts are calculated.
- Likewise, see [Fixed dates option ‘`--cycle-end=`*`argument`*’], page 38, how the dynamical, i.e. depending on the respective clocktime, Moon oriented special texts are calculated for a series of clocktimes for the current day.
- Furthermore, see [Fixed dates option ‘`--adjust-value=`*`argument`*’], page 36, how to change the reference altitude that is used to calculate the rise and set times.
- And last, note [Fixed dates option ‘`--atmosphere=`*`air-pressure`*[,*`temperature`*]’], page 37, how to change the base data of the atmosphere that affects the calculation of the Moon oriented special texts.

All Moon oriented special texts **must** always be trailed by a *whitespace* character which is removed in output!

### G.2.2.7 Environment variable %[*format*]-[*argument*] special text

`%`[*`format`*]`-`[*`argument`*] references the contents of an **environment variable**, e.g.:

> The text '`I am '%-USER '-user`' will be expanded to
> ↦ '`I am 'guest'-user`', in case you logged-in as guest on your system.

This special text **must** always be trailed by a *whitespace* character which is removed in output!

### G.2.3 Replacements without any argument %? special texts

Apart from further useful difference values, the text of a fixed date can be provided with different texts used for highlighting.

#### G.2.3.1 Difference value %[*format*]? special texts

The use of the other difference values as listed here is to calculate distance values between two dates. In case these special texts are directly lead by a '`-`' character, e.g. '`-%d`', Gcal switches the sign of the computed value.

You may depreciate the special meaning of the '`-`' character —in case this character itself is needed— by placing a '\' (backslash) character before it, e.g. '`\-`'. If you need the '`\-`' characters themselves, you have to protect the '\' (backslash) character by another '\' (backslash) character, e.g. '`\\-`'.

The following other difference values `%[`*`format`*`]?` special texts are respected:

`%d` Specifies the current respectively queried **day** relative to the actual system date (≡today), e.g.:

> The resource file line '`0 %d days gone`' will be expanded to
> ↦ '`-10 days gone`', in case you call Gcal with the '`-c10-`' option and no command.

`%w` Specifies the current respectively queried **week** relative to the actual system date (≡today). Started weeks are counted as complete weeks.

`%m` Specifies the current respectively queried **month** relative to the actual system date (≡today). Started months are counted as complete months.

`%y` Specifies the current respectively queried **year** relative to the actual system date (≡today). Started years are counted as complete years. This special text **must** always be trailed by a *whitespace* character which is removed in output!

An example:

Supposing, the actual system date is the 4th September 1999. To calculate, how many days, weeks, months and years are between the actual system date and the birthday of a person, who is born on 21st September 1962, Gcal can be called as follows:

```
$ gcal -f/dev/null -Ux -u -#’19620921 %y %m %w %d~’ 1962
⊣
⊣ -37 -444 -1928 -13497
⊣
$ gcal -f/dev/null -Ux -u -#’19620921 \\-%y \-%m %w -%d~’ 1962
⊣
⊣ \--37 --444 -1928 13497
⊣
$ gcal -f/dev/null -Ux -u -#’00000904 %y %m %w %d~’ %19620921 1999
⊣
⊣ 37 444 1928 13497
⊣
$ gcal -f/dev/null -Ux -u -#’00000904 %y %B19620921’ %19620921 1962
⊣
⊣ 37 36
```

Please do not confound the relative year number `%[`*format*`]y` special text with the age value `%B` special text (see [Age value `%[`*format*`]B[`*date*`]` special text], page 148). The `%y` special text counts started years as complete years, while the `%B` special text does not.

#### G.2.3.2 Highlighting %? special texts

The following highlighting `%?` special texts are respected:

`%1` `%1` is replaced by the starting highlighting sequence respectively the starting marking character that is used for highlighting the actual day (see [Global option ‘`--highlighting=`*text*’], page 12). This attains, that all succeeding text of the line after this special text is displayed in the same way as the highlighted respectively marked actual day. `%1` is used together with the `%2` special text, which turns off this enabled highlighting sequence respectively produces the ending marking character. If a `%1` text is not succeeded by a `%2` text on the line, Gcal automatically inserts such a `%2` text at the end of the line.

For example:

```
Only %1THIS%2 word is highlighted in this line.
%1This text is highlighted up to%2 here.
All from %1here up to the end of the line is highlighted.
```

`%2` `%2` is replaced by the ending highlighting sequence respectively the ending marking character that is used for highlighting the actual day. This attains, that a possibly active highlighting according to the actual day is turned off respectively an ending marking character is produced. `%2` is used together with the `%1` special text. The `%2` text has no affect if no preceding `%1` text was found on the line.

`%3` `%3` is replaced by the starting highlighting sequence respectively the starting marking character that is used for highlighting a holiday, (see [Global option ‘`--highlighting=`*text*’], page 12). This attains, that all succeeding text of the line after this special text is displayed in the same way as the highlighted respectively marked holiday. `%3` is used together with the `%4` special text, which turns off this enabled highlighting sequence respectively produces the ending marking character. If a `%3` text is not succeeded by a `%4` text on the line, Gcal automatically inserts such a `%4` text at the end of the line.

For example:

```
Only %3THIS%4 word is highlighted in this line.
%3This text is highlighted up to%4 here.
All from %3here up to the end of the line is highlighted.
```

`%4`
: `%4` is replaced by the ending highlighting sequence respectively the ending marking character that is used for highlighting a holiday. This attains, that a possibly active highlighting according to a holiday is turned off respectively an ending marking character is produced. `%4` is used together with the `%3` special text. The `%4` text has no affect if no preceding `%3` text was found on the line.

`%5`
: `%5` is replaced by the starting highlighting sequence respectively the starting marking character that is used for highlighting the actual day if a fixed date is on today's date, (see [Global option '`--highlighting=text`'], page 12). This attains, that all succeeding text of the line after this special text is displayed in the same way as the highlighted respectively marked actual day. `%5` is used together with the `%6` special text, which turns off this enabled highlighting sequence respectively produces the ending marking character. If a `%5` text is not succeeded by a `%6` text on the line, Gcal automatically inserts such a `%6` text at the end of the line.

`%6`
: `%6` is replaced by the ending highlighting sequence respectively the ending marking character that is used for highlighting the actual day if a fixed date is on today's date. This attains, that a possibly active highlighting according to the actual day is turned off respectively an ending marking character is produced. `%6` is used together with the `%5` special text. The `%6` text has no affect if no preceding `%5` text was found on the line.

`%7`
: `%7` is replaced by the starting highlighting sequence respectively the starting marking character that is used for highlighting a holiday if a fixed date is on a legal holiday date, (see [Global option '`--highlighting=text`'], page 12). This attains, that all succeeding text of the line after this special text is displayed in the same way as the highlighted respectively marked holiday. `%7` is used together with the `%8` special text, which turns off this enabled highlighting sequence respectively produces the ending marking character. If a `%7` text is not succeeded by a `%8` text on the line, Gcal automatically inserts such a `%8` text at the end of the line.

`%8`
: `%8` is replaced by the ending highlighting sequence respectively the ending marking character that is used for highlighting a holiday if a fixed date is on a legal holiday date. This attains, that a possibly active highlighting according to a holiday is turned off respectively an ending marking character is produced. `%8` is used together with the `%7` special text. The `%8` text has no affect if no preceding `%7` text was found on the line.

`%9`
: `%9` is replaced by the starting highlighting sequence respectively the starting marking character that is used for highlighting the actual day if a fixed date is on today's date; otherwise `%9` is replaced by the starting highlighting sequence respectively the starting marking character that is used for highlighting a holiday if a fixed date is on a legal holiday date, (see [Global option '`--highlighting=text`'], page 12). This attains, that all succeeding text of the line after this special text is displayed in the same way as the highlighted respectively marked actual day or holiday. `%9` is used together with the `%0` special text, which turns off this enabled highlighting sequence respectively produces the ending marking character. If a `%9` text is not succeeded by a `%0` text on the line, Gcal automatically inserts such a `%0` text at the end of the line.

`%0`
: `%0` is replaced by the ending highlighting sequence respectively the ending marking character that is used for highlighting the actual day if a fixed date is on today's date; otherwise `%0` is replaced by the ending highlighting sequence respectively the ending marking character that is used for highlighting a holiday if a fixed date is on a legal holiday date. This attains, that a possibly active highlighting according to

today's date or a holiday is turned off respectively an ending marking character is produced. %0 is used together with the %9 special text. The %0 text has no affect if no preceding %9 text was found on the line.

## G.3 Shell command %![*argument*] special text

The *argument* which trails %! until the end of the line, is passed to the shell as a command for further processing. But before, all text variable references and %?... special texts for replacing particular texts which are given in *argument* are interpreted by Gcal. See Section 5.1.9 [Text variables], page 100, and Section G.2 [%?... special texts for text replacement], page 141. All leading text before the %! special text is processed by Gcal as usual. If the simple '--debug' respectively '--debug=internal' option is given at program start-up, informational messages about the executed command and its *exit code* will be shown on the *standard error channel* (see [Global option '--debug=internal'], page 10).

If the '--debug=abort' option is given, the Gcal program will be aborted with an error code in case an *exit code* not equal zero has occurred during the execution of the command. See [Error Code 2], page 239, and [Global option '--debug=abort'], page 11, for further information.

Use the '--execute-command' option if you want to execute all %![*argument*] special texts by the shell instead of seeing them textually only. See [Fixed date option '--execute-command'], page 40, for more details.

# Appendix H Summary of all %?. . . Special Texts

Here is a short-list of all `%?`. . . special texts which can be used in the text part of Gcal resource file line. Of course, not all possibilities concerning the representation —which are offered by the format instruction— are listed here for those special texts which expanded representation may be modified by an optional *format*, but only some few exemplary. See Appendix J [Format Instruction], page 231, for further details.

`%i[`*date*`][#[`*date*`]]`
Exclusion of all non-inclusive date periods

`%e[`*date*`][#[`*date*`]]`
Exclusion of all exclusive date periods

`%?` Inclusive day period

| | |
|---|---|
| `%v` | Exclusion of all non-legal holidays |
| `%x` | Exclusion of all non-holidays |
| `%a` | Exclusion of all non-Mondays |
| `%c` | Exclusion of all non-Tuesdays |
| `%f` | Exclusion of all non-Wednesdays |
| `%g` | Exclusion of all non-Thursdays |
| `%h` | Exclusion of all non-Fridays |
| `%l` | Exclusion of all non-Saturdays |
| `%p` | Exclusion of all non-Sundays |
| `%q` | Exclusion of all non-Mondays. . . Thursdays |
| `%r` | Exclusion of all non-Mondays. . . Fridays |

`%?` Exclusive day period

| | |
|---|---|
| `%V` | Exclusion of all legal holidays |
| `%X` | Exclusion of all holidays |
| `%A` | Exclusion of all Mondays |
| `%C` | Exclusion of all Tuesdays |
| `%F` | Exclusion of all Wednesdays |
| `%G` | Exclusion of all Thursdays |
| `%H` | Exclusion of all Fridays |
| `%L` | Exclusion of all Saturdays |
| `%P` | Exclusion of all Sundays |
| `%Q` | Exclusion of all Mondays. . . Thursdays |
| `%R` | Exclusion of all Mondays. . . Fridays |

%[*format*]K[*date*] weekday name

- %K[*date*] Complete weekday name
- %>3#K[*date*]
  3-Letter weekday name
- %>2#K[*date*]
  2-Letter weekday name

%[*format*]*?*[*date*] weekday number

- %W[*date*] Weekday number, Mon≡1. . .Sun≡7
- %>1&*W[*date*]
  Weekday number, Mon≡1. . .Sun≡7, Ordinal Number suffix
- %E[*date*] Weekday number, Mon≡0. . .Sun≡6
- %>1&*E[*date*]
  Weekday number, Mon≡0. . .Sun≡6, Ordinal Number suffix
- %I[*date*] Weekday number, Sun≡1. . .Sat≡7
- %>1&*I[*date*]
  Weekday number, Sun≡1. . .Sat≡7, Ordinal Number suffix
- %J[*date*] Weekday number, Sun≡0. . .Sat≡6
- %>1&*J[*date*]
  Weekday number, Sun≡0. . .Sat≡6, Ordinal Number suffix
- %S[*date*] Weekday number, *starting day of week*≡1
- %>1&*S[*date*]
  Weekday number, *starting day of week*≡1, Ordinal Number suffix
- %T[*date*] Weekday number, *starting day of week*≡0
- %>1&*T[*date*]
  Weekday number, *starting day of week*≡0, Ordinal Number suffix

%[*format*]N[*date*] day number

- %N[*date*] Day-of-year number
- %>03*N[*date*]
  Day-of-year number, 3-digits with leading zeroes
- %>1&*N[*date*]
  Day-of-year number, Ordinal Number suffix
- %>03&*N[*date*]
  Day-of-year number, 3-digits with leading zeroes, Ordinal Number suffix

%[*format*]D[*date*] day number

- %D[*date*] Day number
- %>02*D[*date*]
  Day number, 2-digits with leading zero
- %>1&*D[*date*]
  Day number, Ordinal Number suffix
- %>02&*D[*date*]
  Day number, 2-digits with leading zero, Ordinal Number suffix

`%k[date]` ISO week number or standard week number, either 2-digits with leading zero, or 4-alphanumeric characters text

`%[format]U[date]` month name

- `%U[date]` Complete month name
- `%>3#U[date]` 3-Letter month name

`%[format]M[date]` month

- `%M[date]` Month number
- `%>02*M[date]` Month number, 2-digits with leading zero
- `%>1&*M[date]` Month number, Ordinal Number suffix
- `%>02&*M[date]` Month number, 2-digits with leading zero, Ordinal Number suffix

`%[format]Y[date]` year number

- `%Y[date]` Year number
- `%>04*Y[date]` Complete year number, 4-digits with leading zeroes

`%[format]?[date]` Moon phase

- `%O[date]` Moon phase text
- `%>03*O[date]` Moon phase text, 3-digits with leading zeroes
- `%Z[date]` Moon phase text graphics

`%?[date]` biorhythm

- `%,[date]` Biorhythm text (series of values)
- `%;[date]` Biorhythm text graphics (bar)

`%[format]?[date]` difference value

- `%ydate` Relative year number
- `%b[date]` Age value (birthday)
- `%B[date]` Age value (birthday), Ordinal Number suffix

`%[format]?[argument]` actual clocktime

- `%t[*][[+|-]mmmm|hh:[mm]]` Actual local time in *hh*:*mm* format
- `%[format]'[[+|-]mmmm|hh:[mm]]` Actual local time in $m[m\ldots]$ format
- `%[format]_[*][[+|-]mmmm|hh:[mm]]` Actual local time hour in $h[h\ldots]$ format
- `%[format]?[[+|-]mmmm|hh:[mm]]` Actual local time minute in $m[m\ldots]$ format
- `%[format]{[[+|-]mmmm|hh:[mm]]` Actual local time 12-hour format suffix

%@[*][[+|-]*mmmm*|*hh*:[*mm*]]
Actual Universal time in *hh:mm/date* format

%[*format*]`[[+|-]*mmmm*|*hh*:[*mm*]]
Actual Universal time in *m*[*m*...]/*date* format

%[*format*].[*][[+|-]*mmmm*|*hh*:[*mm*]]
Actual Universal time hour in *h*[*h*...] format

%[*format*]/[[+|-]*mmmm*|*hh*:[*mm*]]
Actual Universal time minute in *m*[*m*...] format

%[*format*]}[[+|-]*mmmm*|*hh*:[*mm*]]
Actual Universal time 12-hour format suffix

%"[[+|-]*mmmm*|*hh*:[*mm*]]
Numerical Universal time timezone offset value

%=[[+|-]*mmmm*|*hh*:[*mm*]]
Actual numerical local time timezone offset value

%n[[+|-]*n*] Textual date in fixed '%>02*D-%>3#U-%>04*Y' format

%[*format*]j[[+|-]*n*]
Julian day number

%[*format*]b[*][*mode*]*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-co-ordinate-2*
geographical distance and course angle

%b0*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-co-ordinate-2*
Geographical distance (air line distance) in kilometers

%b*0*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-co-ordinate-2*
Geographical distance (air line distance) in statute miles

%b1*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-coordinate-2*
Course angle (true track) between *ISO-6709:1983-co-ordinate-1* and *ISO-6709:1983-co-ordinate-2* in degrees and arcminutes

%b*1*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-co-ordinate-2*
Course angle (true track) between *ISO-6709:1983-co-ordinate-1* and *ISO-6709:1983-co-ordinate-2* in decimal degrees

%b2*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-co-ordinate-2*
Course angle (true track) between *ISO-6709:1983-co-ordinate-2* and *ISO-6709:1983-co-ordinate-1* in degrees and arcminutes

%b*2*ISO-6709:1983-co-ordinate-1*/*ISO-6709:1983-co-ordinate-2*
Course angle (true track) between *ISO-6709:1983-co-ordinate-2* and *ISO-6709:1983-co-ordinate-1* in decimal degrees

%[*format*]?[*][*mode*]*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]] Sun data

%o0*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s0*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Sun's astronomical midnight time in *hh:mm* *24-hours* format

%o*0*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*0*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Sun's astronomical midnight time in *hh:mm* *12-hours* format with time suffix

%o1ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s1ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Sun's astronomical noon time in hh:mm _24-hours_ format

%o*1ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*1ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Sun's astronomical noon time in hh:mm _12-hours_ format with time suffix

%o2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical rise time of Sun's center in hh:mm _24-hours_ format

%o*2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical rise time of Sun's center in hh:mm _12-hours_ format with time suffix

%s2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical set time of Sun's center in hh:mm _24-hours_ format

%o*2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical set time of Sun's center in hh:mm _12-hours_ format with time suffix

%u2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of geometrical visibility, Sun's center above reference altitude, in hours and minutes in hhhmm' format

%u*2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of geometrical visibility, Sun's center above reference altitude, in decimal hours in hh.h... format

%z2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of geometrical non-visibility, Sun's center below reference altitude, in hours and minutes in hhhmm' format

%z*2ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of geometrical non-visibility, Sun's center below reference altitude, in decimal hours in hh.h... format

%o3ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical rise time of Sun's upper limb in hh:mm _24-hours_ format

%o*3ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical rise time of Sun's upper limb in hh:mm _12-hours_ format with time suffix

%s3ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical set time of Sun's upper limb in hh:mm _24-hours_ format

%s*3ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geometrical set time of Sun's upper limb in hh:mm _12-hours_ format with time suffix

%u3ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of geometrical visibility, Sun's upper limb above reference altitude, in hours and minutes in hhhmm' format

%u*3ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of geometrical visibility, Sun's upper limb above reference altitude, in decimal hours in hh.h... format

%z3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of geometrical non-visibility, Sun's upper limb below reference altitude, in hours and minutes in *hh*h*mm*' format

%z*3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of geometrical non-visibility, Sun's upper limb below reference altitude, in decimal hours in *hh*.*h*... format

%o4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Rise time of Sun's center in *hh*:*mm* *24-hours* format

%o*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Rise time of Sun's center in *hh*:*mm* *12-hours* format with time suffix

%s4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Set time of Sun's center in *hh*:*mm* *24-hours* format

%s*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Set time of Sun's center in *hh*:*mm* *12-hours* format with time suffix

%u4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of visibility, Sun's center above reference altitude, in hours and minutes in *hh*h*mm*' format

%u*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of visibility, Sun's center above reference altitude, in decimal hours in *hh*.*h*... format

%z4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of non-visibility, Sun's center below reference altitude, in hours and minutes in *hh*h*mm*' format

%z*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of non-visibility, Sun's center below reference altitude, in decimal hours in *hh*.*h*... format

%o5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Standard rise time of Sun's upper limb in *hh*:*mm* *24-hours* format

%o*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Standard rise time of Sun's upper limb in *hh*:*mm* *12-hours* format with time suffix

%s5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Standard set time of Sun's upper limb in *hh*:*mm* *24-hours* format

%s*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Standard set time of Sun's upper limb in *hh*:*mm* *12-hours* format with time suffix

%u5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of standard visibility, Sun's upper limb above reference altitude, in hours and minutes in *hh*h*mm*' format

%u*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of standard visibility, Sun's upper limb above reference altitude, in decimal hours in *hh*.*h*... format

%z5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of standard non-visibility, Sun's upper limb below reference altitude, in hours and minutes in *hh*h*mm*' format

%z*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of standard non-visibility, Sun's upper limb below reference altitude, in decimal hours in *hh.h*... format

%o6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Begin time of civil twilight in the morning in *hh*:*mm* *24-hours* format

%o*6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Begin time of civil twilight in the morning in *hh*:*mm* *12-hours* format with time suffix

%s6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
End time of civil twilight in the evening in *hh*:*mm* *24-hours* format

%s*6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
End time of civil twilight in the evening in *hh*:*mm* *12-hours* format with time suffix

%u6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center above reference altitude of civil twilight in hours and minutes in *hh*h*mm*' format

%u*6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center above reference altitude of civil twilight in decimal hours in *hh.h*... format

%z6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center below reference altitude of civil twilight in hours and minutes in *hh*h*mm*' format

%z*6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center below reference altitude of civil twilight in decimal hours in *hh.h*... format

%o7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Begin time of nautical twilight in the morning in *hh*:*mm* *24-hours* format

%o*7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Begin time of nautical twilight in the morning in *hh*:*mm* *12-hours* format with time suffix

%s7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
End time of nautical twilight in the evening in *hh*:*mm* *24-hours* format

%s*7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
End time of nautical twilight in the evening in *hh*:*mm* *12-hours* format with time suffix

%u7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center above reference altitude of nautical twilight in hours and minutes in *hh*h*mm*' format

%u*7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center above reference altitude of nautical twilight in decimal hours in *hh.h*... format

%z7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of Sun's center below reference altitude of nautical twilight in hours and minutes in *hh*h*mm*' format

%z*7ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center below reference altitude of nautical twilight in decimal hours in hh.h... format

%o8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Begin time of amateur-astronomical twilight in the morning in hh:mm *24-hours* format

%o*8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Begin time of amateur-astronomical twilight in the morning in hh:mm *12-hours* format with time suffix

%s8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

End time of amateur-astronomical twilight in the evening in hh:mm *24-hours* format

%s*8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

End time of amateur-astronomical twilight in the evening in hh:mm *12-hours* format with time suffix

%u8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center above reference altitude of amateur-astronomical twilight in hours and minutes in hhhmm' format

%u*8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center above reference altitude of amateur-astronomical twilight in decimal hours in hh.h... format

%z8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center below reference altitude of amateur-astronomical twilight in hours and minutes in hhhmm' format

%z*8ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center below reference altitude of amateur-astronomical twilight in decimal hours in hh.h... format

%o9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Begin time of astronomical twilight in the morning in hh:mm *24-hours* format

%o*9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Begin time of astronomical twilight in the morning in hh:mm *12-hours* format with time suffix

%s9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

End time of astronomical twilight in the evening in hh:mm *24-hours* format

%s*9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

End time of astronomical twilight in the evening in hh:mm *12-hours* format with time suffix

%u9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center above reference altitude of astronomical twilight in hours and minutes in hhhmm' format

%u*9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Period of Sun’s center above reference altitude of astronomical twilight in decimal hours in hh.h... format

%z9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Period of Sun's center below reference altitude of astronomical twilight in hours and minutes in hhhmm' format

%z*9ISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Period of Sun's center below reference altitude of astronomical twilight in decimal hours in hh.h... format

%oaISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%saISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric elevation of Sun in degrees and arcminutes in +|-ddddmm' format

%o*aISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*aISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric elevation of Sun in decimal degrees in +|-ddd.d... format

%obISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%sbISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric azimuth of Sun in degrees and arcminutes in ddddmm' format

%o*bISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*bISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric azimuth of Sun in decimal degrees in ddd.d... format

%ocISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%scISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric declination of Sun in degrees and arcminutes in +|-ddddmm' format

%o*cISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*cISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric declination of Sun in decimal degrees in +|-ddd.d... format

%odISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%sdISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric ecliptic longitude of Sun in degrees and arcminutes in ddddmm' format

%o*dISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*dISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric ecliptic longitude of Sun in decimal degrees in ddd.d... format

%oeISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%seISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric right ascension of Sun as time value in hours and minutes in hhhmm' format

%o*eISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*eISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric right ascension of Sun as time value in decimal hours in hh.h... format

%ofISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%sfISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
: Topocentric distance of Sun in astronomical units in n.n... format

%o*fISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*fISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric distance of Sun in kilometers in n.n... format

%ogISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%sgISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric horizontal parallax of Sun in degrees and arcminutes in ddddmm' format

%o*gISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*gISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric horizontal parallax of Sun in decimal degrees in ddd.d... format

%ohISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%shISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric semidiameter of Sun in degrees and arcminutes in ddddmm' format

%o*hISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*hISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric semidiameter of Sun in decimal degrees in ddd.d... format

%oiISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%siISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Atmospheric refraction in degrees and arcminutes in ddddmm' format

%o*iISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*iISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Atmospheric refraction in decimal degrees in ddd.d... format

%ojISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%sjISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Sun in degrees and arcminutes in +|-ddddmm' format

%o*jISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*jISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Sun in decimal degrees in +|-ddd.d... format

%okISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%skISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric azimuth of Sun in degrees and arcminutes in ddddmm' format

%o*kISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*kISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric azimuth of Sun in decimal degrees in ddd.d... format

%olISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%slISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric declination of Sun in degrees and arcminutes in +|-ddddmm' format

%o*lISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%s*lISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric declination of Sun in decimal degrees in +|-ddd.d... format

%omISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%smISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric ecliptic longitude of Sun in degrees and arcminutes in ddddmm' format

%o*mISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*mISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric ecliptic longitude of Sun in decimal degrees in ddd.d... format

%onISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%snISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric right ascension of Sun as time value in hours and minutes in hhhmm' format

%o*nISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*nISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric right ascension of Sun as time value in decimal hours in hh.h... format

%ooISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%soISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric distance of Sun in astronomical units in n.n... format

%o*oISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*oISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric distance of Sun in kilometers in n.n... format

%opISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%spISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric horizontal parallax of Sun in degrees and arcminutes in ddddmm' format

%o*pISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*pISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric horizontal parallax of Sun in decimal degrees in ddd.d... format

%oqISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%sqISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric semidiameter of Sun in degrees and arcminutes in ddddmm' format

%o*qISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*qISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric semidiameter of Sun in decimal degrees in ddd.d... format

%orISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%srISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%o*rISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*rISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta-t in seconds in +|-n.n... format

%osISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%ssISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Local sidereal time in hh:mm *24-hours* format

%o*sISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]<br>
%s*sISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Local sidereal time in hh:mm *12-hours* format with time suffix

%ot*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%st*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Base time as time value in hours and minutes in *hh*h*mm*' format

%o*t*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*t*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Base time as time value in decimal hours in *hh.h*... format

%ou*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%su*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%o*u*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*u*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Julian date in days in *n.n*... format

%ov*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sv*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%o*v*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*v*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Julian Ephemeris date in days in *n.n*... format

%ow*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sw*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Equation of time as time value in hours and minutes in +|-*hh*h*mm*' format

%o*w*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*w*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Equation of time as time value in decimal hours in +|-*hh.h*... format

%ox*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sx*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Delta topocentric elevation of Sun/Moon in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*x*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*x*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Delta topocentric elevation of Sun/Moon in decimal degrees in +|-*ddd.d*... format

%oy*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sy*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Delta topocentric azimuth of Sun/Moon in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*y*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*y*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Delta topocentric azimuth of Sun/Moon in decimal degrees in +|-*ddd.d*... format

%oz*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sz*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Delta geocentric elevation of Sun/Moon in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*z*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*z*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Delta geocentric elevation of Sun/Moon in decimal degrees in +|-*ddd.d*... format

%oAISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

%sAISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric azimuth of Sun/Moon in degrees and arcminutes in +|-dddmm' format

%o*AISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

%s*AISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric azimuth of Sun/Moon in decimal degrees in +|-ddd.d... format

%oBISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Sun/Moon at standard sunrise time in degrees and arcminutes in +|-dddmm' format

%o*BISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Sun/Moon at standard sunrise time in decimal degrees in +|-ddd.d... format

%sBISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Sun/Moon at standard sunset time in degrees and arcminutes in +|-dddmm' format

%s*BISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Sun/Moon at standard sunset time in decimal degrees in +|-ddd.d... format

%oCISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric azimuth of Sun/Moon at standard sunrise time in degrees and arcminutes in +|-dddmm' format

%o*CISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric azimuth of Sun/Moon at standard sunrise time in decimal degrees in +|-ddd.d... format

%sCISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric azimuth of Sun/Moon at standard sunset time in degrees and arcminutes in +|-dddmm' format

%s*CISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric azimuth of Sun/Moon at standard sunset time in decimal degrees in +|-ddd.d... format

%oDISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric elevation of Sun/Moon at standard sunrise time in degrees and arcminutes in +|-dddmm' format

%o*DISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric elevation of Sun/Moon at standard sunrise time in decimal degrees in +|-ddd.d... format

%sDISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric elevation of Sun/Moon at standard sunset time in degrees and arcminutes in +|-dddmm' format

%s*DISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric elevation of Sun/Moon at standard sunset time in decimal degrees in +|-ddd.d... format

%oEISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric azimuth of Sun/Moon at standard sunrise time in degrees and arcminutes in +|-dddmm' format

%o*E*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta geocentric azimuth of Sun/Moon at standard sunrise time in decimal degrees in +|-*ddd.d*... format

%sE*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta geocentric azimuth of Sun/Moon at standard sunset time in degrees and arcminutes in +|-*ddd*d*mm*' format

%s*E*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta geocentric azimuth of Sun/Moon at standard sunset time in decimal degrees in +|-*ddd.d*... format

%oF*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sF*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta astronomical midnight time of Sun/Moon as time value in hours and minutes in +|-*hh*h*mm*' format

%o*F*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*F*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta astronomical midnight time of Sun/Moon as time value in decimal hours in +|-*hh.h*... format

%oG*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sG*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta astronomical noon of Sun/Moon as time value in hours and minutes in +|-*hh*h*mm*' format

%o*G*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*G*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta astronomical noon of Sun/Moon as time value in decimal hours in +|-*hh.h*... format

%oH*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta time of standard rise of Sun/Moon at standard sunrise time as time value in hours and minutes in +|-*hh*h*mm*' format

%o*H*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta time of standard rise of Sun/Moon at standard sunrise time as time value in decimal hours in +|-*hh.h*... format

%sH*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta time of standard set of Sun/Moon at standard sunset time as time value in hours and minutes in +|-*hh*h*mm*' format

%s*H*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Delta time of standard set of Sun/Moon at standard sunset time as time value in decimal hours in +|-*hh.h*... format

%oI*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sI*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Topocentric elevation of Sun at Sun's astronomical midnight time in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*I*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*I*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Topocentric elevation of Sun at Sun's astronomical midnight time in decimal degrees in +|-*ddd.d*... format

%oJ*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sJ*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Sun at Sun's astronomical noon time in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*J*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*J*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Sun at Sun's astronomical noon time in decimal degrees in +|-*ddd.d*... format

%oK*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Sun at standard sunrise time in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*K*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Sun at standard sunrise time in decimal degrees in +|-*ddd.d*... format

%sK*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Sun at standard sunset time in degrees and arcminutes in +|-*ddd*d*mm*' format

%s*K*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Sun at standard sunset time in decimal degrees in +|-*ddd.d*... format

%oL*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric azimuth of Sun at standard sunrise time in degrees and arcminutes in *ddd*d*mm*' format

%o*L*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric azimuth of Sun at standard sunrise time in decimal degrees in *ddd.d*... format

%sL*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric azimuth of Sun at standard sunset time in degrees and arcminutes in *ddd*d*mm*' format

%s*L*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric azimuth of Sun at standard sunset time in decimal degrees in *ddd.d*... format

%oM*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sM*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Geocentric elevation of Sun at Sun's astronomical midnight time in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*M*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*M*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Geocentric elevation of Sun at Sun's astronomical midnight time in decimal degrees in +|-*ddd.d*... format

%oN*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%sN*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Geocentric elevation of Sun at Sun's astronomical noon time in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*N*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%s*N*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Geocentric elevation of Sun at Sun's astronomical noon time in decimal degrees in +|-*ddd.d*... format

%oO*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric elevation of Sun at standard sunrise time in degrees and arcminutes in +|-*ddd*d*mm*' format

%o*O*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric elevation of Sun at standard sunrise time in decimal degrees in +|-*ddd.d*... format

%sO*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric elevation of Sun at standard sunset time in degrees and arcminutes in +|-*ddd*d*mm*' format

%s*O*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric elevation of Sun at standard sunset time in decimal degrees in +|-*ddd.d*... format

%oP*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric azimuth of Sun at standard sunrise time in degrees and arcminutes in *ddd*d*mm*' format

%o*P*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric azimuth of Sun at standard sunrise time in decimal degrees in *ddd.d*... format

%sP*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric azimuth of Sun at standard sunset time in degrees and arcminutes in *ddd*d*mm*' format

%s*P*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric azimuth of Sun at standard sunset time in decimal degrees in *ddd.d*... format

%oQ*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Single shadow length time in the forenoon in *hh*:*mm* *24-hours* format

%o*Q*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Single shadow length time in the forenoon in *hh*:*mm* *12-hours* format with time suffix

%sQ*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Single shadow length time in the afternoon in *hh*:*mm* *24-hours* format

%s*Q*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Single shadow length time in the afternoon in *hh*:*mm* *12-hours* format with time suffix

%uQ*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of single or shorter shadow length in hours and minutes in *hh*h*mm*' format

%u*Q*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of single or shorter shadow length in decimal hours in *hh.h*... format

%zQ*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of longer than single shadow length in hours and minutes in *hh*h*mm*' format

%z*Q*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
Period of longer than single shadow length in decimal hours in *hh.h*... format

`%oR`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Double shadow length time in the forenoon in *`hh:mm`* *24-hours* format

`%o*R`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Double shadow length time in the forenoon in *`hh:mm`* *12-hours* format with time suffix

`%sR`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Double shadow length time in the afternoon in *`hh:mm`* *24-hours* format

`%s*R`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Double shadow length time in the afternoon in *`hh:mm`* *12-hours* format with time suffix

`%uR`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Period of double or shorter shadow length in hours and minutes in *`hh`*h*`mm`*`'` format

`%u*R`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Period of double or shorter shadow length in decimal hours in *`hh.h`*... format

`%zR`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Period of longer than double shadow length in hours and minutes in *`hh`*h*`mm`*`'` format

`%z*R`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Period of longer than double shadow length in decimal hours in *`hh.h`*... format

`%`[*`format`*]*`?`*[`*`][*`mode`*]*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]] Moon data

`%(0`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
`%)0`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Moon's astronomical midnight time in *`hh:mm`* *24-hours* format

`%(*0`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
`%)*0`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Moon's astronomical midnight time in *`hh:mm`* *12-hours* format with time suffix

`%(1`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
`%)1`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Moon's astronomical noon time in *`hh:mm`* *24-hours* format

`%(*1`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
`%)*1`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Moon's astronomical noon time in *`hh:mm`* *12-hours* format with time suffix

`%(2`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Geometrical rise time of Moon's center in *`hh:mm`* *24-hours* format

`%(*2`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Geometrical rise time of Moon's center in *`hh:mm`* *12-hours* format with time suffix

`%)2`*`ISO-6709:1983-co-ordinate`*[`,`[`+`|`-`]*mmmm*|*`hh`*`:`[*mm*]]
Geometrical set time of Moon's center in *`hh:mm`* *24-hours* format

%(*2*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geometrical set time of Moon's center in *hh*:*mm* *12-hours* format with time suffix

%[2*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical visibility, Moon's center above reference altitude, in hours and minutes in *hh*h*mm*' format

%[*2*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical visibility, Moon's center above reference altitude, in decimal hours in *hh*.*h*... format

%]2*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical non-visibility, Moon's center below reference altitude, in hours and minutes in *hh*h*mm*' format

%]*2*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical non-visibility, Moon's center below reference altitude, in decimal hours in *hh*.*h*... format

%(3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geometrical rise time of Moon's upper limb in *hh*:*mm* *24-hours* format

%(*3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geometrical rise time of Moon's upper limb in *hh*:*mm* *12-hours* format with time suffix

%)3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geometrical set time of Moon's upper limb in *hh*:*mm* *24-hours* format

%)*3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geometrical set time of Moon's upper limb in *hh*:*mm* *12-hours* format with time suffix

%[3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical visibility, Moon's upper limb above reference altitude, in hours and minutes in *hh*h*mm*' format

%[*3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical visibility, Moon's upper limb above reference altitude, in decimal hours in *hh*.*h*... format

%]3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical non-visibility, Moon's upper limb below reference altitude, in hours and minutes in *hh*h*mm*' format

%]*3*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Period of geometrical non-visibility, Moon's upper limb below reference altitude, in decimal hours in *hh*.*h*... format

%(4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Rise time of Moon's center in *hh*:*mm* *24-hours* format

%(*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Rise time of Moon's center in *hh*:*mm* *12-hours* format with time suffix

%)4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Set time of Moon's center in *hh*:*mm* *24-hours* format

%)*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Set time of Moon's center in *hh*:*mm* *12-hours* format with time suffix

%[4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of visibility, Moon's center above reference altitude, in hours and minutes in *hh*h*mm*' format

%[*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of visibility, Moon's center above reference altitude, in decimal hours in *hh.h*... format

%]4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of non-visibility, Moon's center below reference altitude, in hours and minutes in *hh*h*mm*' format

%]*4*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of non-visibility, Moon's center below reference altitude, in decimal hours in *hh.h*... format

%(5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Standard rise time of Moon's upper limb in *hh*:*mm* *24-hours* format

%(*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Standard rise time of Moon's upper limb in *hh*:*mm* *12-hours* format with time suffix

%)5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Standard set time of Moon's upper limb in *hh*:*mm* *24-hours* format

%)*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Standard set time of Moon's upper limb in *hh*:*mm* *12-hours* format with time suffix

%[5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of standard visibility, Moon's upper limb above reference altitude, in hours and minutes in *hh*h*mm*' format

%[*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of standard visibility, Moon's upper limb above reference altitude, in decimal hours in *hh.h*... format

%]5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of standard non-visibility, Moon's upper limb below reference altitude, in hours and minutes in *hh*h*mm*' format

%]*5*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Period of standard non-visibility, Moon's upper limb below reference altitude, in decimal hours in *hh.h*... format

%(6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric horizontal parallax of Moon in degrees and arcminutes in *ddd*d*mm*' format

%(*6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*6*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric horizontal parallax of Moon in decimal degrees in *ddd.d*... format

%(7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric semidiameter of Moon in degrees and arcminutes in *ddd*d*mm*' format

%(*7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*7*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric semidiameter of Moon in decimal degrees in *ddd.d*... format

%(8*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)8*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%(*8*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*8*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric brightness of Moon in magnitude units in +|-*n.n*... format

%(9*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)9*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric phase angle of Moon in range 0.0...1.0 in *n.n*... format

%(*9*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*9*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric phase angle of Moon as percent value in range 0.0...100.0 in *n.n*... format

%(a*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)a*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Moon in degrees and arcminutes in +|-*ddd*d*mm*' format

%(*a*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*a*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric elevation of Moon in decimal degrees in +|-*ddd.d*... format

%(b*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)b*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric azimuth of Moon in degrees and arcminutes in *ddd*d*mm*' format

%(*b*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*b*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric azimuth of Moon in decimal degrees in *ddd.d*... format

%(c*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)c*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric declination of Moon in degrees and arcminutes in +|-*ddd*d*mm*' format

%(*c*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*c*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric declination of Moon in decimal degrees in +|-*ddd.d*... format

%(d*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)d*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric ecliptic longitude of Moon in degrees and arcminutes in *ddd*d*mm*' format

%(*d*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
%)*d*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]

Topocentric ecliptic longitude of Moon in decimal degrees in *ddd.d*... format

%(eISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)eISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric ecliptic latitude of Moon in degrees and arcminutes in +|-ddddmm’ format

%(*eISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*eISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric ecliptic latitude of Moon in decimal degrees in +|-ddd.d... format

%(fISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)fISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric right ascension of Moon as time value in hours and minutes in hhhmm’ format

%(*fISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*fISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric right ascension of Moon as time value in decimal hours in hh.h... format

%(gISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)gISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric distance of Moon in mean Earth equator radii in n.n... format

%(*gISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*gISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric distance of Moon in kilometers in n.n... format

%(hISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)hISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric elongation of Moon in degrees and arcminutes in ddddmm’ format

%(*hISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*hISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric elongation of Moon in decimal degrees in ddd.d... format

%(iISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)iISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Atmospheric refraction in degrees and arcminutes in ddddmm’ format

%(*iISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*iISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Atmospheric refraction in decimal degrees in ddd.d... format

%(jISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)jISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric horizontal parallax of Moon in degrees and arcminutes in ddddmm’ format

%(*jISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*jISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric horizontal parallax of Moon in decimal degrees in ddd.d... format

%(kISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)kISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric semidiameter of Moon in degrees and arcminutes in ddddmm’ format

`%(*k`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*k`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric semidiameter of Moon in decimal degrees in *ddd.d*... format

`%(l`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)l`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%(*l`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*l`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric brightness of Moon in magnitude units in `+|-`*n.n*... format

`%(m`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)m`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric phase angle of Moon in range 0.0...1.0 in *n.n*... format

`%(*m`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*m`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric phase angle of Moon as percent value in range 0.0...100.0 in *n.n*... format

`%(n`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)n`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric elevation of Moon in degrees and arcminutes in `+|-`*ddd*d*mm*' format

`%(*n`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*n`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric elevation of Moon in decimal degrees in `+|-`*ddd.d*... format

`%(o`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)o`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric azimuth of Moon in degrees and arcminutes in *ddd*d*mm*' format

`%(*o`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*o`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric azimuth of Moon in decimal degrees in *ddd.d*... format

`%(p`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)p`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric declination of Moon in degrees and arcminutes in `+|-`*ddd*d*mm*' format

`%(*p`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*p`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric declination of Moon in decimal degrees in `+|-`*ddd.d*... format

`%(q`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)q`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric ecliptic longitude of Moon in degrees and arcminutes in *ddd*d*mm*' format

`%(*q`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
`%)*q`*`ISO-6709:1983-co-ordinate`*[`,`[+|-]*mmmm*|*hh*:[*mm*]]
Geocentric ecliptic longitude of Moon in decimal degrees in *ddd.d*... format

%(rISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)rISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric ecliptic latitude of Moon in degrees and arcminutes in +|-ddddmm' format

%(*rISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*rISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric ecliptic latitude of Moon in decimal degrees in +|-ddd.d... format

%(sISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)sISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric right ascension of Moon as time value in hours and minutes in hhhmm' format

%(*sISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*sISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric right ascension of Moon as time value in decimal hours in hh.h... format

%(tISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)tISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric distance of Moon in mean Earth equator radii in n.n... format

%(*tISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*tISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric distance of Moon in kilometers in n.n... format

%(uISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)uISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric elongation of Moon in degrees and arcminutes in ddddmm' format

%(*uISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*uISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Geocentric elongation of Moon in decimal degrees in ddd.d... format

%(vISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)vISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%(*vISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*vISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta-t in seconds in +|-n.n... format

%(wISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)wISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Local sidereal time in hh:mm *24-hours* format

%(*wISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*wISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Local sidereal time in hh:mm *12-hours* format with time suffix

%(xISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)xISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Base time as time value in hours and minutes in hhhmm' format

%(*xISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*xISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Base time as time value in decimal hours in hh.h... format

%(yISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)yISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%(*yISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*yISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Julian date in days in n.n... format

%(zISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)zISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%(*zISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*zISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Julian Ephemeris date in days in n.n... format

%(AISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)AISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Moon/Sun in degrees and arcminutes in +|-ddddmm' format

%(*AISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*AISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Moon/Sun in decimal degrees in +|-ddd.d... format

%(BISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)BISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric azimuth of Moon/Sun in degrees and arcminutes in +|-ddddmm' format

%(*BISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*BISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric azimuth of Moon/Sun in decimal degrees in +|-ddd.d... format

%(CISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)CISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric elevation of Moon/Sun in degrees and arcminutes in +|-ddddmm' format

%(*CISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*CISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric elevation of Moon/Sun in decimal degrees in +|-ddd.d... format

%(DISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)DISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric azimuth of Moon/Sun in degrees and arcminutes in +|-ddddmm' format

%(*DISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*DISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta geocentric azimuth of Moon/Sun in decimal degrees in +|-ddd.d... format

%(EISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Moon/Sun at standard moonrise time in degrees and arcminutes in +|-ddddmm' format

%(*EISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Delta topocentric elevation of Moon/Sun at standard moonrise time in decimal degrees in +|-ddd.d... format

%)EISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta topocentric elevation of Moon/Sun at standard moonset time in degrees and arcminutes in +|-ddddmm' format

%)*EISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta topocentric elevation of Moon/Sun at standard moonset time in decimal degrees in +|-ddd.d... format

%(FISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta topocentric azimuth of Moon/Sun at standard moonrise time in degrees and arcminutes in +|-ddddmm' format

%(*FISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta topocentric azimuth of Moon/Sun at standard moonrise time in decimal degrees in +|-ddd.d... format

%)FISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta topocentric azimuth of Moon/Sun at standard moonset time in degrees and arcminutes in +|-ddddmm' format

%)*FISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta topocentric azimuth of Moon/Sun at standard moonset time in decimal degrees in +|-ddd.d... format

%(GISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric elevation of Moon/Sun at standard moonrise time in degrees and arcminutes in +|-ddddmm' format

%(*GISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric elevation of Moon/Sun at standard moonrise time in decimal degrees in +|-ddd.d... format

%)GISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric elevation of Moon/Sun at standard moonset time in degrees and arcminutes in +|-ddddmm' format

%)*GISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric elevation of Moon/Sun at standard moonset time in decimal degrees in +|-ddd.d... format

%(HISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric azimuth of Moon/Sun at standard moonrise time in degrees and arcminutes in +|-ddddmm' format

%(*HISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric azimuth of Moon/Sun at standard moonrise time in decimal degrees in +|-ddd.d... format

%)HISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric azimuth of Moon/Sun at standard moonset time in degrees and arcminutes in +|-ddddmm' format

%)*HISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta geocentric azimuth of Moon/Sun at standard moonset time in decimal degrees in +|-ddd.d... format

%(IISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)IISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta astronomical midnight time of Moon/Sun as time value in hours and minutes in +|-hhhmm' format

%(*IISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*IISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta astronomical midnight time of Moon/Sun as time value in decimal hours in +|-hh.h... format

%(JISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)JISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta astronomical noon time of Moon/Sun as time value in hours and minutes in +|-hhhmm’ format

%(*JISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*JISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta astronomical noon time of Moon/Sun as time value in decimal hours in +|-hh.h... format

%(KISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta time standard rise of Moon/Sun at standard moonrise time as time value in hours and minutes in +|-hhhmm’ format

%(*KISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta time standard rise of Moon/Sun at standard moonrise time as time value in decimal hours in +|-hh.h... format

%)KISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta time standard set of of Moon/Sun at standard moonset time as time value in hours and minutes in +|-hhhmm’ format

%)*KISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Delta time standard set of of Moon/Sun at standard moonset time as time value in decimal hours in +|-hh.h... format

%(LISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)LISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Topocentric elevation of Moon at Moon’s astronomical midnight time in degrees and arcminutes in +|-ddddmm’ format

%(*LISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*LISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Topocentric elevation of Moon at Moon’s astronomical midnight time in decimal degrees in +|-ddd.d... format

%(MISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)MISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Topocentric phase angle of Moon at Moon’s astronomical midnight time in range 0.0...1.0 in n.n... format

%(*MISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*MISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Topocentric phase angle of Moon at Moon’s astronomical midnight time as percent value in range 0.0...100.0 in n.n... format

%(NISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)NISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Topocentric elevation of Moon at Moon’s astronomical noon time in degrees and arcminutes in +|-ddddmm’ format

%(*NISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*NISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
Topocentric elevation of Moon at Moon’s astronomical noon time in decimal degrees in +|-ddd.d... format

%(OISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)OISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric phase angle of Moon at Moon's astronomical noon time in range 0.0...1.0 in n.n... format

%(*OISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*OISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric phase angle of Moon at Moon's astronomical noon time as percent value in range 0.0...100.0 in n.n... format

%(KISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric elevation of Moon at standard moonrise time in degrees and arcminutes in +|-ddddmm' format

%(*PISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric elevation of Moon at standard moonrise time in decimal degrees in +|-ddd.d... format

%)PISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric elevation of Moon at standard moonset time in degrees and arcminutes in +|-ddddmm' format

%)*PISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric elevation of Moon at standard moonset time in decimal degrees in +|-ddd.d... format

%(QISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric azimuth of Moon at standard moonrise time in degrees and arcminutes in ddddmm' format

%(*QISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric azimuth of Moon at standard moonrise time in decimal degrees in ddd.d... format

%)QISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric azimuth of Moon at standard moonset time in degrees and arcminutes in ddddmm' format

%)*QISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric azimuth of Moon at standard moonset time in decimal degrees in ddd.d... format

%(RISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric phase angle of Moon at standard moonrise time in range 0.0...1.0 in n.n... format

%(*RISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric phase angle of Moon at standard moonrise time as percent value in range 0.0...100.0 in n.n... format

%)RISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric phase angle of Moon at standard moonset time in range 0.0...1.0 in n.n... format

%)*RISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Topocentric phase angle of Moon at standard moonset time as percent value in range 0.0...100.0 in n.n... format

%(SISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)SISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at Moon's astronomical midnight time in degrees and arcminutes in +|-ddddmm' format

%(*SISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*SISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at Moon's astronomical midnight time in decimal degrees in +|-ddd.d... format

%(TISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)TISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric phase angle of Moon at Moon's astronomical midnight time in range 0.0...1.0 in n.n... format

%(*TISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*TISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric phase angle of Moon at Moon's astronomical midnight time as percent value in range 0.0...100.0 in n.n... format

%(UISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)UISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at Moon's astronomical noon time in degrees and arcminutes in +|-ddddmm' format

%(*UISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*UISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at Moon's astronomical noon time in decimal degrees in +|-ddd.d... format

%(VISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)VISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric phase angle of Moon at Moon's astronomical noon time in range 0.0...1.0 in n.n... format

%(*VISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]
%)*VISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric phase angle of Moon at Moon's astronomical noon time as percent value in range 0.0...100.0 in n.n... format

%(WISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at standard moonrise time in degrees and arcminutes in +|-ddddmm' format

%(*WISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at standard moonrise time in decimal degrees in +|-ddd.d... format

%)WISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at standard moonset time in degrees and arcminutes in +|-ddddmm' format

%)*WISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric elevation of Moon at standard moonset time in decimal degrees in +|-ddd.d... format

%(XISO-6709:1983-co-ordinate[,[+|-]mmmm|hh:[mm]]

Geocentric azimuth of Moon at standard moonrise time in degrees and arcminutes in ddddmm' format

%(*X*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric azimuth of Moon at standard moonrise time in decimal degrees in *ddd.d...* format

%)X*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric azimuth of Moon at standard moonset time in degrees and arcminutes in *ddd*d*mm'* format

%)*X*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric azimuth of Moon at standard moonset time in decimal degrees in *ddd.d...* format

%(Y*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric phase angle of Moon at standard moonrise time in range 0.0...1.0 in *n.n...* format

%(*Y*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric phase angle of Moon at standard moonrise time as percent value in range 0.0...100.0 in *n.n...* format

%)Y*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric phase angle of Moon at standard moonset time in range 0.0...1.0 in *n.n...* format

%)*Y*ISO-6709:1983-co-ordinate*[,[+|-]*mmmm*|*hh*:[*mm*]]
: Geocentric phase angle of Moon at standard moonset time as percent value in range 0.0...100.0 in *n.n...* format

%[*format*]-[*argument*]
: Contents of environment variable

%[*format*]? difference value

| | |
|---|---|
| %d | Relative day number |
| %w | Relative week number |
| %m | Relative month number |
| %y | Relative year number |

%? highlighting

| | |
|---|---|
| %1 | Start of highlighting sequence 1 |
| %2 | End of highlighting sequence 1 |
| %3 | Start of highlighting sequence 2 |
| %4 | End of highlighting sequence 2 |
| %5 | Start of highlighting sequence 1 if a fixed date is on today's date |
| %6 | End of highlighting sequence 1 if a fixed date is on today's date |
| %7 | Start of highlighting sequence 2 if a fixed date is on a legal holiday date |
| %8 | End of highlighting sequence 2 if a fixed date is on a legal holiday date |
| %9 | Start of highlighting sequence 1 if a fixed date is on today's date, otherwise start of highlighting sequence 2 if a fixed date is on a legal holiday date |
| %0 | End of highlighting sequence 1 if a fixed date is on today's date, otherwise end of highlighting sequence 2 if a fixed date is on a legal holiday date |

`%![`*argument*`]`
: Shell executes *argument* as a command

# Appendix I Obsolete Special Texts and Format Elements

Because Gcal 3.01 is able to represent most special texts which may occur in the text part of a resource file line (see Appendix G [Description of all `%?`... Special Texts], page 137), and most of the format elements as they may be used in the date format text of the '`--date-format`' See [Calendar option '`--date-format=de|us|gb|`*text*'], page 33, by using a format instruction, some of them —mainly introduced in the Gcal versions 2.20 and 2.40— are obsolete now. The following complete tables of all these obsolete special texts respectively format elements with their new representation, as used by Gcal 3.01 now, show the changes made.

## I.1 Table of Obsolete Special Texts

This table opposes all obsolete special texts as they may be used in the text part of a resource file line (see Appendix G [Description of all `%?`... Special Texts], page 137), and their new representation, as used by Gcal 3.01 now:

| **Symbolic name** | **Old-style** | | **New-style** |
|---|---|---|---|
| %complete weekday name | `%o` | ≡ | `%K` |
| %3-letter weekday name | `%O` | ≡ | `%>3#K` |
| %2-letter weekday name | `%K` | ≡ | `%>2#K` |
| %weekday number (Mon=1...Sun=7) | `%S` | ≡ | `%W` |
| %weekday number (Mon=1...Sun=7) *ONS* | `%I` | ≡ | `%>1&*W` |
| %weekday number (Mon=0...Sun=6) *ONS* | `%J` | ≡ | `%>1&*E` |
| %weekday number (Sun=1...Sat=7) | `%(` | ≡ | `%I` |
| %weekday number (Sun=1...Sat=7) *ONS* | `%<` | ≡ | `%>1&*I` |
| %weekday number (Sun=0...Sat=6) | `%)` | ≡ | `%J` |
| %weekday number (Sun=0...Sat=6) *ONS* | `%>` | ≡ | `%>1&*J` |
| %weekday number (Mon=1...Sun=7) *FLX* | `%[` | ≡ | `%S` |
| %weekday number (Mon=1...Sun=7) *FLX ONS* | `%{` | ≡ | `%>1&*S` |
| %weekday number (Mon=0...Sun=6) *FLX* | `%]` | ≡ | `%T` |
| %weekday number (Mon=0...Sun=6) *FLX ONS* | `%}` | ≡ | `%>1&*T` |
| %day-of-year number | `%+` | ≡ | `%N` |
| %day-of-year number +*LZ* | `%*` | ≡ | `%>03*N` |
| %day-of-year number *ONS* | `%&` | ≡ | `%>1&*N` |
| %day-of-year number +*LZ ONS* | `%#` | ≡ | `%>03&*N` |
| %day number +*LZ* | `%N` | ≡ | `%>02*D` |
| %day number *ONS* | `%s` | ≡ | `%>1&*D` |
| %day number +*LZ ONS* | `%u` | ≡ | `%>02&*D` |
| %complete month name | `%M` | ≡ | `%U` |
| %3-letter month name | `%T` | ≡ | `%>3#U` |
| %month number | `%U` | ≡ | `%M` |
| %month number +*LZ* | `%W` | ≡ | `%>02*M` |
| %month number *ONS* | `%z` | ≡ | `%>1&*M` |
| %month number +*LZ ONS* | `%Z` | ≡ | `%>02&*M` |
| %complete year number +*LZ* | `%=` | ≡ | `%>04*Y` |
| %age value (birthday) number | `%b` | ≡ | `%B` |
| %age value (birthday) number *ONS* | `%B` | ≡ | `%>1&*B` |
| %moon phase | `%-` | ≡ | `%O` |
| %moon phase +*LZ* | `%_` | ≡ | `%>03*O` |
| %moon phase text graphics image | `%:` | ≡ | `%Z` |

Legend:

| | | |
|---|---|---|
| +*LZ* | ≡ | With leading zero(es) |
| *FLX* | ≡ | Flexible starting day of week (see [Calendar option ‘`--starting-day=`*argument*’], page 30) |
| *ONS* | ≡ | With Ordinal Number suffix |

## I.2 Table of Obsolete Date Format Elements

This table opposes all obsolete date format elements as they may be used in the date format text of the '`--date-format`' option (see [Calendar option '`--date-format=de|us|gb|`*text*'], page 33), and their new representation, as used by Gcal 3.01 now:

| **Symbolic name** | **Old-style** | | **New-style** |
|---|---|---|---|
| %day number | `%d` | ≡ | `%>02*D` |
| %day number +*LS* | `%D` | ≡ | `%>2*D` |
| %day number +*LZ ONS* | `%u` | ≡ | `%02&*D` |
| %day number +*LS ONS* | `%U` | ≡ | `%>2&*D` |
| %complete month name | `%B` | ≡ | `%U` |
| %3-letter month name | `%b` | ≡ | `%<3#U` |
| %month number +*LZ* | `%m` | ≡ | `%>02*M` |
| %month number +*LS* | `%M` | ≡ | `%>2*M` |
| %complete year number +*LZ* | `%y` | ≡ | `%>04*Y` |
| %complete year number +*LZ* | `%Y` | ≡ | `%>4*Y` |
| %last-2-digits year number +*LZ* | `%z` | ≡ | `%>02#Y` |
| %last-2-digits year number +*LS* | `%Z` | ≡ | `%>2#Y` |
| %complete weekday name | `%A` | ≡ | `%K` |
| %3-letter weekday name | `%W` | ≡ | `%<3#K` |
| %2-letter weekday name | `%w` | ≡ | `%<2#K` |

Legend:

| | | |
|---|---|---|
| +*LZ* | ≡ | With leading zero(es) |
| +*LS* | ≡ | With leading space(s) |
| *ONS* | ≡ | With Ordinal Number suffix |

## I.3 Table of Obsolete Date Formats

This table opposes all obsolete date format texts as they may be used in the fixed defaults of the country specific date formats of the '`--date-format`' option for German users, U.S. american users and users in Great Britain (see [Calendar option '`--date-format=de|us|gb|`*text*'], page 33), and their new representation, as used by Gcal 3.01 now:

| **CC** | **Old-style** | | **New-style** |
|---|---|---|---|
| DE | `%w, %1%D%2 %b %y` | ≡ | `%<2#K, %1%>2*D%2 %<3#U %>04*Y` |
| US | `%W, %b %1%U%2 %y` | ≡ | `%<3#K, %<3#U %1%>2&*D%2 %>04*Y` |
| GB | `%W, %1%U%2 %b %y` | ≡ | `%<3#K, %1%>2&*D%2 %<3#U %>04*Y` |

# Appendix J Format Instruction

Gcal 3.01 is able to represent text variables, special texts and date formats especially in a very comfortable and uniform manner by using a format instruction.

Here comes the description of the format instruction components, now called *format* in the further context. In general, the template of a format instruction is:

> *alignment* [*sign*] [*zero*] *width* [*style*] [*suffix*] *format*

If a *format* of above mentioned template is used, its components must be specified in the above mentioned order, at which the definition of the components embraced by '`[ ]`' is of course optional. The single components of the format instruction are assembled as follows:

| **Character** | **Description of the components** |
|---|---|
| | *Alignment* group (exactly one member must be defined): |
| `<` | Field contents is placed at the left margin of the field by using the width *width* |
| `:` | Field contents is placed in centered manner in the field by using the width *width* |
| `>` | Field contents is placed at the right margin of the field by using the width *width* |
| | *Sign* (may be defined): |
| `+` | Numerical value is provided with leading sign |
| | *Zero* (may be defined): |
| `0` | Numerical value is filled with leading zero(es) |
| | *Width* (must be defined): |
| *n* | Field has the width *n* (actual range 1...256). |
| | *Style* group (exactly one member may be defined): |
| `u` | Field contents is converted to upper-case letters |
| `l` | Field contents is converted to lower-case letters |
| `w` | Field contents is converted to capitalized words |
| | *Suffix* (may be defined): |
| `&` | Numerical value is provided with an Ordinal Number suffix |
| | *Format* group (exactly one member must be defined): |
| `*` | Field contents is not cut after position *width* |
| `#` | Field contents is cut after position *width* |

If a given *width* component is greater 256 but less than 999, a field width of 256 is assumed. But if a given *width* component is zero, this interrupts the interpretation phase of Gcal so that the complete *format* is not interpreted anymore and is therefore displayed respectively assigned as is. The same happens if a component is misspelled, given in wrong order, or one of the required components is unspecified.

Furthermore, a *style* component may be specified case-independently. See [Calendar option '`--translate-string=`*`text`*'], page 34, for further annotations how the *style* component is used correctly.

# Appendix K Coding Scheme

The following coding scheme tables explain the relations between the *date part* (*yyyy*. . .) of a resource file line and the modifiers, the short-style option '`-c|C[]`' —which is used for enabling the fixed date feature— respectively the '`--period-of-fixed-dates=`*`argument`*' long-style option may have, i.e. they point out all possible correspondences. This is necessary because Gcal does not respect all entries found in a resource file if a *fixed date* argument is given, it respects only those entries which have a definite relation to the *fixed date* argument to avoid the output of redundant information! See [Fixed dates option '`--period-of-fixed-dates=`*`argument`*'], page 50, for further details.

In the *date part* *yyyy*. . .:

*yyyy* represents any year number (`0001`. . .`9999`).

*mm* represents any month number (`01`. . .`12` or `99`) respectively any textual short month name with length of 3 characters (`Jan`, `Feb`. . .).

*dd* represents any day number (`01`. . .`28`, `29`, `30`, `31` or `99`).

*www* represents any textual short weekday name with length of 2. . .3 characters (`Mon`. . .`Sun`).

*?* represents any case-insensitive single letter.

*n* represents any positive integer number (one or more digits).

## K.1 Coding Scheme Table 1

This table opposes all possible encode variations of a date given in the *date part* of resource file with all fitting modifiers of the short-style option '-c|C[]'.

**Please note:**
The '-c', '-ct', '-cnd', '-c*d|wn[www]', '-c@?[[+|-]n[www]]', '-cmmdd' and '-cmmwwwn' options respect **all** encode variations of the *date part*!

| Date part | Short-style option '-c\|C[]' |
|---|---|
| 00000000 | -cn+\|- |
| 000000dd | -cnw\|+\|-, -cw[+\|-], -cm[+\|-] |
| 0000mm00 | -cn+\|- |
| 0000mmdd | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| yyyy0000 | -cn+\|- |
| yyyy00dd | -cnw\|+\|-, -cw[+\|-], -cm[+\|-] |
| yyyymm00 | -cn+\|- |
| yyyymmdd | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| 000000wwwn | -cnw\|+\|-, -cw[+\|-], -cm[+\|-] |
| 0000mmwwwn | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| yyyy00wwwn | -cnw\|+\|-, -cw[+\|-], -cm[+\|-] |
| yyyymmwwwn | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| 0000*dn[www] | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| yyyy*dn[www] | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| 0000*wn[www] | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| yyyy*wn[www] | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| 0000@?[+\|-]n[www] | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |
| yyyy@?[+\|-]n[www] | -cnw\|+\|-, -cw[+\|-], -cm[+\|-], -cy[+\|-], -clnd\|w\|+\|-, -cl*d\|wn[www], -cl@?[[+\|-]n[www]], -clmmdd, -clmmwwwn |

## K.2 Coding Scheme Table 2

This table opposes all fitting modifiers of the short-style option '-c|C[]' with all possible encode variations of a date given in the *date part* of resource file.

| **Short-style option '-c\|C[]'** | **Date part** |
|---|---|
| -c | |
| -ct | |
| -cnd\|+\|- | |
| -cmmdd | |
| -cmmwwwn | |
| -c*d\|wn[www] | |
| -c@?[[+\|-]n[www]] | 00000000, 0000mm00, yyyy0000, yyyymm00, 000000dd, 0000mmdd, yyyy00dd, yyyymmdd, 000000wwwn, 0000mmwwwn, yyyy00wwwn, yyyymmwwwn, 0000*dn, 0000*dnwww, yyyy*dn, yyyy*dnwww, 0000*wn, 0000*wnwww, yyyy*wn, yyyy*wnwww, 0000@?[+\|-]n, 0000@?[+\|-]nwww, yyyy@?[+\|-]n, yyyy@?[+\|-]nwww |
| -cnw | |
| -cw[+\|-] | 000000dd, 0000mmdd, yyyy00dd, yyyymmdd, 000000wwwn, 0000mmwwwn, yyyy00wwwn, yyyymmwwwn, 0000*dn, 0000*dnwww, yyyy*dn, yyyy*dnwww, 0000*wn, 0000*wnwww, yyyy*wn, yyyy*wnwww, 0000@?[+\|-]n, 0000@?[+\|-]nwww, yyyy@?[+\|-]n, yyyy@?[+\|-]nwww |
| -cm[+\|-] | 000000dd, 0000mmdd, yyyy00dd, yyyymmdd, 000000wwwn, 0000mmwwwn, yyyy00wwwn, yyyymmwwwn, 0000*dn, 0000*dnwww, yyyy*dn, yyyy*dnwww, 0000*wn, 0000*wnwww, yyyy*wn, yyyy*wnwww, 0000@?[+\|-]n, 0000@?[+\|-]nwww, yyyy@?[+\|-]n, yyyy@?[+\|-]nwww |
| -cy[+\|-] | |
| -clnd\|w\|+\|- | |
| -clmmdd | |
| -clmmwwwn | |
| -cl*d\|wn[www] | |
| -cl@?[[+\|-]n[www]] | 0000mmdd, yyyymmdd, 0000mmwwwn, yyyymmwwwn, 0000*dn, 0000*dnwww, yyyy*dn, yyyy*dnwww, 0000*wn, 0000*wnwww, yyyy*wn, yyyy*wnwww, 0000@?[+\|-]n, 0000@?[+\|-]nwww, yyyy@?[+\|-]n, yyyy@?[+\|-]nwww |

# Appendix L Environment Variables

Here is a table of all environment variables which are used by Gcal:

`GCAL`
`GCAL_OPT` Flags which are passed to the Gcal program automatically.

`GCAL_COLUMNS`
`COLUMNS` Sets the number of columns on the screen. Takes precedence over the number of columns specified by the `TERM` environment variable or the system itself.

`GCAL_DATE_FORMAT`
Contains the date format text which defines the ordering of a displayed date. Takes precedence over the default date format text.

`GCAL_LINES`
`LINES` Sets the number of lines on the screen. Takes precedence over the number of lines specified by the `TERM` environment variable or the system itself.

`GCAL_SYS_DATADIR`
Name of a file directory absolute to the *root/$prefix* file directory, where the system-administrator stores common, *public access* resource files (dates files).

`GCAL_USR_DATADIR`
Name of a file directory relative to the `HOME` file directory, where the user stores personal, respectively *private access* resource files (dates files). Takes precedence over `GCAL_SYS_DATADIR`.

`GCALANSI` If defined or set, ANSI escape highlighting sequences are used respectively substituted by default if no other highlighting sequences are defined by command line option or found in the Termcap library.

`GCALPATH` (‘`:`’) colon-separated list of file directories which are searched to find the ‘`.gcalrc`’ file. Takes precedence over `HOME`, `GCAL_USR_DATADIR` and `GCAL_SYS_DATADIR`.

`GCALPROG` Contains the file name of the executable Gcal program that is used by the Tcal program to call the Gcal program. Takes precedence over the file name `gcal` which is burned-in during the compilation step of the Tcal program. The `GCALPROG` environment variable must always be set if the Gcal program is installed under another name than the standard name `gcal`, otherwise Tcal is unable to execute the Gcal program automatically!

`HOME` Name of the user’s home file directory. This is used to find the ‘`.gcalrc`’ file. Takes precedence over `GCAL_USR_DATADIR` and `GCAL_SYS_DATADIR`.

`LANGUAGE` The national language and other cultural-specific conventions used by Gcal at run-time. Takes precedence over `LC_ALL`, `LC_MESSAGES` and `LANG`.

`LC_ALL` The national language and other cultural-specific conventions used by Gcal at run-time. Takes precedence over `LC_MESSAGES` and `LANG`.

`LC_MESSAGES`
The national language and other cultural-specific conventions used by Gcal at run-time. Takes precedence over `LANG`.

`LANG` The national language and other cultural-specific conventions used by Gcal at run-time.

`LOGNAME` The eMail address which is used if the mailing option is wanted. See [Global option ‘`--mail[=`*`address`*`]`’], page 14, for more details.

`MAILPROG` Name of the mailer program which is used if the mailing option is wanted. Takes precedence over the file name `mail` which is burned-in during the compilation step of the Gcal program. See [Global option ‘`--mail[=`*`address`*`]`’], page 14, for more details.

`MAILTO` The eMail address which is used if the mailing option is wanted. Takes precedence over `USER` and `LOGNAME`. See [Global option ‘`--mail[=`*`address`*`]`’], page 14, for more details.

`PAGER` Name of the *external pager* program which is used if the paging option is wanted. See [Global option ‘`--pager`’], page 11, for more details.

`PATH` (‘`:`’) colon-separated list of file directories which are searched to find the *external pager* program. See [Global option ‘`--pager`’], page 11, for more details.

`TERM` The type of terminal on which the Gcal program is being run.

`TXT2GCALPROG`

Contains the file name of the executable Txt2gcal program that is used by the Gcal program to call the Txt2gcal program. Takes precedence over the file name `txt2gcal` which is burned-in during the compilation step of the Gcal program. The `TXT2GCALPROG` environment variable must always be set if the Txt2gcal program is installed under another name than the standard name `txt2gcal`, otherwise Gcal is unable to execute the Txt2gcal program automatically! See Section 5.1.9 [Text variables], page 100, for further details.

`TZ` The timezone information which is added to the date text of sent eMails, created response files and created shell script files.

`USER` The eMail address which is used if the mailing option is wanted. Takes precedence over `LOGNAME`. See [Global option ‘`--mail[=`*`address`*`]`’], page 14, for more details.

# Appendix M Error Codes

Gcal produces the following *exit codes*:

127 Request for help screen, software license or program version (adjustable).

126 Invalid command line argument / option.

125 Year out of range for computing the Easter Sunday’s date.

124 Virtual memory exhausted.

123 Invalid date part in resource file or in include file.

122 Invalid month field entry in resource file or in include file.

121 Invalid day field entry in resource file or in include file.

120 Malformed *#include* statement in resource file or in include file.

119 Illegal recursive/cyclic *#include* statement in resource file or in include file.

118 Resource file or include file not found.

117 Invalid *n*’th weekday field (only `1...5` or `9` valid).

116 Missing *whitespace* separator character between *date part* and *text part*.

115 Response file respectively shell script cannot be written.

114 Invalid date variable operation.

113 Referenced date variable undefined.

112 Invalid date value assigned to date variable.

111 Sending output to eMail address failed.

110 Internal error, C function pipe(), fork(), exec*() or unlink() failed.

109 Error reading resource, include respectively response file.

108 Illegal character found in response file.

107 Internal error, invalid value for the size of an internal table occurred.

106 Invalid date format text specified.

105 Invalid text variable definition.

104 Invalid regular expression search pattern specified.

3 Program aborted by signal.

2 *Exit code* not equal zero returned by external command.

1 Activated ‘`-c|C`[]’, ‘`-f|F` *`name`*[`+...`]’ or ‘`-#` *`line`*’ option caused no output of resource file entries.

0 Normal program termination.

# Appendix N Glossary

Here is an alphabetical list of most terms with short description that are introduced and used in this manual:

*Actual date modifier*
: Gcal uses that date as today's date which is declared by the actual date modifier, instead of using the date that is given by computer's clock as system date.

*Alternation operator*
: The ‘`|`’ metacharacter is used to specify alternatives in a regular expression.

*Amateur-astronomical twilight*
: The time when the center of the Sun passes a reference altitude of 15 degrees below a mathematical horizon.

*Anchor* The ‘`^`’ and ‘`$`’ metacharacters in a regular expression.

*ANSI* The American National Standards Institute. This organization produces many standards, among them the standards for the C and C++ programming languages.

*Asr prayer time*
: The third Islamic prayer time. Begins in the afternoon when the length of the shadow cast by a vertical pole has a definite length, and ends just before sunset.

*Astronomical twilight*
: The time when the center of the Sun passes a reference altitude of 18 degrees below a mathematical horizon.

*Astronomical unit*
: An astronomical unit, abbreviated by ae, is equal to the mean distance of the Sun from the Earth, which is about 149,597,870.691 kilometers.

*Autumnal equinox point on the ecliptic*
: The point where the descending Sun's orbit intersects the celestial equator.

*Azimuth* The horizontal angular distance between the vertical circle, that passes a celestial body, and the North point.

*Biorhythm*
: Ascending and descending curves which show three coherent biological cycles, the physical, the emotional and the intellectual cycle.

*BSD* Short term for *Berkeley Software Distribution*, a Unix operating system variant.

*C* The system programming language that most GNU software is written in.

*Calendar option class*
: The option which control the calendar layout.

*Celestial axis*
: The connecting line which runs between the celestial North and South Pole.

*Celestial body elevation*
: The vertical angular distance between the celestial body and the horizon.

*Celestial equator*
: A celestial great circle, which is vertical to the celestial axis and simultaneously represents the Earth's equator that is projected onto the celestial sphere.

*Celestial Pole*
The northern and southern lengthening of the Earth's rotation axis, respectively, projected on the celestial sphere.

*Character class*
A character class in a regular expression is a special notation for describing lists of characters that have a specific attribute, but where the actual characters themselves can vary from country to country and/or from character set to character set.

*Character list*
Matches any one of the characters that are enclosed in the square brackets in a text, which is processed by a regular expression.

*Character set*
The set of numeric codes used by a computer system to represent the characters (letters, numbers, punctuation, etc.) of a particular country or place. The most common character set in use today is ASCII (American Standard Code for Information Interchange). Many European countries use an extension of ASCII known as ISO-8859-1 (ISO Latin-1).

*Civil Islamic calendar*
The arithmetically determined Islamic lunar calendar.

*Civil midnight time*
In terms of Gcal, this is 0 o'clock, the day change just took part. This clocktime is alternatively also denoted as 24 o'clock, or 12pm.

*Civil twilight*
The time when the center of the Sun passes a reference altitude of 6 degrees below a mathematical horizon.

*Collating element*
A special sequence that can appear in a character lists of a regular expression. This apply to non-ASCII character sets, which can have single symbols that are represented with more than one character, as well as several characters that are equivalent for collating, or sorting, purposes.

*Collating symbol*
A multi-character collating element of a regular expression that is enclosed in '`[.`' and '`.]`' metacharacters.

*Command line argument*
A special text that is mostly given on the command line to run Gcal, and which serves for controlling the program.

*Command* A special type of command line argument which controls the periods Gcal respects.

*Common option class*
The standard options all GNU software should implement at least partially.

*Conjunction*
A celestial body is either behind the Sun or is between Sun and Earth (seen from the Earth), and is therefore non-visible from the Earth. In terms of the Moon, this is the New Moon phase.

*Critical day*
Those days in the biorhythm in which one or more of the biological cycles crosses the zero point. At that time, one's system is said to be in a state of flux and it may be desirable to exhibit caution.

*Cycle mode*
: The dynamical, i.e. depending on the respective clocktime, Sun and Moon oriented special texts are calculated for a series of clocktimes for the current day, i.e. determination is done for any time interval that can be individually defined by the user.

*Date part* That part of a Gcal resource file line that contains the information at which date a fixed date happens.

*Decimal system*
: Base-ten notation, where the digits are `0...9`.

*Declination*
: The vertical angular distance between the center of a celestial body and the celestial equator.

*Delta-t* Delta-t is the difference between the Terrestrial Dynamical time (abbreviated by TDT), that was formerly known as *Ephemeris time* (abbreviated by ET), and the Universal time (UT). Thus, '`delta-t` ≡ `TDT - UT`'.

*Early years*
: In terms of eternal holidays that are determined by Gcal, these are the years within the period AD 30 until the midth of the 20th century.

*Ecliptic latitude*
: The vertical angular distance between the center of a celestial body and the ecliptic.

*Ecliptic longitude*
: The vertical angular distance between the center of a celestial body and the vernal equinox point on the ecliptic.

*Ecliptic* The zodiacal line or Sun's orbit.

*Elongation*
: The horizontal angular distance between the center of a celestial body and the Sun's center, as seen from the Earth.

*Ending date*
: A part of a Gcal special text that defines the last or latest date at which the fixed date may occur.

*Environment*
: A collection of strings, of the form `name=`*value*, that each program has available to it. Users generally place values into the environment in order to provide information to various programs. Typical examples are the environment variables `HOME` and `PATH`.

*Equation of time*
: A correction to be added to the true solar time —as read on a sundial— to obtain the mean solar time.

*Equivalence class*
: A list of equivalent characters of a regular expression that is enclosed in '`[=`' and '`=]`' metacharacters.

*Escape sequence*
: A special sequence of characters used for describing non-printing characters, such as `\E` or `\033` or `\x1B` for the ASCII `ESC` (escape) character itself.

*Evening width*
*Set width* The horizontal angular distance between the set azimuth of a celestial body and the West direction.

*Exclusive day period*
*Exclusive date period*
A Gcal special text that can be specified with the effect, that only those fixed dates are respected and displayed which are not part of a definite period.

*Extended file name*
A name that contains one ore more '`/`' characters, and which denotes a file, whose fixed access path either starts from the root file directory, like '`/foo/bar/file`', or from the actual file directory, like '`./bar/file`' or '`../bar/file`'.

*Extended language code information*
A simple language code information, which additionally has a territory information that is separated by a '`_`', like `en_us` for american English.

*Fajr prayer time*
The first Islamic prayer time. Begins with the beginning of astronomical morning twilight, and ends just before sunrise.

*Fixed date option class*
The options which control the fixed date layout and intensity.

*Format instruction*
Text variables, special texts and date formats can be provided with such an instruction text that results the special representation of the contents of text variables, and the evaluated special texts and date formats.

*FSF*
*Free Software Foundation*
A non-profit organization dedicated to the production and distribution of freely distributable software. It was founded by Richard M. Stallman, the author of the original Emacs editor.

*Geocentric data*
Observations of celestial bodies that are done from the fictitious center of the Earth yield in geocentrically based values.

*Geocentrical horizon*
That fictitious horizontal plane that passes through the Earth's center, orthogonal to the observer's local vertical.

*Geometrical horizon*
*Mathematical horizon*
*Mathematical-geocentric horizon*
A purely geometrically-built horizon which disregards the phenomenon of refraction as it arises in reality by the influence of the Earth's atmosphere.

*Global date variable*
A Gcal date variable, which is defined in the command line. Is visible in all referenced resource files.

*Global option class*
The options which modify the program output.

*Global text variable*
A Gcal text variable, which is defined in the command line. Is visible in all referenced resource files.

*Gnomon* A vertical pole which is used to ascertain the length of a shadow.

*GNU* *GNU's not Unix*. An on-going project of the Free Software Foundation to create a complete, freely distributable, POSIX-compliant computing environment.

*GPL*
*General Public License*
The terms under which Gcal and its source code may be distributed.

*Gregorian calendar*
The calendar which was introduced in AD 1582 by Pope Gregory XIII, that has an average year length of 365.2425 days. This calendar is most respected around the world today.

*Hexadecimal system*
Base 16 notation, where the digits are `0...9` and `A...F`, with '`A`' representing 10, '`B`' representing 11, and so on up to '`F`' for 15.

*Highest culmination point*
The highest point above or below the horizontal plane which transits a celestial body during a day.

*Highlighting sequence*
A control character sequence which cause a color or intensity switch in output text. Typical control character sequences are the ANSI escape sequences which have a leading escape character, and trailing more characters that define the type of the ANSI escape sequence.

*Horizontal parallax*
The horizontal parallax of a celestial body specifies the diameter of the Earth as it is seen from the surface of the celestial body.

*Hybrid calendar mode*
The automatical change from one calendar system to another if there are relations to a date for which the other calendar system is valid. Thus, Gcal automatically changes from the Julian calendar system to the Gregorian calendar system if output is related to dates after the Gregorian Reformation has happened.

*Inclusive day period*
*Inclusive date period*
A Gcal special text that can be specified with the effect, that only those fixed dates are respected and displayed which are part of a definite period.

*Integer number*
A whole number, i.e. a number that does not have a fractional part.

*Interval expression*
One or two numbers inside braces in a regular expression.

*Isha prayer time*
The fifth Islamic prayer time. Begins at the end of the astronomical evening twilight, and ends just before the astronomical midnight time of the Sun.

*ISO* ISO is the abbreviation of *International Organization for Standardization*.

*ISO week number*
The week starts on a Monday, and the first week of a year is the one which includes the first Thursday; equivalently, the one which includes the 4th January.

*Julian Ephemeris date*
A Julian date that is corrected by *delta-t*.

*Julian calendar*

The calendar which was introduced in 46 BC by Julius Caesar and in the later years corrected by emperor Augustus, that has an average year length of 365.25 years. This calendar was replaced by the Gregorian calendar in AD 1582.

*Julian date*
*Julian day number*

A day number that is based on the date 1st January 4713 BCE, which is the starting day zero of a consecutive day counting used in astronomical computations. The real zero of this date is at 12 o'clock Universal time (UTC/GMT); the day does not change at midnight, but at noon Universal time.

*Leap day* The 29th February in the Gregorian calendar.

*Leap year* In the Gregorian calendar, a leap year is any year which number can be divided by 4 without a remainder, and years ending in hundreds are no leap years unless they are divisible by 400. But the Eastern Orthodox churches compute leap years within the Gregorian calendar by using another rule; and that, a leap year is any year which number can be divided by 4 without a remainder, and years ending in hundreds are leap years, if a remainder of 2 or 6 occurs when such a year is divided by 9.

*Legal holiday*

A day people don't have to work — as in most companies, and businesses are closed; like '`New Year's Day`' on January 1st.

*Lists of commands*
*Ranges of commands*

Produce more than one calendar sheet per program run.

*Local date variable*

A Gcal date variable, which is defined in a resource file. Is visible only in this specific resource file, and that file which is included by it.

*Local star time*
*Local sidereal time*

The momentary period which is past between the last upper culmination of the vernal equinox point in the meridian of the observer's location (the momentary hour angle of the vernal equinox point), thus the right ascension of the stars in the observer's meridian at the moment.

*Local text variable*

A Gcal text variable, which is defined in a resource file. Is visible only in this specific resource file, and that file which is included by it.

*Long-style option*

A special —non-standard— style to write an option, e.g. '`--foo`'.

*Lowest culmination point*

The lowest point below or above the horizontal plane which transits a celestial body during a day.

*Lunar Arctic Circle*

The geographical latitude circle that is between about 61.5 and 71.9 decimal degrees, and the geographical pole.

*Lunation*
*Synodic month*

The mean time between two consecutive New Moon phases.

*Maghrib prayer time*
: The forth Islamic prayer time. Begins at sunset, and ends at the beginning of the *Isha* prayer time.

*Magnitude unit*
: Is used to define the brightness of a star, and is a non-metrical value (Latin term *magnitudo*, abbreviated m). The difference between two consecutive magnitudes is 1 to 2.512. Therefore, a star with the brightness of 1m is 2.512 times brighter than a star of 2m. A negative magnitude denotes a very bright star, for example almost -27m for the Sun, whereas the hardly visible planet Pluto has a magnitude of a bit more than +14m. The Full Moon has a visual brightness of about -12m.55.

*Marking character*
: Single, printable characters which lead and trail the output text for marking purposes.

*Mean Earth equator radius*
: The mean radius of Earth at the equator is about 6,378.137 kilometer.

*Mean solar time*
*Mean solar day*
*Mean sun*
: A star day is also known as a *mean solar day*. Because the Sun apparently shifts with respect to the vernal equinox point on the ecliptic due to the Earth's orbit around the Sun, the star day and the true solar day have a different length. As the true Sun namely moves irregularly through the ecliptic, a fictitious *mean Sun* with a symmetrical motion through the celestial equator is used for deriving the mean solar time.

*Meridian*
: Either a geographical longitude or an astronomical vertical circle.

*Metacharacters*
*Regular expression operators*
: Special characters in a regular expression.

*Midnight height*
: The elevation of a celestial body as it happen at astronomical midnight time of the celestial body.

*Midnight phase angle*
: The phase angle of a celestial body as it happen at astronomical midnight time of the celestial body.

*Modifier*
: One or more characters which directly trail a short-style option, and which define a special mode of operation for this option.

*Morning width*
*Rise width*
: The horizontal angular distance between the rise azimuth of a celestial body and the East direction.

*Nautical twilight*
: The time when the center of the Sun passes a reference altitude of 12 degrees below a mathematical horizon.

*Negated character list*
*Complemented character list*
: Matches any one of the characters, except those in the square brackets, in a text, that is processed by a regular expression.

*Negative day*
Those days in the biorhythm in which one or more of the biological cycles have a negative maximum value. At that time, one's system is said to be in a lessen state.

*Noon height*
The elevation of a celestial body as it happen at astronomical noon time of the celestial body.

*Noon line* The straight line which connects the North and the South point.

*Noon phase angle*
The phase angle of a celestial body as it happen at astronomical noon time of the celestial body.

*Octal system*
Base-eight notation, where the digits are `0...7`.

*Opposition*
The Earth is in one line between the celestial body and the Sun, and therefore, the celestial body is completely visible from the Earth. In terms of the Moon, this is the Full Moon phase.

*Option* A special type of command line argument which controls how Gcal behaves.

*Orthodrome*
A great circles arc that is casted between two points on a surface of a sphere. Is the shortest geodetic connecting line between two points on a sphere.

*Pattern* A pattern is an arbitrary conditional expression against which input is tested. If the condition is satisfied, the pattern is said to *match* the input. A typical pattern might compare the input against a regular expression.

*Phase angle*
The phase angle states the phase of a celestial body, i.e. the illuminated fraction of it as it can be seen from the Earth.

*Pipelining* The output of a program is used as an input data stream for another program.

*Positive day*
Those days in the biorhythm in which one or more of the biological cycles have a positive maximum value. At that time, one's system is said to be in a raised state.

*POSIX* The name for a series of standards being developed by the IEEE that specify a Portable Operating System interface. The *IX* denotes the Unix heritage of these standards.

*Proleptic calendar mode*
This means, Gcal only uses a definite calendar system during a definite period, although there was a change to another calendar system in the historic reality during this definite period.

*Rational number*
*Floating Point number*
This is a number that can have a fractional part.

*RFC* RFC is the abbreviation of *Request for Comments*.

*Redirection*
The output of a program is sent to another device instead of sending it to the *standard output channel*.

*Refraction* Refraction is an optical phenomenon caused by the Earth's atmosphere, which leads to an apparent raising of the location of a celestial body. The amount of refraction increases with growing air pressure and sinking temperature and vice-versa.

*Regexp*
*Regular expression*
A way of describing a set of strings.

*Relative time offset value*
A time offset value which is related to a base time.

*Religious Islamic calendar*
The astronomically determined Islamic lunar calendar.

*Resource file*
A file that essentially contains the fixed date entries.

*Response file*
A file used by Gcal to preload options and commands at program start-up. A response file may contain options and commands, but **no** references to further response files.

*Right ascension*
The horizontal angular distance between the center of a celestial body and the hour circle that passes through the vernal equinox point on the ecliptic.

*Rise azimuth*
The azimuth of a celestial body as it happen at rise time of the celestial body.

*Rise height*
The elevation of a celestial body as it happen at rise time of the celestial body.

*Rise phase angle*
The phase angle of a celestial body as it happen at rise time of the celestial body.

*Set azimuth*
The azimuth of a celestial body as it happen at set time of the celestial body.

*Set height* The elevation of a celestial body as it happen at set time of the celestial body.

*Set phase angle*
The apparent phase angle of a celestial body as it happen at set time of the celestial body.

*Shell* The command processor of the operating system. This operating system program interprets and runs given command lines.

*Short-style option*
The traditional —standard— style to write an option, e.g. '`-x`'.

*Simple file name*
A name that denotes a file, whose access-path either does not start from the root file directory or from the actual file directory, like '`file`' or '`bar/file`'.

*Simple language code information*
A language code that consists of two-letters like `de` for the German language, as it is defined in the ISO-639:1988.

*Single command*
*3-Month mode command*
Produce only one calendar sheet per program run.

*Solar Arctic Circle*

The geographical latitude circle that is between about 66.7 and 65.9 (corrected for atmospheric refraction) decimal degrees, and the geographical pole. For computing the different twilight phases, this latitude circle decreases to about:

- 60.7 decimal degrees for civil twilight.
- 54.7 decimal degrees for nautical twilight.
- 51.7 decimal degrees for amateur-astronomers twilight.
- 48.7 decimal degrees for astronomical twilight.

*Special text*

A text in a Gcal resource file which is started by a '`%`' character and whose purpose is to suppress output of fixed dates in definite cases, either, or to provide them with particular texts respectively to start external commands.

*Summer point on the ecliptic*
*Summer solstice*
*Solstice in summer*

The Sun's orbit crosses the highest point above the celestial equator that is passed during a revolution of the Earth around the Sun.

*Standard week number*

The week starts on the respective starting day of the week, and the days in a new year that are preceding the first starting day of the week are in the last week of the previous year, respectively in week zero of the new year.

*Star day* The period between two consecutive upper culminations of the vernal equinox point on the ecliptic in the meridian of the observer's location.

*Starting date*

A part of a Gcal special text that defines the first or earliest date at which the fixed date may occur.

*Statute mile*

A linear measure, one statute mile is equal to 1.609344 kilometer.

*String* A chain of characters.

*Switch character*

An option is defined by a leading switch character, either the '`-`' or the '`/`' character for traditional short-style options, or '`--`' for mnemonic long-style options; a command may not have a leading switch character.

*Text part* That part of a Gcal resource file line that contains the fixed date text.

*Topocentric data*

Observations of celestial bodies that are done from the surface of the Earth yield in topocentrically based values.

*True solar time*
*True solar day*

A true solar day is the period between two consecutive lower culminations of the Sun. This entity is taken as the base for deriving the true solar time (as it is also shown by a sundial during the day).

*True track*
*Course angle*

The course angle gives the direction, which is the vertex of a great circles arc (Orthodrome) that is casted between two points on a surface of a sphere, at a meridian

(Gcal uses a geoid (rotation ellipsoid) that is calculated taking pattern from the *World Geodetic System*, short W.G.S, that was designed in 1961). Such an arc is the shortest geodetic connecting line between two points on a sphere. Indeed, the course angles on such an arc are changing continually, because the circular line that is casted intersects each meridian with another angle — except the case, that both co-ordinates are referring to exactly the same longitude or to exactly the same latitude.

*Twilight* The interval of time before sunrise and again after sunset, during which natural light is provided by the upper atmosphere that does receive direct sunlight and reflects parts of it towards the Earth's surface.

*Unix* A computer operating system originally developed in the early 1970's at AT&T Bell Laboratories. It initially became popular in universities around the world, and later moved into commercial environments as a software development system and network server system. There are many commercial versions of Unix, as well as several work-alike systems whose source code is freely available (such as Linux, NetBSD, and FreeBSD).

*Vernal equinox point on the ecliptic*
The point where the ascending Sun's orbit intersects the celestial equator.

*Whitespace character*
For example *Tab*, *Space*, *Form-feed* and similar characters.

*Winter point on the ecliptic*
*Winter solstice*
*Solstice in winter*
The Sun's orbit crosses the lowest point below the celestial equator that is passed during a revolution of the Earth around the Sun.

*Word* A sequence of one or more letters, digits, or underscores (‘_’) in a regular expression.

*Zuhr prayer time*
The second Islamic prayer time. Begins after the astronomical noon time of the Sun, and ends at the beginning of the *Asr* prayer time.

# Command Line Argument Index

Here is an alphabetical list of the command line arguments processed by Gcal:

# Environment Variable Index

Here is an alphabetical list of the environment variables checked by Gcal:

# Table Index

Here is an alphabetical list of the most import tables, summaries and other general views that can be found in this manual:

# Concept Index