



SOFTWARE DOCUMENTATION

Claire Reed

TESSELLA SUPPORT SERVICES PLC

Issue V1.R3.M1

July 1998



SOFTWARE DOCUMENTATION

Introduction

'*Help, I do not understand the manual*' is an all too common complaint, made by software users due to the poor quality of much of the accompanying technical documentation. Manuals are often written with little, or no consideration for the needs of new users: a fact reinforced by the number of '*Dummy's Guide*' books subsequently written to interpret the manufacturer's own documentation. This supplement provides guidelines on writing software documentation, which, if followed, should ensure that users of *your* software will be fully satisfied.

What is Good Documentation?

Good technical documentation is easy to use, comprehensive and aimed at the level of the potential users. Some points to consider are:

- Level** - The technical level of the document should be aimed to match that of the potential users. Phrases like '*Close the Window by double-clicking the mouse*' may be quite clear to you, but not necessarily to a new computer user. Conversely an experienced user will not wish to waste time reading about how to use a mouse, for example.
- Size** - A single hefty manual is more daunting to use and prone to damage than two narrower volumes, e.g. split into User Guide and Reference Manual.
- Manageability** - Manuals that require a hand or weighty object to stay open can be annoying to use, especially when typing/copying from the text.
- Quality of print** - Good quality print in a pleasing font makes a manual easier to read.
- Use of English** - Make sentences clear and concise, but not to the point of being stilted or terse.
- Jargon and acronyms** - Use them as infrequently as possible and where used make sure the meaning is clearly documented.
- Page layout** - Keep it uncluttered, leaving plenty of white space.

- ❑ **Signposting** - Make it as easy as possible to find the solution to an individual question with sensible section headings, contents page and cross-referencing. Remember, the user will rarely be reading the manual from cover to cover.
- ❑ **Consistency** - Ensure that the system documentation is up-to-date with the software. There is no point using a help system if it does not refer to the version of the software being used.

Styles of Documentation

There are two styles of user documentation generally used: ‘instruction’ or ‘tutorial’ style, useful for the new user, and ‘reference’ style, useful for the more experienced user who needs information on specific topics. These two styles are often respectively known as the **User Guide** and the **Reference Manual**. For a large and complicated package, with possibly a wide range of potential users, both documents are essential. For less extensive software packages with a more specific user base, a single document may suffice, so long as it is designed with the beginner in mind. The document styles are described more fully below.

A **User Guide** is an instructional document giving step-by-step procedures and explanations for accomplishing specific tasks. It should be ordered according to the learning path, with simple and necessary operations appearing first and the more complicated, advanced operations appearing later. It is not necessary to cover all the software features in detail in this section, but provide enough information for all but the most experienced users. This section should be designed to ‘teach’ new users how to use the software, so worked examples of useful operations should be included.

A **Reference Manual** contains the basic operations, facts and key definitions, ordered for easy reference (e.g. alphabetically). The aim of this section is to allow an experienced user to look up terms and procedures as simply and rapidly as possible. This section should be more rigorous, formal and exhaustive than the User Guide, and a working knowledge of the package may be assumed. Descriptions in this section should contain **all** the options, parameters and qualifiers available in the software, with worked examples where possible

In addition to the complete Reference Manual it can be useful to include a **Quick Reference Card** giving the experienced user memory jogging, at-a-glance information.

Electronic Documentation

Instead of providing paper versions of manuals, it is common practice these days to include on-line interactive help within the software itself. This helps to remove the dependence on paper documentation and generally offers a faster means of answering a particular query. The level of complexity of this service will depend on the user requirements for the software and the available resources and time. At their very best, on-line help facilities can remove the need to examine paper manuals completely, with the entire user manual including illustrations available. There are three commonly used methods of providing on-line help: **Simple**, **Hyperhelp** and **Context Sensitive**. The relative merits of these facilities are described below.

A **Simple** help system is just that: simple. The user is provided with a Help button or menu item on every window or control panel. Selection of this will display a page or two of information relating to the panel or window from which it was selected. At the very least it should describe each control, display and option on the window in terms of function, type of input and range of input. The user should also be able to operate the software whilst viewing the help information, so that any step-by-step instructions may be followed without recourse to pen and paper. This system is very easy to implement and easy to update. Displaying the information is simply a matter of reading help text from a file and displaying it on screen. Updates are achieved by simply editing the relevant text file.

A **Hyperhelp** system is more complicated, but much more powerful. Hyperhelp documents use interactive cross-referencing to allow the user to jump from one section to another following related links. The entire manual can be stored on-line in this format, giving the user complete access to all the information required and hence removing the need to view the paper version. Hyperhelp systems are more complicated to implement than a simple system, mainly in the setting up of the cross-referencing links. A number of packages are available to generate documents of this nature, (e.g. RoboHelp, HDK - Hypertext Development Kit, Netscape Communicator) and these *greatly* simplify creating the cross-referenced document. Displaying the information requires a hyperhelp viewer (such as Microsoft Windows Help). The main advantage of a system like this is that the user has complete freedom to follow links or jump to anywhere within the manual, at the click of a mouse. Of course, more care is needed when updating a document of this nature, to keep all links and cross references up to date, although commercial packages will do this automatically.

A **Context sensitive** help system is a useful addition to the above methods. Context sensitive help will display information relating to the currently selected Window, dialogue box or control, when a help button (e.g. F1) is pressed. The information displayed should include a description of the control, type and range of input (if any) and any special characteristics. Help systems of this type are very useful for quick reference, saving the user from looking up the control in the index of the user manual.

Web Pages

A new, increasingly popular, method is to convert the manual to html (Hypertext markup language), for example using Word, and post it onto a Web site for users to access. If maintained properly a system of this nature can provide many benefits. With the documentation posted onto public Web pages, users anywhere in the world can access it, using their own WWW browser. Since the document only exists in one location any changes or updates can be made quickly and are instantly available to all users of the system, wherever they are. If you have an internal Web server set up with say, every member of staff having their own home page, then users can easily publish html versions of documents via links from their home pages. In addition, each major project could have its own home page, with links from there to all the relevant documents. This system means that anyone requiring a copy of any document can have access to it immediately, without having to track down someone who can copy a paper version. An e-mail address can also be provided for users to send their specific queries to, similar to a help desk. To provide an optimum service on the Web however, there are a few points to remember:

- Size** - transferring large amounts of data across the internet can become intolerably slow at peak times of day. For this reason it is much more user friendly to split the document (which could be 100s of pages) into manageable sections, with a separate link to each. However, if users are accessing the manual frequently, it is also helpful to have the entire manual available as a file to download.

- Updates** - it is important to remember to update the html version of the document at the same time as the standard version. It is very easy to forget to update the links and thus allow it to lag behind the most current version.

Document Structure

The basic structure of a document should be designed first. Following the IEEE Standard for Software User Documentation (1987), a software user document should have the following sections:

- Table of contents
- Introduction
- Overview section
- Instruction section (User Guide)
- Reference section (Reference Manual)
- Error messages and remedial actions (Appendix A)
- Glossary of terms (Appendix B)
- Index (Appendix C)

These sections are described in detail below.

The **Table of Contents** must show at a glance what the document is about. It should include **only** the main headings and the major subsections: too much detail here makes a document look over-complex.

The **Introduction** should gain the confidence of the reader, providing the following information:

- Intended readership** - describing who should read the user manual.
- Applicability statement** - stating to which software release this version of the manual applies.
- Purpose** - describing the purpose of the software and this document.
- How to use the document** - describing how the document is intended to be read, for example indicating sections that may be omitted by the more experienced user.
- Related documents** - describing any other relevant design or user manuals relating to this manual.

- ❑ **Conventions** - describing any command syntax or style conventions used, such as typeface for different features.
- ❑ **Problem reporting instructions** - explaining the system for reporting problems with the software.

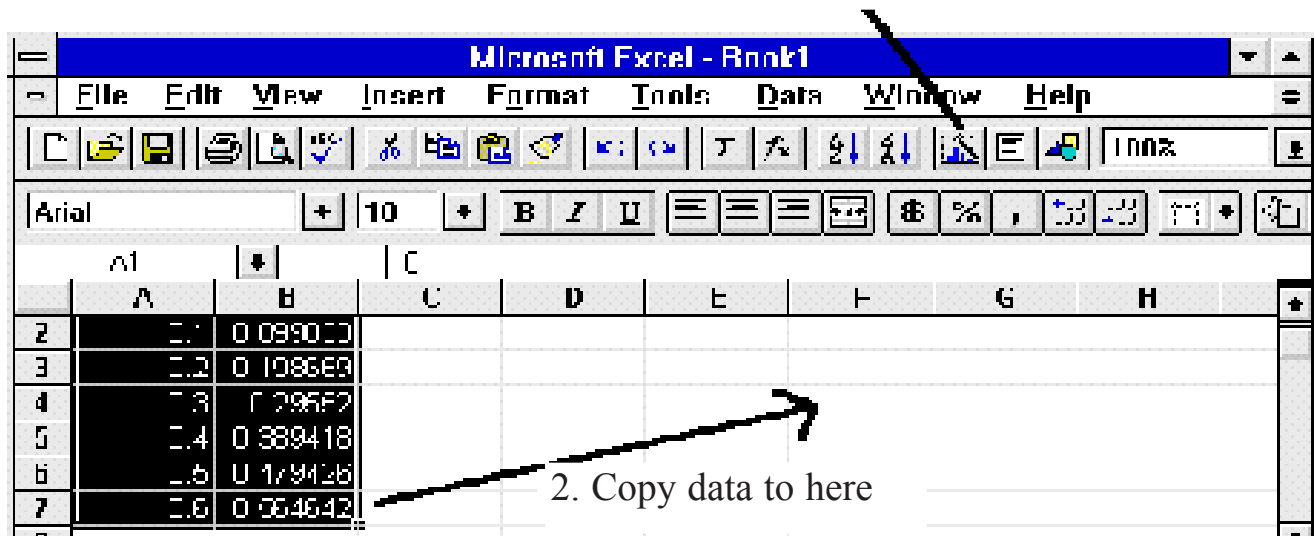
The **Overview** section should give a brief description of what the software is intended to do, what the user can do with the software and anything the user needs to supply to the software, (e.g. specific hardware), to make it work. This section is designed to introduce the software to the new user before getting into the detail of how to use it.

The **Instruction** section should be divided into manageable sections. The main divisions will depend on the type of document, but should always follow a clear logical order, with headings chosen carefully to identify the major topics. For example a simple document could be divided into chapters which relate to specific tasks, whereas a more complex document could be broken down into sections relating to simple, intermediate and advanced tasks. For each task you should provide: a functional description of what the task will achieve, do's and don'ts, initialization, input operations and expected results and probable errors with remedial actions.

In addition, it is also invaluable to add screen shots of expected screen or menu appearance. A well constructed diagram or annotated screen shot can make instantly clear what would have taken a paragraph of text. The user will be interacting graphically with the software, so why not add graphical instructions? (see figure 1) Screen shots pasted into the text are an excellent method for displaying the steps a user follows. New users may be unsure or nervous of this unfamiliar software and need as much reassurance as possible that the correct procedures are being followed.

Remember, this section should be written from the new user's point of view. Never underestimate a novice's ability to misinterpret any instruction in the manual to the detriment of the software. Constantly examine the text and think '*How could this be misinterpreted?*'. In a set of instructions, make sure that no steps are missed out - something you may consider trivial might be the sticking point for a novice. Do not just describe what the user has to do, but also what the expected result is, e.g. what the user will see on the screen, what will happen next.

3. Press this button



1. Select data

2. Copy data to here

Figure 1: An example of a screenshot that could be included in a User Guide

The **Reference** section should be from the expert's point of view. It should contain formal descriptions of each basic operation ordered logically (e.g. alphabetically) with ease of access in mind. It should also be searchable on key words to allow the user to easily reach the topic they require. For every basic operation you should describe: what the operation does, do's and don'ts, a formal description including parameters, options and syntax, worked examples, possible errors and likely causes and cross references to other operations. It is important to state any limitations in any of the functionalities e.g. a timing counter resets after 99 hours. Do not hide this from the user thinking *'No-one will ever use it for this long'*, because they will. Remember advanced users may be pushing the system to its limits and will want to know every operation inside out.

The **Error Message** section should be dealt with away from the main body of the text (e.g. as Appendix A): a user with a problem wants to get to the solution without leafing through pages of text. If an error number is displayed as part of the error message, then ordering in terms of error number is a logical way of structuring them. Otherwise use another logical alternative (e.g. alphabetical). Ordering in terms of error number when no number is displayed to the user is useless. Give a meaningful and helpful description of the error message including what it means, the severity of the error, the possible reasons for the error occurring and most importantly possible remedial actions to resolve the problem.

The **Glossary** is required for technical terms, abbreviations and acronyms which are unavoidable or for terms which have a different meaning depending on the context. Remember to keep this section to a minimum - it should not be used as an apology for using jargon in the text. Being able to search this section is also useful.

The **Index** is one of the main signposting devices in the document. A long, complex document can be rendered useless for certain specific enquiries without an index. For short documents (e.g. less than 40 pages) an index may be unnecessary if the contents page is sufficiently explanatory, and subparagraphs logically ordered. It is useful to build up the index as the document is written, reducing the risk of leaving keywords out. Where there may be ambiguity in what the reader may look up, then include both entries; for example “parameter settings” and “setting parameters”. Remember also that an index with too many references can be as useless as not having one at all, e.g. including every instance of the word “mouse” may well produce tens of references. Obviously this section must also be searchable.

Presentation

A well-presented document is always easier to read and study than an ill-designed one. The design of page layout, the use of lists, tables and illustrations and the choice of typeface, all contribute to the general presentation.

Good page layout involves the use of white space to break up text visually as well as logically according to sections and subsections, leading to a far less daunting appearance. If possible each major section should start on a new page, so the reader can see immediately what to expect. If a subsection extends to more than one page, the use of a list or table could serve to provide a visual break. Illustrations, tables and screen shots should be close to the text to which they refer, without breaking up the flow of the text. Running headers and footers are useful means of identifying the chapter and section on each page.

Lists can be used to draw together short, related items of information. A numbered list is needed only when there is an order of priority between the items, for example to indicate the order of operation of steps or the order of importance of the items. An unnumbered list gives items equal importance and no implied time sequence, but a logical ordering should be used if possible.

Tables should not be too complicated or contain a lot of text because then they

become difficult to interpret. Even in a boxed table, the correct alignment of text and figures is important.

Illustrations of all sorts are important to reinforce visually the messages in the text: images can often make a point more powerfully than words. Conversely, trivial illustrations which add nothing to the text can only serve to distract the reader. Detail on illustrations must be large enough to be clear when the document is reproduced, and labels must not intrude.

Sample programs which can be tried out will enable the reader to feel a sense of achievement, but sections of program in isolation can be unhelpful and confusing. Ensure that the sample program included however does indeed do what the manual said it should.

Screen shots are useful to help the reader check the exact layout and appearance of information on the screen, and they can reinforce examples. However, it is important to distinguish between what the reader will actually see on the screen and what should be typed in. When including screen shots (perhaps saved as bitmaps) in on-line help files, bear in mind that these can greatly increase the size of the files. Another point to remember is to limit the colours in an on-line help system, some users may be viewing the help in only 16 colours.

Standardization - Clearly defined conventions must be used throughout the document - and in any related documents - to distinguish between, say, explanatory text and features of the software. These conventions should be explained in the Introduction. They could include the use of different typefaces, upper and lower case, underlining, and various sorts of brackets and icons. There should be clear distinctions between:

- Explanatory text
- Example programs
- Text appearing on a screen
- User response at a keyboard
- Key names e.g. 'Return', 'Escape'
- Generic text entry
- Specific text entry, i.e. typing in exactly what is shown

In addition there must be some way of indicating repeated elements and choices.

Language

Keep sentences short and simple, without being terse or stilted. Do not use ‘clever’ words or unnecessarily complex explanations. Saying “Press this button to change the screen colours” is much clearer than “The visual display unit’s chromatic qualities may be altered via the depression of this interactive control”.

A training document should address the reader directly, so making it clear what is required. The statement “The operator must enter a password to use the system”, or “A password must be entered to gain access to the system” could be misinterpreted, whereas the instruction “Enter your password to use the system” is quite clear. Research shows that explanations rarely help people use software more effectively, so the documentation should only explain what is really essential. If an explanation is necessary, it should be short and to the point.

Production Methods

When deciding how to reproduce a document there are two major aspects to consider:

- Future updates and corrections
- Manageability for the reader

After the document has been issued, updates and corrections are inevitable: equipment models become obsolete, software applications expand and errors are corrected. This consideration will affect the choice of text reproduction and document binding.

The original text must be prepared on a system which allows for easy alteration and issue control. If amendments are to be sent out to all users, then the binding must allow for these to be inserted quickly and easily. Alternatively, correction instructions will have to be issued.

The binding method also affects the general ease of use for the reader: there is nothing more infuriating than a manual which will not stay open at a chosen page. Most manuals for widely distributed products are of A5 size, issued in a hard-backed ring file. For specialized documents however, the cost of the binding must be justified for a much smaller distribution. The precise method of binding will depend to some extent on the thickness of the document. There are a number of systems on the market for plastic comb binding, which is generally

preferred to a glued or stapled binding because it stays open readily, and offers an easier method of updating. The binding should always be strong enough to take the weight of the document inside it and a reasonable amount of abuse. Manuals that fall apart rapidly degenerate into uselessness.

Depending on the size and complexity of the document, it can be helpful to identify the major sections by clearly recognizable divisions such as coloured title pages or tabs. This gives the reader instant access to the required section such as “Getting Started” or “Index”.

For generating the actual document there are a wide range of options. Most popular are PC based word processing packages such as Word or Wordperfect. These offer a wide range of facilities for formatting documents and are excellent for previewing work. The more complex facilities such as automatic cross references and complex page layouts vary and each package needs to be investigated on its merits.

An older method of production is the use of a text mark-up language such as LaTeX or SGML (Standard Generalized Mark-up Language). These are often more complex to use but once learned can be more powerful. Their main advantage is that they are available on a wide range of platforms and in some cases they are free. Files are stored in ASCII format and so are highly portable.

Hyperhelp and Web Page production can be archived using word processing packages with additional tools. This is traditionally messy and time consuming. More useful is the use of specialist tools but these carry an additional cost. The HTML Web Page language is a mark-up language so can be edited using an ASCII editor but this should only be attempted by those very familiar with the format.

Configuration control of software documentation can be achieved using the standard version control packages. The traditional benefits of using ASCII file formats such as the mark-up languages LaTeX or SGML with a standard package such as SCCS (Source Code Control System) to maintain document versions have diminished as new configuration control software allows the maintenance of the non-ASCII files produced by word processing packages e.g. Starteam.

Authors

The programmer who has spent many hours developing the software package is not necessarily the best person to write documentation for prospective users.

Over familiarity with the system can lead to assumptions about prior knowledge being made.

The ideal author will have a body of experience in writing and using software which can be brought to bear on the documentation of a new package. The essential qualities in an author are the ability to write clear and concise English, and an understanding of the problems likely to be encountered by users of the software. In addition, access to a proof reader who is not computer-literate can be a help in preparing a text in plain English, explicit enough for a novice to use.

There are also many professional companies that specialize in producing user documentation for software houses and work on a contractual basis. A web search with the subject as 'writing & documentation' provides links to some of these companies.

The best way of testing the user manual is of course through use. Ideally this will be by someone who is unfamiliar with the software in question. If they cannot use the software satisfactorily without asking you for help, then your potential users will certainly struggle.

Tessella Support Services plc
Creating Software for Science and Engineering

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software
Data Capture
Simulation Software
Advanced Graphics
Systems Support
Database Applications

Other Technical Supplements available include:

- | | |
|---------------------------------------------------------|--------------------------------------------------------|
| <input type="checkbox"/> Archiving of Electronic Info | <input type="checkbox"/> Object Oriented Programming |
| <input type="checkbox"/> Active Server Pages | <input type="checkbox"/> Pocket PC |
| <input type="checkbox"/> Automated GUI Testing | <input type="checkbox"/> Portable GUI Development |
| <input type="checkbox"/> Bayesian Statistics | <input type="checkbox"/> Printer Technology Guide |
| <input type="checkbox"/> Beowulf Clusters | <input type="checkbox"/> Real Time Systems |
| <input type="checkbox"/> C++ | <input type="checkbox"/> Regression Testing |
| <input type="checkbox"/> Client-Server Technology | <input type="checkbox"/> Security and the Internet |
| <input type="checkbox"/> COM | <input type="checkbox"/> Simulation |
| <input type="checkbox"/> Computational Fluid Dynamics | <input type="checkbox"/> Soft Computing |
| <input type="checkbox"/> Computer Image Processing | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Decision Support Systems | <input type="checkbox"/> Software Development Cycle |
| <input type="checkbox"/> Electronic Data Capture | <input type="checkbox"/> Software Documentation |
| <input type="checkbox"/> Electronic Lab Notebooks | <input type="checkbox"/> Software Portability |
| <input type="checkbox"/> Excel | <input type="checkbox"/> Software Re-engineering |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Specification |
| <input type="checkbox"/> Federal Drug Administration | <input type="checkbox"/> SQL |
| <input type="checkbox"/> FORTRAN 90 | <input type="checkbox"/> UNIX Inter-Process Comms |
| <input type="checkbox"/> Grid Computing | <input type="checkbox"/> UNIX Systems Performance |
| <input type="checkbox"/> High Throughput Screening | <input type="checkbox"/> UNIX Workstations |
| <input type="checkbox"/> Instrumentation | <input type="checkbox"/> Visual Basic 6 |
| <input type="checkbox"/> Integrated Lab Systems | <input type="checkbox"/> WAP |
| <input type="checkbox"/> J2EE | <input type="checkbox"/> Web Services |
| <input type="checkbox"/> Java | <input type="checkbox"/> Windows 2000 Services |
| <input type="checkbox"/> Lims | <input type="checkbox"/> XML |
| <input type="checkbox"/> Linux | <input type="checkbox"/> X Windows |
| <input type="checkbox"/> Microsoft Net | |



INVESTOR IN PEOPLE

Tessella Support Services plc

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: info@tessella.com Web Address: <http://www.tessella.com>