Document Reader File Format

Historical note:

The FRED document reader was written by Simon Cooke, and appeared first in FRED disc magazine, issue 17. It was the first of its kind to display compressed text faster than the equivalent BASIC uncompressed text reader. Since then, two more versions have appeared, one a minor bodge to provide printing facilities for the text reader (when version 1 was written, I didn't have a printer, or printer interface, so I had to give it my best guess on how it all worked. I was wrong), and the other being a slightly updated token set, as well as cleaned up file formats, addition of a "Return to help page" button instead of the "Article menu doesn't exist" message which was put in with the printer fix, and a tidier compressor program to go with it.

Version 1.2 will be the last one to use this format explicitly; work is in progress on a new, multi-format text reader program (the Entropy *Chimera* project) which will not only interpret Document Reader files, but also COMET assembler source files, SAM Small C source, Tasword and Outwrite files, MSDOS text files, Hypertext files, SAM BASIC programs, etc etc. It's all just a matter of time.

Document reader data types

There are three types of file associated with Document Reader – DCP files, MAG files and JYN files. However, the JYN file type only applies to version 1.2 of the document reader program.

***.DCP** files contain a magic string to identify its own presence in the SAM system, as well as details of the number of pages contained in the MAG archive, and the page and offset in memory at which they can be found. **DCP** files are always loaded at 16384. *NB:* if the magic string is not found when the document reader program is run, then it will abort with an error message.

***.MAG** files contain (for version 1.0) article menu data, giving page numbers at which certain topics held in the document can be found, as well as (for all versions) the actual compressed data stream.

***.JYN** *(joined)* files are in actuality merely version 1.2 DCP and MAG files concatenated into a single file, in order to reduce the directory space utilised. The document reader program cannot read these files by itself; a short BASIC program stub is needed to arrange the data in memory in a suitable format before calling the reader.

Version 1.0 format: *.DCP file (load to 16384)

Offset	Function	Length
0	`Magic String      "©1991•Cookie"`	(12)
12	`Number of pages present in document`	(1)
15	`Page data: page number at which document page starts`	(1)
	`followed by an offset in the range 0-16383`	(2)

The page data repeats until all pages in the document have been accounted for. (NB: The "•" character is the space character, or &20 hex. The © symbol is &7F hex). The page data forms a 3-byte address within the SAM's memory for the start of each page of text in the document archive.

Version 1.0 format: *.MAG file (load to 38233)

Offset	Function	Length
0	`Address of article page table`	(2)
2	`Number of articles in article table`	(1)
3	`Maximum article name width (in characters)`	(1)
4	`Start of article name data`	

Article name data can be any length, and each article topic is delimited by the last character of the name having bit 7 set. The actual document text starts immediately after the article text in the file, and is pointed to by the DCP file for speed. The article width field is used in rendering menus. The article page table mentioned is a list determining which article topic corresponds to which page of the document. It is worthy of note that the first page in a MAG archive is the help page displayed when the document reader is first loaded, and that it corresponds to page number zero.

Version 1.1 file format

The version 1.1 file format is in fact identical to the version 1.0 format (the reason for this being so that it would be possible to load in old files for printing purposes). The only difference is that where the v1.0 file would nearly always have an associated article menu, v1.1 files usually have the article menu set to zero entries. Thus, the first 4 bytes of a version 1.1 file will nearly always be:

38233	Address of article page table:	38237
38235	Number of articles found:	0
38236	Max article descriptor width:	0

There will be no article name entries, or page table entries; thus the address pointer at 38233 is invalid, and the magazine page data itself will start at this address.

All a reader has to be able to do in order to read both v1.0 files and v1.1 files reliably is to be able to determine the number of articles in the article menu, and to ignore it is the number of articles found field is set to zero. This criterion is not met by the actual v1.0 Document Reader program, and is the reason why it often crashes if attempts to view files made for v1.1 with no article menu are made.

Version 1.2 file format: *.DCP files (loads to 16384)

```
Offset  Function                                           Length
0       Magic String      "©•1992•ENTROPY"                 (14)
14      Number of pages present in document                (1)
15      Page data: page number at which document page starts  (1)
        followed by an offset in the range 0-16383         (2)
```

The page data repeats until all pages in the document have been accounted for. Version 1.2 files are differentiated from their v1.0 and v1.1 equivalents by the different magic string; this is necessary, as all remnants of the article menu data have been removed from the v1.2 MAG file, and also v1.2 uses a different, slightly more optimised compression token table than versions 1.0 and 1.1 (which both use the same one).

Version 1.2 file format: *.MAG files (loads to 38300)

Whereas the earlier versions held article data (or remnants thereof) here, all such data has been removed in version 1.2 so as to allow the files to take up less disc space. Thus, the only data to be found in the MAG file is the compressed document text itself, as pointed to by the DCP file.

Version 1.2 file format: *.JYN files

As mentioned above, JYN files are composed of a DCP file, with a MAG file tacked on immediately afterwards. To extract the DCP and MAG files, it is necessary to load the file at, say, 49152, and to calculate the DCP length (using the number of pages value stored 14 bytes into the file), poke the DCP section into memory at 16384, and then move the rest of the file down in memory to 38300.

The length of the DCP file is calculated as: (NUMPAGES*3)+18.

As the file is just a DCP file joined to a MAG file, it can be validity checked by looking for the string "©•1992•ENTROPY" at the start of the file.

Text compression method

Document text in a MAG file is compressed using a combination of run-length compression and tokenised strings. Each page is 1344 bytes long (64 characters per line, 21 lines per page). Data bytes below 128 are passed directly to the output routine, bytes with the value 128 are passed onto the run-length subroutine, and the rest (129-255) are passed onto the detokenisation routine. Arguably, better compression could be provided by allowing tokens to also have the value 0-31 – increasing the total number of tokens available from 127 to 159 – but as this is not done by the Document Reader, and there are no further incarnations of the original reader program planned, this is a moot point.

Run-Length Compression

Spaces are run-length compressed. The compressor works by looking for a string of 3 or more spaces in the document text. (In Outwrite and Tasword format text, 64 spaces are used for a blank line, so this can lead to considerable savings). Thus, whenever a code of &80 hex is found in the compressed text, the next byte is taken, and this is used as a counter to print spaces to the screen. Occurences of two consecutive spaces are compressed by the tokenising routine.

For example, if the data in the text stream was (in hexadecimal):

```
21 41 45 80 10 82 83 41
```

This would print out as:

```
!AE••••••••••••••••<TOKEN01><TOKEN02>A
```

The &80 hex is the compressed space character key, and the byte after it – &10 – indicates that 16 spaces are to be printed out. The &82 and &83 bytes are compressed tokens, and until we have gone over the operation of the detokeniser, I have printed them as **<TOKEN>**.

Tokenising Compression

Tokenising compression works by replacing strings of characters with a reference to a dictionary. This dictionary can then be used to recreate the text. It's similar to short-hand, except for efficiency, we do not use just whole words, but word fragments as well.

To compress the text, the tokeniser looks through the uncompressed data to see if it matches one of the "words" or *tokens* in the dictionary. (This is similar to what BASIC does when the editor puts a line into the program. BASIC does it for two reasons; to save space, and to make it much quicker to actually *run* a program. We're only doing it for space reasons).

If a token is found, 129 is added to the token's dictionary reference number, and it takes the place of the equivalent text in the compressed data. Thus all it's necessary to do to decompress the tokens, is to have a copy of the appropriate dictionary, and then to use the token data to access that table.

In the tables below, the dictionary entries are referenced by the normalised (ie with 129 subtracted from them) token numbers; this is to make it easier to write a fast decompressor routine. Each token has bit 7 set on the last character, to mark that the end of the token has been reached.

In the above example text string, the compressed data was:

```
21 41 45 80 10 82 83 41
```

And the uncompressed string was:

```
!AE••••••••••••••••<TOKEN01><TOKEN02>A
```

For version 1.0 and version 1.1, we can now write this as:

!AE••••••••••••••••**screens•screen•**A

And for version 1.2, it is now:

!AE••••••••••••••••**ouldouse**A

The decompressed tokens are in bold.

Version 1.0 and 1.1 Token Dictionary

	00	01	02	03
00	address•	screens•	screen•	•issue•
04	memory•	screen•	•don't•	•SAMCO•
08	•SAMCo•	Coupe•	•FRED•	bytes•
0C	•data•	•it's•	•from•	•SAM•
10	©•199	code•	Code•	Data•
14	ould•	•out•	•had•	Coupe
18	SAMCO	SAMCo	The•	the•
1C	tion	•at•	empt	©199
20	comp	Comp	cons	Cons
24	...•	•you	'll•	ere•
28	You•	•it•	.)•	n't
2C	ity	At•	199	ing
30	een	and	And	ght
34	mag	pro	oum	ove
38	age	•-•	'm•	's•
3C	You	•I•	ant	ial
40	•••	•(	er	,•
44	••	.•	!•	?•
48	A•	or	ss	ee
4C	ch	sh	un	ly
50	th	Th	To	to
54	ow	qu	Qu	Be
58	be	Up	up	Re
5C	re	en	En	us
60	Us	ed	oo	."
64	!"	?"	;•	:•
68	)•	pe	Pe	ir
6C	Ir	my	pp	I•
70	dd	ea	ff	ss
74	it	rr	at	At
78	e•	y•	ic	**<No Token>**

Version 1.2 Document Reader Token Dictionary

	00	01	02	03
00	you'll	ould	ouse	cons
04	comp	I'll	entr	ight
08	•••	ent	ing	out
0C	ang	cei	ial	ant
10	mag	pro	age	I'm
14	'll	had	n't	ean
18	eem	ove	I'd	een
1C	all	oup	SAM	the
20	The	dis	key	ave
24	opy	oil	air	eer
28	ure	ion	vis	ban
2C	mon	hor	ard	ish
30	nal	••	.•	,•
34	's	om	sh	ch
38	ew	ng	ic	tr
3C	cr	it	ff	ss
40	ee	oo	ou	ie
44	ei	'm	nt	fl
48	ph	qu	be	up
4C	re	en	us	ed
50	to	ow	rr	ea
54	ar	pe	mu	th
58	Th	ll	ff	In
5C	in	pp	my	I•
60	or	on	et	sc
64	ut	ex	ce	ck
68	at	At	A•	a•
6C	It	is	Is	su
70	Co	er	de	di
74	bi	ey	sp	go
78	aw	ay	il	op
7C	an	oc	id	**<No Token>**