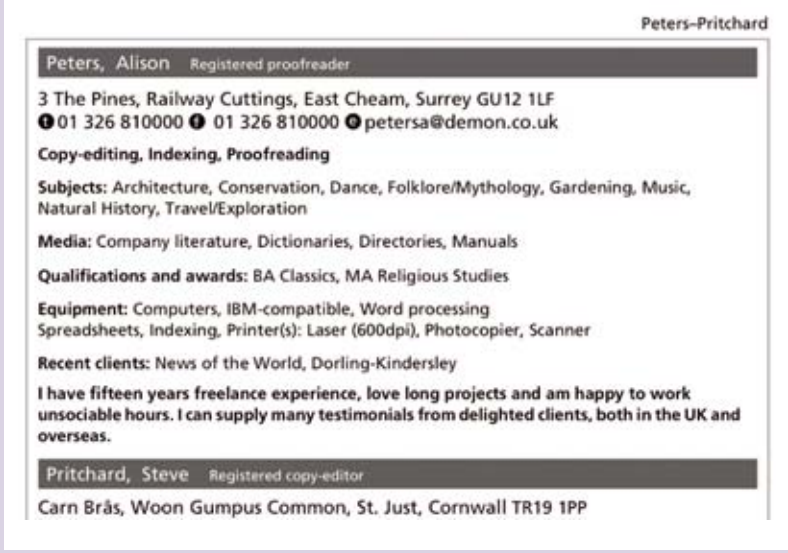
Taking FrameMaker a little further

Steve Rickaby demonstrates FrameMaker's support for long documents with repetitious layouts.



Peters–Pritchard

Peters, Alison Registered proofreader

3 The Pines, Railway Cuttings, East Cheam, Surrey GU12 1LF
01 326 810000 01 326 810000 petersa@demon.co.uk

Copy-editing, Indexing, Proofreading

Subjects: Architecture, Conservation, Dance, Folklore/Mythology, Gardening, Music, Natural History, Travel/Exploration

Media: Company literature, Dictionaries, Directories, Manuals

Qualifications and awards: BA Classics, MA Religious Studies

Equipment: Computers, IBM-compatible, Word processing Spreadsheets, Indexing, Printer(s): Laser (600dpi), Photocopier, Scanner

Recent clients: News of the World, Dorling-Kindersley

I have fifteen years freelance experience, love long projects and am happy to work unsociable hours. I can supply many testimonials from delighted clients, both in the UK and overseas.

Pritchard, Steve Registered copy-editor

Carn Brâs, Woon Gumpus Common, St. Just, Cornwall TR19 1PP

Figure 1. Part of a sample page from the directory of members

Jane Dards' article in the Winter 2005 *Communicator* looked at how FrameMaker (FM) can create tables of contents and indexes. This article goes a little further, to show how FM's indexing is powerful and flexible enough to support quite complex documents. It also looks at some other useful features, such as paragraph tag sequences, autonumbers, custom variables, and running headers, that can help when working with long documents with repeating layouts.

Constructing a directory

The example used here is a directory of members of a professional body. The original was produced a few years ago as a regional directory of members of the Society for Editors and Proofreaders (SfEP), but the ideas behind the template can be applied to any document designed to list information that requires a repeating layout and multiple indexes. Examples include parts lists, business listings, recipes, or anything that needs to be made accessible in an organised fashion. FM's long-document abilities really start to shine in such applications.

Setting up the layout

The directory consists of a section that contains members' details, laid out in a formal fashion, followed by several indexes. Each member's entry consists of name, address and contact details, followed by paragraphs listing work skills, subject matter areas, media handled, qualifications, available equipment, recent clients and, finally, a free-text paragraph.

Figure 1 shows part of a page with fictitious data. Several things are worth noting:

- The sequence of paragraphs for each member's entry is always the same.
- Some paragraphs are prefixed by a run-in heading.
- Reversed headings are used to highlight the start of each member's entry.
- Specific keywords repeat throughout each entry, particularly in the skills and subjects sections.
- The page includes a running header that echoes the surnames of the members listed on the page.

FM provides features to help with all these layout aspects.

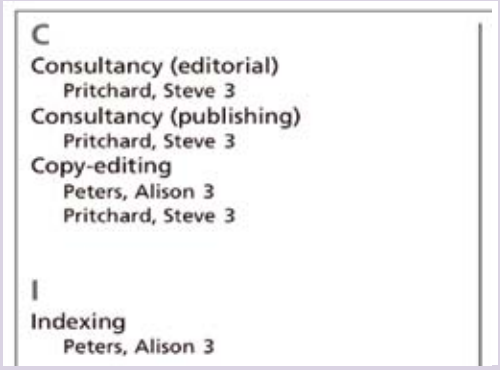
The members' details section is followed by separate indexes that list members by skills, subjects, media, and geographical location. Figure 2 shows a section of the index by skill, created from the page shown in Figure 1.

More than one index

FM is not limited to one index. By default it provides the **Index**, **Author** and **Subject** marker types, which can all be used in indexes, but custom marker types can also be created. Why would one want to do this? As the default marker types imply, one application is to include separate indexes in a book for different topics, such as authors. The example here uses custom marker types for skill, location, and media, as well as the default **Subject** marker type, to give the required four indexes.

Creating a custom marker type is easy: choose **Special>Marker** to display the **Marker** dialog, then choose **Edit**… from the **Marker Type** pop-up menu. The **Edit Custom Marker Type** dialog allows custom marker types to be created, edited, or deleted. New marker types are then available at the book level when the indexes are set up.

In this application, each index lists only one



C
Consultancy (editorial)
Pritchard, Steve 3
Consultancy (publishing)
Pritchard, Steve 3
Copy-editing
Peters, Alison 3
Pritchard, Steve 3

I
Indexing
Peters, Alison 3

Figure 2. An extract from one of the indexes

marker type, as shown in Figure 3 for the index by skills. Indexes are added to the book using **Add>Index Of>Markers** from the book window.

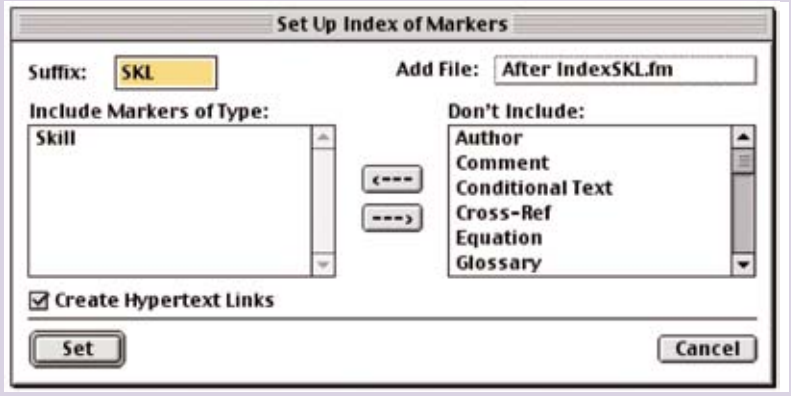


Figure 3. Setting up a custom index

Controlling the paragraph sequence

Although FM 7.x can be used in structured mode to exert rigid control over a document's structure through an appropriate EDD (Element Definition Document), in many cases the time, expense, and complexity of creating an EDD is not justified.

The application described here uses a much simpler approach. Each paragraph shown in Figure 1 has a unique paragraph tag: **Individual** (Peters, Pritchard), **Christian** (Alison, Steve), **Registration** (Registered proofreader, Registered copy-editor), **Address**, **Telephone**, **Skill**, **Subjects**, **Media**, **Qualifications**, **Equipment**, **Clients**, **Freetext**. The **Next Pgf Tag** field in the basic properties of the paragraph designer is set up to sequence these tags. If a paragraph tag **Telephone**, for example, has **Skill** as its next tag, FM applies the **Skill** tag to the paragraph created when **Return** is pressed at the end of a **Telephone** paragraph. This provides a simple and quick means of controlling the formatting of information as it is entered.

Setting up the run-in headings

Figure 1 shows that the **Subjects**, **Media**, **Qualifications**, **Equipment** and **Clients** paragraphs have their titles set as run-in headings. FM explicitly supports run-in paragraphs, but in this case the simplest way to create these headings is to define them as autonumbers in the paragraph tag definition. As previous articles in this series pointed out, autonumbers don't have to be numbers. Figure 4 shows the autonumber definition for the **Subjects** paragraph tag. The **Character Format** field applies an existing character tag **Bold** to the heading.

In combination with FM's 'next paragraph' feature, this means that each member's section can be laid out almost completely just by pressing the **Return** key repeatedly.

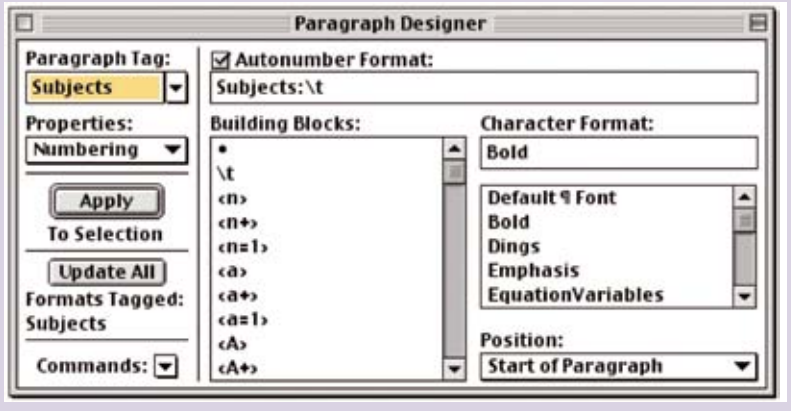


Figure 4. Using autonumbers to create run-in titles

Creating the reversed headings

The reversed headings shown in Figure 1 are one-cell tables. FM offers a wide selection of table formatting controls: in this case the tables are set with no borders and a 70% black fill. This adds a little extra work to the layout, because insertion of the tables cannot be achieved using the 'next paragraph tag' method described above. As all heading tables are identical, however, they can simply be copied and pasted. (Third-party plug-ins are also available that make the insertion of 'boilerplate' layout objects very quick and easy.)

An alternative way of achieving a similar effect without using tables is to include a named graphic from the reference pages and position it behind the text. This wasn't done here, though: it's fiddly, and counts as a 'hack' but, more importantly, three separate paragraph types are used in each heading table, for surname (**Individual**), Christian name (**Christian**) and registration, as explained later.

The text of the three paragraph tags that occupy the heading tables is set in white, using the **Color** field in the **Default Font** pane of the paragraph designer, to achieve the reversed effect. The heading tables are anchored in a 2-point paragraph whose **Next Pgf Tag** is **Address**, so that pressing **Return** in the paragraph containing the table anchor creates a new address paragraph below the table.

Setting up the index entries

Each member is listed by skills, subjects, media and location in the indexes. A lot of index markers therefore have to be entered: a marker of the correct type is required for every entry in the address, skill, subject and media paragraphs of each member. Each must contain the key term as the first-level index entry and the member's name as the second-level entry, separated by a colon. So the copy-editing skill entry for 'Alison Peters' contains '**Copy-editing:Peters, Alison**'.

Anything that reduces the work involved in entering these markers is going to help. To make the process as simple and repeatable as possible, the relevant key terms can be set up as custom variables. Figure 5 shows the **Variables** dialog. Defining a variable called, for example,

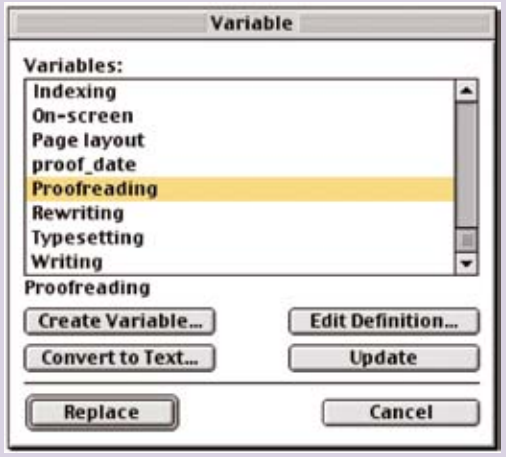


Figure 5. Using variables to control invariant text

Proofreading that contains only the word 'Proofreading' might seem on the face of it a rather strange and pointless thing to do. However, it gives several advantages in relation to indexing:

- Defining key terms as variables in an FM template ensures that they appear identically every time they are used. This means that the indexes won't end up with one set of entries for, say, 'Copy-editing', and another for 'Copyediting'.
- FM enters a word in the **Marker** dialog when it is double-clicked in the body of the document. If the word is a variable, however, FM places it in the **Marker** dialog with only one click: a click saved.
- Double-clicking on a word in a document to place it in the **Marker** dialog only works for single words. Triple-clicking in FM selects a whole paragraph. If some key terms contain multiple words, such as 'Consultancy (editing)' or 'Travel/Exploration', defining them as custom variables allows the whole of the key term to be entered into the **Marker** dialog with only one click.

As the second-level marker information is the member's name for all markers within a member's entry, a sequence such as '**:Peters, Alison**' can be stored on the clipboard and pasted repeatedly into the **Marker** dialog as the markers for each member's information is entered.

Unfortunately, once the index terms are inserted in markers, they are no longer variables, just text, and won't respond to changes in their original variable's definition. FM does allow specific marker text to be searched for, however.

Whenever dealing with repeating terms, phrases or text in an FM application, it's always worth considering using custom variables to contain them, and keyboard shortcuts make them fast to enter, too.

Setting up the running headers

FM has a set of special header/footer variables that can only be used on master pages. There are four of these in FrameMaker 6, **Running H/F 1–4**, and twelve in FrameMaker 7.x. These variables allow running headers and footers to reference other objects, such as the contents of heading paragraphs or chapter titles.

For example, if a document uses the paragraph tags **Heading1** and **Heading2** for the top two levels of heading, defining a running H/F variable as **<$paratext[Heading1,Heading2]>** will display the contents of the most recent level 1 or level 2 heading.

The page headers in Figure 1 show the first and last member's names listed on each page. To create such 'dictionary-style' headers, the first two running header/footer variables are defined as **<$paratext[Individual]>** and **<$paratext[+,Individual]>**. The first variable definition picks up the text of the first instance on the page of the paragraph tag **Individual**, while **<$paratext[+,Individual]>** tells FM to use the text of the last instance of the **Individual** tag it finds on the page. The same technique is used to produce subject headers in the indexes.

It might now be obvious why the member's headings are set up with multiple paragraphs for each member, one for their surname (the **Individual** tag) and one for their Christian name (the **Christian** tag): the surname needs to be available for use in the running headers. To keep all this information on the same line, the **Individual** tag must be defined as a run-in paragraph in the **Pagination** pane of the paragraph designer. This ensures that FM does not create a linefeed at the end of an **Individual** paragraph, so that the **Christian** paragraph can run on in the same line. The same method is used to allow the **Registration** paragraph to run on after the member's Christian name.

The **Default Punctuation** field in the **Pagination** pane tells FM to insert characters automatically at the end of a run-in paragraph. Here we want a comma–space sequence after a member's surname. FM helpfully excludes this default punctuation from a running H/F variable's **$paratext[paratag]** definition. Alternatively, the comma–space sequence could be defined as the autonumber of the **Christian** tag's definition: this approach is used to create the white space before the registration details.

Why not just type the required punctuation? Defining it in the template enforces uniformity, but also, if the punctuation were part of the paragraph's text, the running headers would look like this: **Peters, –Pritchard,**… which is definitely not what is required. C

For more information

All the FM features described in this article are covered in the *FrameMaker User Guide* and online help. The FrameUsers site and mail list are also very useful resources: www.frameusers.com.

Steve Rickaby BSc MISTC has been a freelance technical author and editor for 15 years, and has used FrameMaker for most of that time.
E: srickaby @wordmongers.com
W: www.wordmongers.com