

DOSFOP Manual

Literate Programming of OPAL programs

Klaus Didrich

Copyright © 1997, 1998, 1999

DOSFOP, the documentation system for OPAL projects, is a *literate* programming tool, i.e. documentation and program code are kept in the same file. Documentation is integrated into OPAL source code by means of special marked comments.

- Provide *hierarchical documentation* for hierarchic software.
- Use information that is *already contained* in the source code.
- Provide documentation on *paper and as hypertext*.
- Take the *programmer's individuality* into account.
- *Low barrier* for initial use.
- *Compatibility with undocumented code*.

This manual contains technical details for the use of DOSFOP, version 1.92g (for changes from older versions, see Appendix D [Changes], page 34).

DOSFOP takes input from three different sources:

- DOSFOP does not directly produce documentation, rather it generates an intermediate output file in the Texinfo language. This intermediate output can then be converted to DVI files for printed output, to INFO files for the GNU info help system and to HTML files for web browsers.

Before documentation can be produced, the user must set up the global configuration (see Section 2.2.1 [Global Configuration], page 6) and the project data base (see Section 2.2.2 [Project Browser], page 9).

1.2 DOSFOP's History

A first prototype of the system was developed by Torsten Klein as his *Studienarbeit* – undergraduate dissertation – and was further enhanced by user-friendly features as part of his master's thesis, both of which were supervised by the author of this manual.

The system was developed in OPAL itself, with the exception of the graphical user interface, which was developed in TCL/Tk.

Since the completion of Torsten Klein's master's thesis, the DOSFOP system has been maintained and developed by Klaus Didrich.

The current version is 1.92g.

2 Using DOSFOP

Dosfop needs no adjustment of environment variables. The variable `DOSFOP`, which was mandatory in former versions, may be set to point to a non-standard DOSFOP installation, if necessary.

DOSFOP requires two directories, both of which are initialized automatically.

DOSFOP The `DOSFOP` subdirectory is used to store intermediate output and administrative data.

doc The `doc` subdirectory contains the generated documentation. If the user does not change the default, the (root) output files are called ‘`doc/out.dvi`’, ‘`doc/out.info`’ and ‘`doc/out.html`’ for DVI, INFO and HTML format, respectively. There is always an additional top-level HTML file ‘`index.html`’. Note that there may be more than one file for INFO and HTML format. See Section 2.2.1.1 [Administrative Data], page 6 to change the name of the generated file.

2.1 Writing Documentation

Documentation is incorporated in OPAL source code in the form of special comments. Every comment that starts with a per cent sign (%) is treated as documentation. All other comments are treated as source code.

The documentation is written in Texinfo, the same language used for documentation of GNU utilities. Those who know \TeX but not Texinfo should refer to the Appendix (see Appendix B [Texinfo for Experienced \TeX Users], page 30).

2.1.1 Project and Subsystem Surveys

There are two ways to include a survey of a project or a subsystem. Either insert the survey into the documentation by configuring or create a file ‘`Survey.texi`’ at toplevel or the path to the subsystem respectively.

2.1.2 Structure Surveys

The first documentation of an OPAL signature part is considered to contain an overview of the structure’s contents, provided it appears before the initial keyword `SIGNATURE`.

2.1.3 Source-Code Documentation

Every comment that starts with a per cent sign(%) is treated as documentation. This applies to line comments and nested comments.

There are five types of source-code documentation available.

Ordinary documentation

Ordinary documentation is any documentation that does not belong to the remaining four categories. Ordinary documentation may contain arbitrary Texinfo text.

Example:

```
-- %This function computes the length of a sequence.
```

Tagged documentation

Tagged documentation (also called *level documentation*) is ordinary documentation with an additional tag. This tag appears in curly brackets immediately after the initial per cent sign.

Example:

```
/* %{optimize} This function needs further attention
   if used with bigger arguments.
*/
```

The name “level documentation” originates in the intention to provide documentation for different ‘levels’ of users. In GRASP the same concept is called *ribbon*.

As a default, tags are ignored (i.e. tagged documentation is treated like ordinary documentation). You may configure globally or for subsystems (see Section 2.2.1.2 [Global Options], page 7) that the level mechanism is turned on. In this case, you must provide a set of tags that are included.

If level handling is turned on, documentation tagged with *tag* is enclosed in a pair of macro calls, @BEGINDOctag and @ENDDOtag. **These macros must be defined by the user!** Macros for the above example might be:

```
@macro BEGINDOctoptimize
@strong{optimize:}%i{
@end macro
@macro ENDDOctoptimize
}
@end macro
```

which would generate the following documentation:

optimize: *This function needs further attention if used with bigger arguments.*

Documentation sectioning

In addition to the structuring provided by DOSFOP, it is useful to insert additional headings into the source code. Headings are marked by between one and four dollar signs ('\$') after the initial per cent sign. The number of dollar signs indicates the sectioning level relative to the structure heading. The heading must be closed by a single dollar sign.

Example:

```
-- %$$$Accessing elements of a sequence$
```

Note that the total number of sectioning levels required by the documentation may be less than that provided by the formatting tool.

The characters used for the heading must be suitable for several tools and may therefore not contain punctuation.

Hidden documentation

Hidden documentation is marked by a minus sign after the initial per cent sign. Unlike OPAL comments, hidden documentation does not appear at all in the generated documentation.

Example:

```
-- %- Check below carefully for typos!
```

Property references

Property references consist of the name of a law enclosed in square brackets. The reference is replaced by the law itself.

The point is that property references may occur in the signature and implementation parts of a structure, and that they are treated correctly even if the property parts are not included in the generated documentation.

Example:

```
FUN ++ : seq ** seq -> seq
-- %[+_assoc]
```

2.2 The Graphical User Interface

The graphical user interface is invoked by the command `dosfop`. The DOSFOP main window provides buttons to start other programs needed for producing documentation.

The components:

File (menu)

Close the DOSFOP main window and the DOSFOP execution window.

Help (menu)

Provide global or context-sensitive help.

DOSFOP: documenting *project* (title)

The *project* is the title given in the global configuration.

Global configuration

Edit administrative data and global switches. See Section 2.2.1 [Global Configuration], page 6.

Project browser

Enter structures and subsystems of the project. Local configuration of subsystems and structures. See Section 2.2.2 [Project Browser], page 9.

Check configuration consistency

Check whether the source code and abstract syntax (interopal) files are present. Result is presented in the DOSFOP execution window.

Start DOSFOP

Start generation of intermediate Texinfo documentation. Progress is presented in the DOSFOP execution window. Errors and warnings encountered during the translation are repeated at the end.

This button does not invoke the consistency check any more.

convert to DVI**convert to Info****convert to HTML**

Start conversion of the intermediate Texinfo documentation into the respective format. Progress is presented in the DOSFOP execution window.

These buttons are disabled if no intermediate documentation is present. They turn red if the intermediate documentation is newer than the output available in the respective format.

DVI previewer**Info hypertext reader****HTML viewer**

Invoke a (pre)viewer for the respective format. These buttons are disabled if the documentation is not available in the appropriate format. There are three environment variables which may be set if the default is not convenient:

XDVI The name of a previewer for DVI files.

INFO The name of a previewer for INFO files.

WWWBROWSER

The name of a browser for HTML files.

All of these are called with file name to be displayed as their argument. Note that all should open a separate window and should consist of a single name (no arguments).

2.2.1 Global Configuration

Global Configuration consists of administrative data and global options for customizing the documentation. On the first call, only the administrative data is displayed; on pressing a button, all the switches are displayed. The configurator is an independent process and is not closed when the DOSFOP main window is exited.

Changes are only effected after the settings have been saved.

2.2.1.1 Administrative Data

File (menu)

Three options for saving the changes and exiting the configurator, for saving the changes, or for discarding the changes and exiting the configurator.

Help (menu)

Invoke a window with global or context-sensitive help.

Root-location of the Project

This is the location where DOSFOP looks for structures at the ‘top level’.

Intermediate code output file

This is the name of the file where DOSFOP stores the intermediate Texinfo output. The default is *root-location/DOSFOP/out.texi*. You may change *out* in order to change the name of the generated files in the *doc* directory (see Chapter 2 [Using DOSFOP], page 3).

Note that this file contains the surveys and the structure of the documentation only. Documentation of structures is located in their associated ‘OCS’ subdirectory.

Top structure name

The name of the top structure of the project, which must be located at the Root-location (see above). DOSFOP will include all structures which are directly or transitively imported by the top structure in the documentation.

You may use the special name *** to indicate that *all structures* contained in the project data base should be included. Note that this set must be closed under the import relation (with the exception of library structures).

Name of the project**Authors’ names****Date of creation**

These entries are used for the title page. The name of the project also appears in the DOSFOP main window.

Display all options

Display switches for global options (see Section 2.2.1.2 [Global Options], page 7).

2.2.1.2 Global Options

Options marked with *SUB* can be configured individually for subsystems, those marked *STR* can be configured individually for structures.

Survey (SUB)

Introduction to the documentation (or the subsystem). Note that this information is ignored, if a file ‘*Survey.texi*’ exists at the time the documentation is generated. In this case, the survey is included from that file.

Macros This feature has been made obsolete by the introduction of macros into Texinfo.

Level mechanism (SUB, STR)

Treatment of tagged or level documentation (see Section 2.1.3 [Source Code Documentation], page 3).

No Level Handling

Tagged documentation is treated as ordinary documentation.

Modify Levels

You may edit a list of tags. Tagged documentation is only included if its tag is an element of this list. Note that you must pro-

vide macros `BEGINDOctag` and `ENDDOctag` (see Section 2.1.3 [Source Code Documentation], page 3).

Functionality index

Generate a functionality index. Identifiers can be annotated with the instantiation, with the instantiation and the origin, or with all components. In fact, this switch controls several indices:

Function Index

Function Index By Domain

Function Index By Codomain

Functions declared in structures of the project sorted alphabetically, by types in their domain and their codomain, respectively.

Sort Index

Sorts declared in structures of the project.

Property Index

Algebraic properties (laws and free type declarations) from structures of the project.

Application Index

This switch controls two indices:

Index to the places where functions are applied. Verbosity is the same as with the functionality index. You may additionally decide to index applications of library functions. See Section 4.3 [Files in the Project Data Base], page 19, for what DOSFOP considers to be a library file.

Definition index

Index to the places where functions are defined. Verbosity is the same as with the functionality index.

Concept index

Insert a concept index. The user must define entries by adding `@cindex entry` lines in the documentation.

Structure index

Alphabetic index of all structures (without libraries).

Subsystem inclusion

Include only top-level structures or top-level structures and all subsystems.

Library inclusion

Include the library files referenced from other files in the generated documentation.

Property inclusion (SUB, STR)

Also include external and (if “include only interfaces” – see below – is off) internal property parts of structures.

Include only interfaces (SUB, STR)

If on, include only signature and (if “property inclusion” – see above – is on) external property parts of structures,

Hierarchy visualization (SUB)

Include graphic visualization of dependencies between subsystems and/or structures of a subsystem.

This option is mainly for use in printed documentation, but works for all three output formats.

Sort structures (SUB)

Structures cannot be sorted by the user. DOSFOP can sort the structures according to their dependencies, top-down or bottom-up, alphabetically, or not at all.

Import referencing (SUB, STR)

Include links to imported structures, with or without library structures.

Used function tables (SUB, STR)

At the end of a section, include a table for every function defined in that section which lists the function it references. For the verbosity, see above: functionality index.

Basic language

Generate English or German section names.

Start new pages (SUB, STR)

Start new pages where necessary.

Drop empty lines (SUB, STR)

Remove empty lines between documentation and source code.

Structure in single node

This option has (almost) no effect on printed output. For INFO and HTML output, all parts and the overview are put into a single node. This is useful when most structures are not documented as a few less mouse clicks are needed to access information.

2.2.2 Project Browser

The project browser is used to edit the project data base, that is, the information as to which structures are part of the project and how are they grouped into subsystems. The project browser always starts at ‘top level’. It is a separate process and not closed when the DOSFOP main window is closed.

Changes to the project data base are immediately effective.

2.2.2.1 Structures

The left-hand side contains a list of the structures of the project that are located in the current subsystem.

Delete The marked structure is deleted.

Insert You may insert a new structure name, either by typing it in or by using the file browser (see Section 2.2.2.3 [Filebrowser], page 10). Click on **Exit** when all structures have been entered.

Rename The marked structure may be given a new name.

Config global

Config Configure each structure of the subsystem or the marked structure individually. See Section 2.2.1.2 [Global Options], page 7, for the available options.

2.2.2.2 Subsystems

The right-hand side contains a list of the subsystems of the current subsystem.

Level up This button – immediately above the list – takes you to the parent subsystem. It is not displayed at the top level.

Delete Delete the marked subsystem.

Insert Insert a new subsystem. The name of the new subsystem (to be used in headings) and its path are required. You may again use the file browser (see Section 2.2.2.3 [Filebrowser], page 10) to facilitate your work. After clicking insert, move the mouse to the list of subsystems and click on the subsystem before which the new subsystem is to be inserted.

Rename Rename the marked subsystem. (You cannot change the path.)

Move Move the marked subsystem. After clicking on the move button, click on the structure before which the marked structure is to be inserted.

Config Edit options for the marked subsystem. See Section 2.2.1.2 [Global Options], page 7, for the available options. Additionally, the path to the subsystem may be changed.

2.2.2.3 File Browser

The file browser can be invoked when inserting either structures or subsystems. It is again a separate process which must be terminated explicitly. You need (and should) not have more than one file browser open at one time.

You can change the current directory by double-clicking on the directory name in the list.

Insertion of structures is done by double-clicking on the file name. In the case of OPAL files, the extension (`‘.sign’`, `‘.impl’`, `‘.extp’`, `‘.intp’`) is removed from the file name. All other file names are left as they are. (see Section 2.6 [DOSFOP and Non-OPAL Programs], page 13). The file is immediately inserted into the structures of the current subsystem.

Insertion of subsystems is done by double-clicking on the directory name *above the list widget*. You must then provide a name for the subsystem and insert it yourself.

2.3 DOSFOP without Windows

DOSFOP can also be invoked without the graphic user interface. This may come in handy if neither the project data base nor the global options have been changed.

The program is called `pureDosfop`. The following options are available:

- addonly* This switch is followed by names of structures which are added to the ‘DOSFOP/only’ file. (see Section 2.7 [Incrementally Generating Documentation], page 14)
- cc* Identical to the button ‘Check Configuration Consistency’.
- clearonly* Remove the ‘DOSFOP/only’ file. (see Section 2.7 [Incrementally Generating Documentation], page 14)
- config* Construct the file ‘**config**’ from its parts. Must be used if the configuration has been changed manually. (See also Section 4.3 [Files in the Project Data Base], page 19.)
- configascii*
- configbin* DOSFOP can read the configuration either in ASCII (readable) format or in binary format (which is stored in ‘DOSFOP/config.bin’. If both files exist, DOSFOP reads in the newer one and writes a new binary configuration.
The switch **-configascii** removes the binary configuration. The switch **-configbin** creates a dummy binary configuration and causes DOSFOP to create a real one in the next run.
The binary configuration can be quite large.
- dosfop* Construct the intermediate Texinfo output.
- dvi*
- info*
- html* Construct the DVI file (INFO file, HTML files) from the intermediate Texinfo output.
- qhtml* Construct the HTML files from the intermediate Texinfo output, using only structures in ‘DOSFOP/only’. This is faster but may result in dangling links and incomplete indices.
- usage* A short usage message.

Calling **pureDosfop** with no option is identical to calling **pureDosfop -dosfop**.

2.4 DOSFOP and OCS

OCS has been extended by some targets which allow the automatic generation of documentation of an OPAL project. The configuration may afterwards be edited with the standard **dosfop** program.

The MAKE mechanism is not well suited for DOSFOP, we had to compromise. For one, documentation is always generated anew and not if the respective source code has changed. Second, an environment variable **PROJECTROOT** must be defined, which contains the absolute path to the top level directory.

- `doc`
Generate HTML documentation for the current project.
- `dvi`
Generate DVI documentation for the current project.

Incremental generation of documentation (see Section 2.7 [Incrementally Generating Documentation], page 14) is supported. If the environment variable `PROJECTROOT` is defined, every time a structure is compiled its name is added to the ‘`DOSFOP/only`’ file. A call ‘`ocs doc`’ or ‘`ocs dvi`’ will then only generate new documentation for the changed files. The ‘`DOSFOP/only`’ file is always removed afterwards.

DOSFOP often uses a large amount of resources (even for nowadays’ computers), and it may happen that documentation cannot be generated at once. The following targets help in these cases.

- `thisdoc`
Write the structures of the current subsystem to ‘`DOSFOP/only`’ and generate documentation for these structures.
- `thisdocall`
Perform `thisdoc` for current system and all subsystems.

The calls ‘`ocs doc`’ and ‘`ocs thisdocall`’ have similar effects. The latter one calls `dosfop` for every subsystem and may not correctly compute the hierarchy visualization correctly. On the other hand, it often works, where ‘`ocs doc`’ fails for lack of resources.

2.5 DOSFOP and OASYS

If OASYS is used, the usage of DOSFOP is made much simpler. Simply execute the line
`<DOSFOP path>/tc14/dfo.tcl`

from the OASYS command line or include this line in your ‘`.oasysrc`’ file. You need not change your `PATH` variable. The DOSFOP distribution is searched for as follows:

1. If the environment variable `DOSFOP` is set, its contents are used.
2. If the environment variable `OCSDIR` is set, DOSFOP is sought under ‘`$OCSDIR/dosfop`’.
- 3.

If neither `DOSFOP` nor `OCSDIR` are defined, ‘`/usr/ocs/dosfop`’ is used.

Within OASYS, four additional commands are available:

- `doc-dvi`
- `doc-info`
- `doc-html` These commands construct the DOSFOP data base from the OASYS data base, invoke the DOSFOP translation to construct the intermediate Texinfo output, and finally convert Texinfo to the desired output format. Only the necessary steps are performed.
- `doc-help` Provide a brief introduction to these commands.

2.6 DOSFOP and Non-OPAL Programs

DOSFOP can incorporate non-OPAL sources. To use this feature, create a file ‘dosfop.filetypes’ in the DOSFOP directory with the following content:

- Empty lines and lines starting with # are treated as comments.
- Other lines must contain six components separated by colons. Empty components must be indicated by one space. The six components have the following semantics:

Suffix Any file with this suffix will be regarded as a foreign file.

Implementation suffix

Suffix of the corresponding implementation file. Empty if no implementation exists.

Line comment starter

Empty if no line comments are available.

Nested comment starter

Empty if no nested comments are available. Subsequent documentation from line comments is then merged.

Nested comment ender

Empty if no nested comments are available.

Documentation designator

String which distinguishes documentation from comments.

The file ‘dosfop.filetypes’ is searched in the given order and before an OPAL signature file is searched.

Example file:

```
# almost standard OPAL (note leading space for documentation designator!)
.sign:.impl:--:/*:*/: %:
# C files (assuming .h to be the interface for .c)
# empty documentation designator -> all comments are documentation
.h:.c: :/*:*/: :
# Tcl files
.tcl: :#: : : :
# SysDefs file
SysDefs: :#: : : :
# Modula-2 files
.md:.mi: :(*:*): :
```

Note that scanning still obeys the OPAL rules. For example, /* is not regarded as a valid comment starter for ‘.h’ or ‘.c’ files if followed by a special character. In particular, /*****/ is *not* recognized as a comment.

Other problems might occur if string constants are denoted differently than in OPAL, but so far we have had no problems in this regard.

You should make sure that DOSFOP does not search for structure parts that do not exist (property inclusion should be off; only interfaces should be off if no implementation exists). ‘Check Consistency’ has not been adapted yet and will complain if you include foreign structures.

DOSFOP puts the intermediate output for structure located at *path* into a subdirectory *path*‘/OCS’. You must ensure that this directory exists.

DOSFOP can incorporate import references and indices generated by other programs:

For file ‘foo’, the imported structures are contained in ‘DOSFOP/foo.deps’ (‘foo’ includes the suffix), each line containing one structure name.

The index entries have to be given in file ‘DOSFOP/foo.index’. Every line has the following format

<line number>:<index generating command>:<index entry>

Line numbers start with zero! The command for generating the index entry could for example be @findex or @pindex. The index entry may contain other colons. Escaping of Texinfo special characters is done by DOSFOP.

The files ‘DOSFOP/foo.{deps,index}’ are not mandatory.

2.7 Incrementally Generating Documentation

DOSFOP can generate documentation incrementally. To use this feature, create a file ‘only’ in the ‘DOSFOP’ subdirectory which contains the names of structures (one in each line), for which documentation shall be generated. The same name may occur several times in the file.

If the file ‘DOSFOP/only’ exists, DOSFOP will generate the ‘DOSFOP/out.texi’ file and the documentation files for the structures named in the ‘only’ file.

There is a switch -qhtml for the program pureDosfop. The generation of HTML files will in this case cover the main file, the indices and the files associated with structures referred to in ‘only’. Note that this may produce dangling links and incomplete indices.

3 The Texinfo Environment for DOSFOP

3.1 T_EX macros

The following macro packages are available for use in printed manuals:

dosfopLatin1.sty

Characters from the ISO-8859-1 (latin1) character set can be used directly for DVI, for INFO and for HTML files.

psfig.tex Macros to include postscript figures.

pstricks.tex

The pstricks macro package is included.

pst-node.tex

Macros to draw nodes in graphics for the **pstricks** package.

Some other macros are defined for the construction of used function tables and the table of contents. Indentation and skip between paragraphs are defined.

3.2 Texinfo setup

The following switches are set:

```
@setchapternewpage off
@finalout
```

These indices are used by DOSFOP: **st**, **ap**, **cd**, **dm**, **pr**. The predefined indices are also reserved for use by DOSFOP. Indices starting with the letters ‘x’, ‘y’ and ‘z’ will not be used by DOSFOP.

The variable **@dfStarLine** is used. Further variables will all use the prefix ‘**df**’.

3.3 Predefined Macros

The following macros may be used to reference structure parts in **@ref**, **@xref** or **@pxref** commands:

```
@Sign{structure name}
@Impl{structure name}
@Extp{structure name}
@Intp{structure name}
@LibSign{structure name}
@LibExtp{structure name}
```

For other predefined macros which are not designed to be used in documentation, see Section 4.1 [TeX and Texinfo Related Files], page 18.

3.4 Switches

Some options are translated to variables:

`html`

Set iff HTML output is produced. (see Section 3.5 [Conditionally Visible Text], page 16)

`colon`

Set to ‘:’ if DVI or HTML output is produced; set to ‘;’ if INFO is produced. This is used to overcome difficulties with info’s inability to include colons in entries.

`structureindexflag`

Set iff a structure index is part of the generated documentation.

`dfDomainIndexFlag`

Set iff a function index by domain is part of the generated documentation.

`dfCodomainIndexFlag`

Set iff a function index by codomain is part of the generated documentation.

`dfEnglish`

Set iff the chosen language is English.

`dfGerman`

Set iff the chosen language is german.

`dfSingleNode`

Set iff all parts of a structure are to be part of a single node.

`dfProjectName`

Set to the project title.

`dfAuthors`

Set to the authors’ names.

`dfDate`

Set to the date.

`dfSorttop_down`

`dfSortbottom_up`

`dfSortalphabetically`

`dfSortuser_defined`

`dfSortoff`

Set iff the respective option for sorting structures has been chosen.

3.5 Conditionally Visible Text

While most text shall be used for all three output formats, sometimes it is desirable to include text in only one of the formats, or even to pass text untouched by Texinfo directly to \TeX or HTML.

This is possible with the following Texinfo environments:

```
@iftex
@end iftex
```

Text in this environment is processed by Texinfo and then passed on to T_EX and HTML. If this text is not passed to HTML, you may use T_EX commands, except that you must replace the ‘\’ with an ‘@’.

If you want to exclude HTML, enclose this environment in the environment `@ifclear html/@end ifclear`.

```
@tex
@end tex
```

Text in this environment is passed directly on to T_EX, so you may enter plain T_EX here.

```
@ifset html
@end ifset
```

Text in this environment is processed by Texinfo and then passed on to HTML.

```
@ifinfo
@end ifinfo
```

Text in this environment appears only in the INFO output.

```
@ifhtml
@end ifhtml
```

Text in this environment is passed directly on to HTML; you may thus enter HTML tags here.

3.5.1 Macros and Conditions

Instead of putting the condition inside the macro, one should define the macros conditionally:

```
@ifset html
@macro TEXT
This text shows up in HTML.
@end macro
@end ifset
@ifclear html
@macro TEXT
This text shows up in TeX and Info.
@end macro
@end ifclear
```

4 The Files DOSFOP Uses

4.1 TeX- and Texinfo-Related Files

These files are located in the directory `<DOSFOP path>/tex`.

The following files may be modified by the user by placing a copy in the local ‘DOSFOP’ directory.

‘dosfopPrelude.tex’

TeX commands which must be carried out, before ‘texinfo.tex’ is included (which resets some category codes).

‘dosfopPrelude.texi’

Texinfo commands to set up variables, indices and the like. Some files are included from other packages.

‘dosfop.macros’

Texinfo macro definitions. Many of the macros contained here are used to generate headings and node names. You may change these freely (for example, for a new language), but should ensure that all parameters are used when constructing node names, otherwise the uniqueness of node names is no longer guaranteed. The following macros are of particular interest :

`@BEGINOPAL{}`

`@ENDOPAL{}`

Every piece of OPAL source code is surrounded by a call to these macros. The default is `@example/@end example`.

`@BEGINDOC{}`

`@ENDDOC{}`

Every piece of documentation — with the exception of the surveys — is enclosed in a call to these macros. The default is `@value{dfStarLine}` for both, which will put a line consisting of 70 ‘*’ around documentation in INFO output. No effect on DVI or HTML output.

`@BEGINOVW{}`

`@ENDOVW{}`

Every survey (project, subsystem, structure survey) is enclosed in these macros. They are empty by default.

`@FINALIZEDOC{}`

This macro is called at the very end of the documentation. The default definition is `@contents`.

4.2 TeX Styles

These files are included to make the DOSFOP distribution self-contained.

dosfopLatin1.sty

This style file defines T_EX commands to allow usage of characters from the ISO-8859-1 (latin1) character set. These are no longer strictly necessary anymore, because Texinfo provides commands for these characters now.

psfig.sty**psfig.tex (1.9)****pst-node.tex (0.93a)****pstricks.con (0.93a)****pstricks.sty****pstricks.tex (0.93a)**

Some files from the **pstricks** package.

texinfo.tex (2.185)

A very recent version of the Texinfo macro package; DOSFOP will not produce correct DVI output if older versions are used.

4.3 Files in the Project Data Base

The project data base and the global and local options are stored in files in the DOSFOP directory. They may be edited by the user.

‘config’

This file contains the entire project data base and the option settings. It is read in by the DOSFOP translation program and is not affected by changes to other files.

Note that the file ‘**config**’ is frequently rebuilt from the parts. It is therefore not advisable to edit this file — a better option is to edit one of the parts and call **pureDosfop -config** (see Section 2.3 [DOSFOP without Windows], page 10).

‘Toplevel.config’

Contains the administrative data and the global option settings.

‘*subsystem name*.config’

Contains administrative data and option settings for the specific subsystem.

‘*subsystem name*.names’

Contains the names of the subsystems of subsystem *subsystem name*. (Spaces have already been replaced by underscores).

‘*structure name*.config’

Contains the local option settings for this structure.

‘Library.config’

Contains the subsystem structure of what DOSFOP considers the ‘library’. Every structure in this file is considered to be a library structure. This file is normally read from **DOSFOP-path/defaults/Library.config**, but you may copy it to the local DOSFOP directory and modify it.

Backup files are given the extension ‘.old’.

4.4 Other Files

There are some other files located in the DOSFOP directory.

`'dosfop.switches'`

Constructed every time the translation to Texinfo is started. This file contains variable settings which change according to the global option settings. See Section 3.4 [Switches], page 16, for an explanation of the switches.

`'diagnostics'`

This file contains all warnings encountered during the translation to intermediate output.

`'lastChange'`

This file is `'touch'`ed every time the global configuration changes.

`'options.data'`

This file contains X defaults for the graphical appearance of the windows of the graphical user interface.

5 The Programs DOSFOP Uses

Most programs named in this section are located in `<DOSFOP path>/bin`, if not otherwise stated.

5.1 DOSFOP Programs

These are the core programs of DOSFOP, all written in OPAL.

`checkConsistency`

This program verifies the existence of all source-code files and associated interopal files.

It is called with one argument: the top-level directory.

`dosfopTranslator`

This program reads the project data base and the global configuration found in the `DOSFOP` subdirectory and produces the Texinfo intermediate output.

It is called with the top-level directory as the current directory and without arguments.

`getsetting`

It extracts the local settings for a structure or a subsystem.

It is called with two arguments: the name of the subsystem or the structure and the top-level directory as the second argument.

5.2 Graphical User Interface

The TCL/TK programs are located in the directory `<DOSFOP path>/tcl`.

The files ending in `‘.tcl’` are read by the main programs; the other programs are started as separate processes.

`browse`

Call the project browser.

`dosfop`

Open the DOSFOP main window.

`globalConfig`

Edit the global options.

`pureDosfop`

Call DOSFOP without using windows.

`structureConfig`

Edit the local options of a structure, whose name must be given as a parameter.

`subsystemConfig`

Edit the local options of a subsystem, whose name must be given as a parameter.

5.3 Programs for Producing DVI Files

`dosfopMacroExpander`

This program is used instead of `makeinfo` for the expansion of macros (see Section 5.6 [External Interpreters], page 23).

`dosfopTexindex`

This program is needed as an wrapper of the `texindex` program. The generated indices need to be processed before and after the call to `texindex` (see the following entry).

`fixtexindexpre`

`fixtexindexpost`

The generated indices have some flaws and bugs which are corrected by these two scripts.

`fixtexindexpre` removes `\penalty 10000` from the index entries and corrects name entris starting with a backslash.

`fixtexindexpost` disables hyphenation of the function arrow and corrects the initial letter of names starting with underscore.

`texi2dvi --verbose <intermediate Texinfo file>`

This program controls the production of a DVI file from a Texinfo file. Note that we need Version 1.11 or newer (the Texinfo 3.9 distribution contains Version 1.10). Three environment variables are set before the call to `texi2dvi`:

`TEXINDEX dosfopTexindex`

– to replace the standard `texindex` program with the DOSFOP version.

`TEXINPUTS .:<DOSFOP path>/tex:<old value of TEXINPUTS>`

– to ensure that files included by the generated Texinfo file are found at the proper places.

`MAKEINFO dosfopMacroExpander`

– to replace `makeinfo` as the macro-processing program.

Before `texi2dvi` is called, the current directory is changed to the DOSFOP directory.

`texindex`

This program is needed to process the indices for DVI output.

5.4 Programs for Producing INFO Hypertext

Formerly, the INFO format was the only supported hypertext output. Nowadays, INFO output is rarely used; you can achieve a similar effect with an ASCII web browser like `lynx`. Support for INFO output may not be available in future versions of DOSFOP.

`dosfopMacroExpander`

This program is used instead of `makeinfo` for the expansion of macros (see Section 5.6 [External Interpreters], page 23).

The program is called with these parameters:

```
dosfopMacroExpander --verbose -I <DOSFOP subdirectory>
                    -I <DOSFOP path>/tex
                    -E <intermediate Texinfo file>.new
                    <intermediate Texinfo file>
```

makeinfo

The `makeinfo` program generates the INFO file. It is known to have difficulties with long Texinfo files.

It is called with the following parameters:

```
makeinfo --verbose --no-validate --no-warn -I DOSFOP
        -I <DOSFOP path>/tex
        <intermediate Texinfo file>
        -o <info output file>
```

The program is called with the top-level directory as the current directory.

5.5 Programs for Producing HTML Hypertext

dosfopMacroExpander

This program is used instead of `makeinfo` for the expansion of macros. (see Section 5.6 [External Interpreters], page 23).

The program is called with these parameters:

```
dosfopMacroExpander --verbose -I <DOSFOP subdirectory>
                    -I <DOSFOP path>/tex
                    -E <intermediate Texinfo file>.new
                    <intermediate Texinfo file>
```

texi2html

This program was originally written by Lionel Cons at CERN and has been adapted to the special needs of generating documentation (copying the links from the INFO file, better presentation of indices and some other more minor adjustments).

The program is called with these parameters:

```
texi2html -toc_name <Table of Contents string>
          -I ../DOSFOP -I <DOSFOP path>/tex
          -split_node -menu -verbose
          <intermediate Texinfo file>
```

The *<Table of Contents string>* is a language-dependent string used for links to the top node. The program is called with the top-level directory as the current directory.

5.6 External Interpreters

While the translation to the intermediate Texinfo file does not rely on external interpreters, some of the programs which form the environment do. These interpreters are listed here. We also include the description of the `dosfopMacroExpander` here, because there is no better place for this.

sh

The **sh** interpreter is used for **texi2dvi** and **dosfopTexindex**.

perl

Perl is used for **dosfopMacroExpander**, **fixtexindexpre**, **fixtexindexpost** and **texi2html**. These programs run both under Perl 4 and Perl 5. The location of perl is read from the environment variable **PERL** which is set by **dosfop** and **pureDosfop**.

wish

TCL/Tk Version 4.1 (or newer) is used for the graphical user interface and the integration of DOSFOP and OASYS. The location of wish is read from the environment variable **WISH** which is set by **dosfop** and **pureDosfop**.

dosfopMacroExpander

The macros, which were introduced into the latest Texindex edition (version 3.9, October 1996) are expanded by a call to the **makeinfo** program. Unfortunately, the macro expansion in **makeinfo** is not bug free, so I decided to re-implement a separate program for macro expansion. This implementation has some restrictions:

- Macro names must start with an uppercase letter.
- Parameters may not contain other macro calls or commands (except for special macro invocation, where the argument is the rest of the entire line).
- No macro calls may occur in lines in which the Texinfo commands **@TeX{}**, **@AA** or **@AE** are used.

Appendix A Tips and Tricks

This appendix contains some tips and tricks for achieving certain effects through sophisticated usage of Texinfo and the possibilities for (re)defining macros. In most cases you will need to copy the file(s) ‘dosfop.macros’ and/or ‘dosfopPrelude.texi’ from ‘<DOSFOP path>/tex’ and modify them.

Trouble-shooting:

Commands are not processed and appear verbatim in the generated documentation.

Some commands are allowed only at the beginning of a line, *not even spaces or tabs* may occur before them. If this happens to commands introduced by a macro, start the macro with an empty line.

A.1 Coloured Source Code

To make your source code appear in a different colour in HTML hypertext, use these definitions:

```
@macro BEGINOPAL
@ifhtml
<font color="8B000">
@end ifhtml
@example
@end macro

@macro ENDOPAL
@end example
@ifhtml
</font>
@end ifhtml
@end macro
```

Similarly, bychanging the macro pairs @BEGINDOC/@ENDDOC and/or @BEGINOVW/@ENDOVW, you can give a different colour to documentation and/or overviews.

A.2 Highlight Warnings

If you want to make warnings stand out, try these definitions:

```
@macro BEGINDOCwarning
@ifhtml
<font color="FF0000"><blink>
@end ifhtml
@cartouche
@end macro

@macro ENDDOCwarning
@end cartouche
@ifhtml
</blink></font>
```

```
@end ifhtml
@end macro
```

This will result in a box with rounded corners around the documentation in the printed manual (`@cartouche`) and make the warning blink and appear in red in the HTML hypertext.

The documentation must then be *tagged* with the tag ‘`warning`’:

```
-- %{warning} This function is completely untested!
```

resulting in:

```
This function is completely untested!
```

A.3 No Indentation of Source Code

By default, source code is put into an `@example` environment, which indents it. To remove the indentation, redefine `@BEGINOPAL` and `@ENDOPAL`:

```
@macro BEGINOPAL
@t{
@format
@end macro

@macro ENDOPAL

@end format
}
@end macro
```

A.4 Indentation of Documentation

Normally, source code is indented and documentation is not. To reverse the default, redefine `@BEGINOPAL`, `@ENDOPAL`, `@BEGINDOC` and `@ENDDOC`:

```
@macro BEGINOPAL
@t{
@format
@end macro

@macro ENDOPAL
@end format
}
@end macro

@macro BEGINDOC
@quotation
@end macro
```

```
@macro ENDDOC
@end quotation
@end macro
```

A.5 Obey Line Breaks in Documentation

When you process ASCII-style comments (perhaps when using DOSFOP for non-OPAL sources, see Section 2.6 [DOSFOP and Non-OPAL Programs], page 13), you might want to retain the line breaks. This is achieved by these definitions:

```
@macro BEGINDOC
@format
@end macro

@macro ENDDOC
@end format
@end macro
```

A.6 Add New Indices

If you want to add an ‘xy’ index, you must edit two files:

‘dosfopPrelude.texi’

Add the line

```
@defcodeindex xy
```

Use `@defindex` if entries are to appear in Roman and not fixed-width font.

‘dosfop.macros’

Change the `@FINALIZEDOC` macro to insert your index at the end:

```
@macro FINALIZEDOC
@node XY Index, To Do, Tips And Tricks, Top
@chapter XY Index
@printindex xy

@contents
@end macro
```

You may now use the command `@xyindex` to include entries:

```
-- %@xyindex An XY index entry
```

A.7 Find Unfinished Source Code

DOSFOP can help you keep track of all the places in your code you think need a second look.

We just use the technique presented in Section A.6 [Add New Indices], page 27, to add a new ‘To Do’ index:

‘dosfopPrelude.texi’

Add the line

```

@defcodeindex td

‘dosfop.macros’
    Change the @FINALIZEDOC macro to insert your index at the end:
        @macro FINALIZEDOC
        @node To Do, Texinfo for Experienced TeX Users, XY Index, Top
        @chapter To Do
        @printindex td

        @contents
        @end macro

```

The code then looks like this:

```

/* %This function sorts its argument.
@tdindex FUN sort implemented inefficiently
*/
FUN sort: seq[mytype] -> seq[mytype]

```

If you want to highlight these parts of the documentation, too, see Section A.2 [Highlight Warnings], page 25.

A.8 Hide Source Code

You can exclude source code from the generated documentation with the help of the `@ignore` environment:

```

-- %@ignore
...
... lots of hidden functions
...
-- %@end ignore

```

Do not forget the `@end ignore` line! Otherwise no more documentation will be generated after the initial `@ignore`.

A.9 Javadoc

DOSFOP cannot process the special tags recognized by `javadoc`. But if you are willing to sacrifice DVI and INFO output, DOSFOP can also process files documented in JAVADOC style.

The following definitions allow HTML documentation:

```

@macro BEGINDOC
@ifhtml
@end macro

@macro ENDDOC
@end ifhtml
@end macro

```

To allow inclusion of JAVA files, create a file `‘dosfop.filetypes’` in the DOSFOP directory, which consists of this line:

```
.java: : :/**:*/: :
```

This will tell DOSFOP how to process ‘.java’ files: nested comments starting with `/**` and ending with `*/` are documentation, everything else is source code.

Appendix B Texinfo for Experienced TeX Users

Experienced T_EX users should not have much difficulty writing Texinfo. The look and feel of Texinfo is similar to a special LaTeX style.

The following differences are the most important:

Escaping by @

The escape character is the at sign ('@'). The backslash character ('\') need not be escaped. The only characters that must be escaped are the at sign (write @@) and the curly brackets (write @{ and @} resp.).

Environments

An environment `env` is begun with `@env` and terminated with `@end env`. Both commands must appear on a line of their own and may not be preceded by spaces or tabs.

Commands that occupy an entire line

Most commands are followed by an argument list in braces, as in T_EX. But there is a set of commands that occupy an entire line. These commands must be written at the beginning of a line (no spaces or tabs allowed). They take the rest of the line as their single argument.

Itemize lists

There are two traps for T_EX users with itemize lists:

1. The command to begin an environment has an additional argument for the command generating the mark (most often `@bullet` or `@minus`).
2. The `@item` command must appear on a line of its own and may not be preceded by spaces or tabs.

An itemize list might look like this:

```
@itemize @bullet
@item
Text for first item.
@item
Text for second item.
@item
...
@end itemize
```

Appendix C Useful Texinfo Commands

Texinfo is a language which combines commands for setting up hyperlinks, for chapter structuring and for text formatting. Fortunately, the user of DOSFOP is only concerned with the text-formatting parts.

For a very quick introduction, you need to learn only the following commands:

- @@**
- @{**
- @}** The characters ‘@’, ‘{’ and ‘}’ must be written with an initial at sign (‘@’). All other characters can be used as they are.
- @code{*text*}**
 Use **@code** to mark references to identifiers from the source code.
- @emph{*text*}**
 Use **@emph** to highlight (emphasize) text.
- @ref{@Overview{*structure name*}}**
 Use **@ref{@Overview{*structure name*}}** to create a hyperlink to structure *structure name*.

A full description is given in the Texinfo manual. The most important Texinfo commands are briefly described in this appendix.

C.1 Environments

The following list contains the commands needed to begin an environment. Environments are ended by the command **@end environment**.

These commands must be on a line of their own and may not be preceded by spaces or tabs.

- @enumerate**
 Begin a numbered list. Use a separate **@item** line at the beginning of each entry. You may give the beginning command a number or a letter, which is then used for the first entry.
- @itemize *mark-gen-char-or-command***
 Produce a sequence of indented paragraphs with a mark inside the left margin at the beginning of each paragraph. Use a separate **@item** line at the beginning of each entry. The beginning command takes an argument which is used to generate the initial mark. Most often, **@bullet** or **@minus** are used.
- @table *formatting-command***
 Begin a two-column table. Use **@item *first-column*** for each entry. First-column entries are printed in the font given in *formatting-command*.
 If you want to group several entries, use **@itemx** instead of **@item** for the second and following entries. Add the explanation after the last **@itemx** line.

C.2 Font Modification

`@asis{text}`

No font modification (useful for two-column tables, see Section C.1 [Environments], page 31).

`@b{text}`

Use bold font.

`@code{text}`

Highlight text that is an expression, a syntactically complete token of a program, or a program name.

`@emph{text}`

Highlight (emphasize) text.

`@i{text}`

Print text in italic font.

`@r{text}`

Print text in roman font.

`@sc{text}`

Set text in a printed output in the small caps font.

`@t{text}`

Print text in a fixed-width, typewriter-like font.

C.3 Formatting Commands

Paragraphs are separated by an empty line. In the generated documentation, every comment that contains documentation is given a separate paragraph. (One exception: In non-OPAL source code which does not have nested comments, subsequent line comments are merged. See Section 2.6 [DOSFOP and Non-OPAL Programs], page 13.)

`@*`

Force a line break.

`@-`

Mark possible hyphenation for printed output.

`@noindent`

Do not indent the following paragraph.

`@page`

Start a new page in a printed manual.

C.4 Cross References

Text may contain cross references to other ‘nodes’. DOSFOP provides the commands `@Overview{structure name}`, `@Sign{structure name}`, `@Impl{structure name}`, `@Extp{structure name}`, `@Intp{structure name}` to reference the overview, signature part, implementation part, external property part, internal property part respectively. Use `@LibSign{structure name}` `@LibExtp{structure name}` to refer to library structures.

`@ref{node name}`

Generate a reference to the named node.

`@xref{node name}`

Produce an initial ‘See’ followed by a reference to the named node.

`@pxref{node name}`

Use it only inside parentheses and do not type a comma or period after the command’s closing bracket. Produces an initial ‘see’ followed by a reference to the named node.

C.5 Miscellaneous

`@c comment`

`@comment comment`

Begin a comment in Texinfo. The rest of the line does not appear in the generated documentation.

`@cindex entry`

Add entry to the index of concepts. You may switch on the concept index in the global options (see Section 2.2.1.2 [Global Options], page 7).

`@email{mail address}`

Indicate an electronic mail address. In HTML this produces a hyperlink.

`@footnote{text of footnote}`

Enter a footnote.

`@url{URL}`

Highlight text that is a uniform resource locator for the World Wide Web. In HTML this produces a hyperlink.

Appendix D Changes

DOSFOP 1.92g has the following differences to version 1.92f:

- In the ‘doc’ directory, an additional top-level HTML file ‘`index.html`’ is created.
- Bug fix: Earlier versions were unable to handle the region variant of InterOpal positions, which are generated by ocs 2.3c and newer.

DOSFOP 1.92f has the following differences to version 1.92d:

- Output of superfluous empty lines has been reduced.
- Indices in HTML output use `<TABLE>` and colours. Functionality in indices now printed in italics.
- If a subsystem (or toplevel) contains a file ‘`Survey.texi`’, this file is included instead of the survey in the configuration.
- Defaults have been slightly changed. Coloured source code (as given in Section A.1 [Coloured Source Code], page 25) is no used by default.
- OCS has now support for building documentation.

DOSFOP 1.92d has the following differences to version 1.92c:

- Definition of `DOSFOP` and extension of `PATH` is no longer necessary.
- The initial HTML file is called ‘`doc/out.html`’.
- The previewers for DVI, INFO and HTML files can be configured by setting the variables `XDVI`, `INFO` and `WWWBROWSER` respectively.
- Locations and some names of TCL/Tk scripts have changed.
- Location of `perl` and `wish` are not hard-wired into the code, but read from the variables `PERL` and `WISH` respectively.

DOSFOP 1.92c has the following differences with respect to version 1.90h:

- For every structure `S`, a Texinfo file `OCS/S.doc.texi` is created.
- Incremental generation of documentation is supported, See Section 2.7 [Incrementally Generating Documentation], page 14.
- Some bugfixes, especially index generation.
- The generated HTML files uses frames. Default output has changed to ‘`doc/out.html`’.

Index

*

* (as special top structure)..... 7

.

.oasysrc..... 12

@

@*..... 32

@-..... 32

@AA..... 24

@AE..... 24

@asis{text}..... 32

@b{text}..... 32

@BEGINDOC (macro)..... 18, 26, 27, 28

@BEGINDOCoptimize (macro)..... 4

@BEGINDOCtag (macro)..... 4, 7

@BEGINDOCwarning (macro)..... 25

@BEGINOPAL (macro)..... 18, 25, 26

@BEGINOVW..... 18

@bullet..... 30, 31

@c comment..... 33

@cartouche..... 25

@cindex..... 8

@cindex entry..... 33

@code{text}..... 32

@comment comment..... 33

@contents..... 18

@defcodeindex..... 27

@defindex..... 27

@dfStarLine..... 15

@email{mail address}..... 33

@emph{text}..... 32

@ENDDOC (macro)..... 18, 26, 27, 28

@ENDDOCoptimize (macro)..... 4

@ENDDOCtag (macro)..... 4, 7

@ENDDOCwarning (macro)..... 25

@ENDOPAL (macro)..... 18, 25, 26

@ENDOVW..... 18

@enumerate..... 31

@example..... 18

@Extp (macro)..... 15

@FINALIZEDOC (macro)..... 18, 27

@finalout..... 15

@findex..... 14

@footnote{text of footnote}..... 33

@i{text}..... 32

@ifclear html..... 17

@ifhtml..... 17

@ifinfo..... 17

@ifset html..... 17

@iftex..... 17

@ignore..... 28

@Impl (macro)..... 15

@Intp (macro)..... 15

@item..... 30, 31

@itemize..... 30

@itemize *mark-gen-char-or-command*..... 31

@itemx..... 31

@LibExtp (macro)..... 15

@LibSign (macro)..... 15

@minus..... 30, 31

@noindent..... 32

@page..... 32

@pindex..... 14

@pxref..... 15

@pxref{node name}..... 33

@r{text}..... 32

@ref..... 15

@ref{node name}..... 33

@sc{text}..... 32

@setchapternewpage..... 15

@Sign (macro)..... 15

@t{text}..... 32

@table *formatting-command*..... 31

@tdindex..... 27

@tex..... 17

@TeX..... 24

@url{URL}..... 33

@xref..... 15

@xref{node name}..... 33

@xyindex..... 27

A

administrative data..... 6

Application index..... 8

Application index (option)..... 8

Authors'Names (option)..... 7

B

Basic Language (option)..... 9

binary configuration..... 11

browse..... 21

C

change path of subsystem..... 10
 Check Configuration Consistency (button)... 5, 11
 checkConsistency..... 21
 colon..... 16
 colouring text 25
 commands that occupy an entire line..... 30
 Concept Index (option)..... 8
 conditionally defined macros..... 17
 conditionally visible text..... 16
 config..... 11, 19
 Config (button)..... 10
 Config Global (button)..... 10
 config.bin..... 11
 configuration of structure..... 7, 10, 21
 configuration of subsystem..... 7, 10, 21
 configurator 6
 convert to DVI (button) 6
 convert to HTML (button)..... 6
 convert to Info (button) 6
 customizing the documentation 6

D

Date of Creation (option)..... 7
 Definition index 8
 Delete (button)..... 9, 10
 deletion of structures 9
 deletion of subsystems 10
 dependencies between subsystems and/or
 structures..... 9
 dfAuthors..... 16
 dfCodomainIndexFlag..... 16
 dfDate..... 16
 dfDomainIndexFlag..... 16
 dfEnglish..... 16
 dfGerman..... 16
 dfo.tcl..... 12
 dfProjectName..... 16
 dfSingleNode..... 16
 dfSortalphabetically..... 16
 dfSortbottom_up..... 16
 dfSortoff..... 16
 dfSorttop_down..... 16
 dfSortuser_defined..... 16
 dfStarLine..... 18
 diagnostics 20
 Display All Options (button) 7
 doc..... 12
 doc directory..... 3, 7
 doc-dvi 12

doc-help..... 12
 doc-html..... 12
 doc-info..... 12
 documentation headers 4
 documentation sectioning..... 4
 dosfop..... 5, 21
 DOSFOP (environment variable)..... 3, 12
 DOSFOP execution window..... 5
 DOSFOP main window..... 5, 21
 dosfop.filetypes 13, 28
 dosfop.macros 18, 25, 27, 28
 dosfop.switches..... 20
 DOSFOP/foo.deps..... 14
 DOSFOP/foo.index..... 14
 dosfopLatin1.sty 15, 19
 dosfopMacroExpander..... 22, 23, 24
 dosfopPrelude.tex..... 18
 dosfopPrelude.texi..... 18, 25, 27
 dosfopTexindex..... 22
 dosfopTranslator..... 21
 Drop Empty Lines (option)..... 9
 dvi..... 12
 DVI files, constructing..... 6, 11, 12
 DVI previewer (button)..... 6

E

edit list of tags 7
 empty lines 9
 environment variables..... 3
 environments 30, 31
 escaping by @ 30
 exclude source code..... 28
 execution window (of DOSFOP)..... 5
 external property part 8

F

filebrowser..... 9, 10
 fixtexindexpost..... 22
 fixtexindexpre..... 22
 foo.deps..... 14
 foo.index..... 14
 free type declaration..... 8
 Function Index 8
 Function Index by Codomain 16
 Function Index By Codomain..... 8
 Function Index by Domain..... 16
 Function Index By Domain 8
 Functionality Index (option) 8

G

generated documentation 3
getsetting..... 21
 global configuration..... 6
 Global Configuration (button) 5
 global options..... 6, 7, 19, 21
globalConfig..... 21
 GNU utilities..... 3
 graphic user interface, without..... 10
 graphic visualization of dependencies..... 9
 graphical user interface 21

H

headers, additional h. in documentation..... 4
 hidden documentation 5
 hide source code..... 28
 Hierarchy Visualization (option) 9
html 16
 HTML files, constructing 6, 11, 12
 HTML files, constructing quickly 11
 HTML tags..... 17
 HTML viewer (button) 6
 hyphenation 32

I

ignore source code 28
 ignored documentation..... 5
 import references for non OPAL sources 14
 Import Referencing (option) 9
 import relation 7
 Include only Interfaces (option) 8
 indentation 26
 ‘**index.html**’ 3
 indices for non OPAL sources..... 14
 indices used by DOSFOP 15
 individual configuration of structure..... 7
 individual configuration of subsystem 7
INFO (environment variable) 6
 INFO files, constructing 6, 11, 12
 Info hypertext reader (button)..... 6
 Insert (button)..... 9, 10
 insertion of structures 9, 10
 insertion of subsystems 10
 Intermediate code output file (option)..... 7
 intermediate Texinfo output..... 6, 7, 11, 12
 internal property part..... 8
 introduction 7
 introduction to project..... 3
 introduction to subsystem 3
 ISO-8859-1 character set..... 15

itemize lists 30

J

javadoc 28

L

lastChange..... 20
 latin1 character set 15
 law 5
 laws 8
 level documentation (= tagged doc.) 4, 7, 26
 level handling..... 7
 Level Up (button) 10
 levels of sectioning 4
 library..... 8
 Library Inclusion (option) 8
 library structures 7, 19
Library.config..... 19
 line breaks 27
 line numbers 14
 local options..... 19
lynx 22

M

macro calls 24
 macro definitions 18
 macro names 24
 macro packages..... 15
 main window (of DOSFOP) 5, 21
makeinfo 22, 23
MAKEINFO (environment variable)..... 22
 memory overflow 12
 Move (button)..... 10
 moving of subsystems..... 10

N

Name of the Project (option) 7
 new indices 27
 new page 32
 new pages 9
 non OPAL source code..... 13, 32

O

OASYS 12
 OCS 11
OCSDIR (environment variable) 12
only..... 11, 12
options.data 20
 ordinary documentation 3

P

paragraphs, formatting of 32
 path of subsystem, changing 10
perl 24
PERL (environment variable) 24
 plain **T_EX** 17
 predefined indices 15
 project browser 9, 21
 Project Browser (button) 5
 project data base 9, 19
 project, introduction to 3
 project, survey of 3
PROJECTROOT (environment variable) 11
 properties 8
 Property Inclusion (option) 8
 Property Index 8
 property parts 8
 property parts and property references 5
 property references 5
psfig.sty 19
psfig.tex 15, 19
pst-node.tex 15, 19
pstricks.con 19
pstricks.sty 19
pstricks.tex 15, 19
pureDosfop 10, 19, 21

R

Rename (button) 10
 renaming of structures 10
 renaming of subsystems 10
 ribbon 4
 Root-location of the Project (option) 7

S

sh 24
SIGNATURE keyword 3
 single node 9
 Sort Index 8
 Sort Structures (option) 9
 Start DOSFOP (button) 6
 Start New Pages (option) 9
 Structure in single node (option) 9
 Structure Index 16
 Structure Index (option) 8
structure name.config 19
 structure, survey of 3
structureConfig 21
structureindexflag 16
 Subsystem Inclusion (option) 8

subsystem name.config 19
subsystem name.names 19
 subsystem, introduction to 3
 subsystem, survey of 3
subsystemConfig 21
 survey 7
 Survey (button) 7
 survey of project 3
 survey of structure 3
 survey of subsystem 3
Survey.texi 3, 7

T

tagged documentation 4, 7, 26
TAPSOFT 1
TCL/Tk 21, 24
 technical report 1
T_EX macros 15
texi2dvi 22
texi2html 23
texindex 22
TEXINDEX (environment variable) 22
Texinfo 3
Texinfo environment 15
Texinfo special characters 14
texinfo.tex 18, 19
TEXINPUTS (environment variable) 22
thisdoc 12
thisdocall 12
 title page 7
 To-Do Index 27
 top structure 7
 Top Structure Name (option) 7
 toplevel 7
Toplevel.config 19
 trouble-shooting 25
 types of documentation 3

U

unfinished source code 27
 Used Function Tables (option) 9

V

visualization of dependencies 9

W

wish 24
WISH (environment variable) 24
WWWBROWSER (environment variable) 6

X

X defaults	20
xy index	27
XDVI (environment variable)	6

Table of Contents

1	Introduction	1
1.1	DOSFOP's Architecture	1
1.2	DOSFOP's History	2
2	Using DOSFOP	3
2.1	Writing Documentation	3
2.1.1	Project and Subsystem Surveys	3
2.1.2	Structure Surveys	3
2.1.3	Source-Code Documentation	3
2.2	The Graphical User Interface	5
2.2.1	Global Configuration	6
2.2.1.1	Administrative Data	6
2.2.1.2	Global Options	7
2.2.2	Project Browser	9
2.2.2.1	Structures	9
2.2.2.2	Subsystems	10
2.2.2.3	File Browser	10
2.3	DOSFOP without Windows	10
2.4	DOSFOP and OCS	11
2.5	DOSFOP and OASYS	12
2.6	DOSFOP and Non-OPAL Programs	13
2.7	Incrementally Generating Documentation	14
3	The Texinfo Environment for DOSFOP	15
3.1	TeX macros	15
3.2	Texinfo setup	15
3.3	Predefined Macros	15
3.4	Switches	16
3.5	Conditionally Visible Text	16
3.5.1	Macros and Conditions	17
4	The Files DOSFOP Uses	18
4.1	TeX- and Texinfo-Related Files	18
4.2	TeX Styles	18
4.3	Files in the Project Data Base	19
4.4	Other Files	20

5	The Programs DOSFOP Uses	21
5.1	DOSFOP Programs	21
5.2	Graphical User Interface	21
5.3	Programs for Producing DVI Files	22
5.4	Programs for Producing INFO Hypertext	22
5.5	Programs for Producing HTML Hypertext	23
5.6	External Interpreters	23
Appendix A	Tips and Tricks	25
A.1	Coloured Source Code	25
A.2	Highlight Warnings	25
A.3	No Indentation of Source Code	26
A.4	Indentation of Documentation	26
A.5	Obey Line Breaks in Documentation	27
A.6	Add New Indices	27
A.7	Find Unfinished Source Code	27
A.8	Hide Source Code	28
A.9	Javadoc	28
Appendix B	Texinfo for Experienced TeX Users	
	30
Appendix C	Useful Texinfo Commands	31
C.1	Environments	31
C.2	Font Modification	32
C.3	Formatting Commands	32
C.4	Cross References	33
C.5	Miscellaneous	33
Appendix D	Changes	34
Index	35