A Redefinable Telesoftware Format

Document Version 1.2

& Notes on The RTF Decoder V1.1

(part of the Teletext filing system for the BBC micro)

A Redefinable Telesoftware Format

Document Version 1.2

& Notes on The RTF Decoder V1.1
(part of the Teletext filing system for the BBC micro)

Author: C.J.Oswald
11.5.82 & 14.7.82

Published by: Telesoftware Group
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN

Contents

Contents (continued)

Part Three
The B.B.C. Microcomputer's Telesoftware Decoder

Introduction

This Redefinable Telesoftware Format (RTF) provides an adaptable method for transfer of data across a network.

0.1 Usage

The format has been designed principally for the following applications:

(i) Retrieval by microcomputers of programs from larger computers' libraries (eg connection by telephone to a mainframe or Prestel),

(ii) Retrieval by microcomputers of programs available on Teletext,

(iii) The transfer of files from one microcomputer to another by means of radio or telephone,

(iv) Electronic mail.

However the format is deliberately general and can be changed as desired whilst in use. Thus it both suits a wide range of applications and provides for future expansion.

0.2 Other Formats

Several formats have been proposed already for use with telesoftware. They are all essentially fixed, offering little scope for expansion or adaptation. Not one has been agreed upon as a standard.

The RTF is the first format to move one level above these fixed formats, encompassing them by having the ability to learn new formats. Instead of forcing the data into the fixed format being used, the RTF can be adapted to the data and the medium available for its transmission. Features such as data compression are therefore available.

Part One: General Structure

1.1 The RTF Approach

1.1.1 Encoding

Data that is to be transmitted using the RTF is not sent as it stands. Instead, it is encoded into a form which pays attention to the nature of the data and communications medium concerned, and to the possibilities of corruption on transmission.

The unit of a complete transmission is a file. This may be split up logically into segments, which may themselves be split up logically into records.

The encoding is done in three stages:

(i) Naming

All bytes in the file are named; either individually or in ordered groups. Any commands that should be issued with the data are similarly named. Much of this naming operation is done by default (see section 2.2.2).

(ii) Conversion

The file and these commands (including their operands) are converted to an ordered list of such names.

(iii) Dissection

The list of names is physically split up into blocks usually of approximately 1000 bytes each. The names of the commands to start decoding (start block), end decoding (end block), etc are added to the blocks.

1.1.2 Transmission and Decoding

The blocks are transmitted from the sender to the receiver one block at a time, repeating a block if necessary until it is received without any transmission errors.

The receiver decodes each block when it is correctly received, looking up the names that it is composed of in name tables (see section 1.2.3) to find their meanings. The original file is thus gradually rebuilt.

The roles of sender and receiver may be interchanged between the transmission of blocks, allowing conversations between two computers.

1.2 Names

In an eight-bit byte medium, up to 512 distinct names are available for use in the naming stage of encoding. These names have associated meanings to the RTF decoder by default (see section 2.2.2). However different meanings can be given to any of them instead. In this case suitable instructions to the decoder are added to the data to be sent.

The names are split into two types; lone-names and escaped-names. Any name may represent either a command subroutine to be executed (see section 1.3) or a string of zero or more bytes of decoded telesoftware.

1.2.1 Lone-Names

A lone-name is a transmitted byte that does not directly follow one of the names of the escape operator (see section 2.3.2.5).

1.2.2 Escaped-Names

An escaped-name is two (or more) byte transmitted in succession, all but the last byte being names of the escape operator (see section 2.3.2.5).

1.2.3 Name Tables

On decoding, any received lone-name is looked up in the lone-name tables; any escaped-name is looked up in the escaped-name tables. The default settings of the name tables are given in section 2.2.2.

1.3 Command Subroutines

Available to the RTF decoder is a selection of command subroutines (with the ability to add more). An example is the subroutine <DSL> which marks the start of a logical record.

Appropriate command subroutines are chosen from this selection for the format of telesoftware required to be decoded. Links are made in the name tables between these relevant subroutines and the names to be received for their execution. Some suitable links are made by default (see section 2.2.2).

Each command subroutine has a three character code (eg <DEF>), which the subroutine is referred to when it is an operand of another command. This is so that reference can be made to it, whether it is mentioned in the name tables or not.

1.3.1 Structure

An instruction to the decoder appears encoded in the RTF as the name of the relevant command subroutine to be executed, followed by the names of the operands with descriptions of the number and sizes of the operands if they are not implied (see below).

Each subroutine has either a fixed (implied) number of operands or a variable number of operands. Each of these operands consists of either a fixed (implied) number of decoded bytes or a variable number of decoded bytes.

Thus a command subroutine might always have two operands; the first operand always consisting of three decoded bytes and the second operand consisting of a variable number of bytes.

Any variable quantity (ie variable number of operands or decoded bytes in an operand) is preceded by a number giving its value on this occasion.

By default, this is a single decoded byte (0 to 255).

However when the name tables are set up specifically for seven-bit data with parity the number is either a single decoded ASCII hexadecimal digit ('0' to '9', 'A' to 'F') or the decoded ASCII character 'X' followed by two decoded ASCII hexadecimal digits; most significant first. This happens after a <CET>, <TXT> or <VDX> command has been executed (see section 2.3.2.1). On these occasions, any numerical data passed as an operand is given in similar form; a single ASCII hexadecimal digit or 'X' followed by a number of ASCII hexadecimal digits.

The general structure of a command before the naming and conversion stages is:

```
<command subroutine>

<the number of operands, n,
 if not always the same>

<the number of names that the
 first operand will encode to,
 i, if not always the same>

<the first operand>

<the number of names that the
 second operand will encode to,
 j, if not always the same>

<the second operand>
        .
        .
        .

<the number of names that the
 nth operand will encode to,
 m, if not always the same>

<the nth operand>
```

1.3.2 Obedience of Commands

The command subroutines divide themselves into three classes depending on when they may be obeyed by a decoder. Provided that a command is of the expected type at the time it is encountered, then it comes into effect straight away.

Between receiving blocks, RTF decoders await a start block or medium description command. The decoders then make two passes of the data in a block; a first pass to ensure that it has been correctly received, and a second (decoding) pass. Thus there are three modes of decoding.

(i) Start Block and Medium Description Commands

Any bytes received outside the limits of a block are ignored unless they are either the names of a start block command subroutine and its operands or a decription of the medium being used. For the purposes of the first and second passes of the data, the block runs from after a start block command. See section 2.3.

(ii) First Pass Commands

Some commands are obeyed on the first pass of the data in the block. These are of the first pass type.

If the encountered command of this type marks the end of the block, then the extent of the block (for the purposes of the second pass) is up to, but not including, the name of the command subroutine. All other first pass commands will be encountered again on the second pass of the block, thus any changes to the meanings of their names should take effect after their respective occurences within the block. See section 2.3.

(iii) Second Pass Commands

The remainder of commands are obeyed on the second pass of the data in the block. These are of the second pass type. See section 2.3.

Note: In teletext (and some other media), the decoder does not itself need to check that data has been correctly received as there are other mechanisms to do this outside the scope of the decoder. Thus only one pass of the data is needed for decoding.

1.3.3 The Escape Operator and Error in Transmission Commands

The escape operator and error in transmission commands are in special classes of commands of their own. Their functions are described in sections 2.3.2.5 and 2.3.3.4 respectively.

Part Two: The RTF Decoder

2.1 Environment

The name tables and the library of command subroutines described in this section are suitable for a medium capable of sending eight-bit bytes, even if only seven or fewer of those bits are siginificant due to parity, etc. In the cases of teletext and viewdata, no more than one block is permitted on each page (frame). In other media, the basic unit of transmission should be a block.

Idling, start-bit and stop-bit specifications and baud-rates are not rigidly defined by the RTF. Any parity used may be even or odd, and can be checked as part of the decoding operation.

2.2 Initialisation

2.2.1 Initialisation of the Decoder

Prior to decoding a file, the RTF decoder (re)sets the name tables to their default state (see section 2.2.2) unless either this is not the first file to be decoded and a <DND> command was executed on decoding the last file or the user has instructed otherwise. The record of blocks received is cleared, and all buffers are flushed. Internal variables are reset and the references to all command subroutines that are not usually available in the libaray are destroyed.

The decoder is set in a state to recognise only commands to start a block and to describe the medium.

See also section 3.3.

2.2.2 Initialisation of the Name Tables

Usually prior to decoding a file, and whenever a <DEF> command is obeyed, the name tables are initialised to a suitable state for decoding general telesoftware.

All lone names except the escape operators hex 1B, 7C, 9B, FC are set to decode to themselves. The escaped names hex 1B, 7C, 9B, FC are also set to decode to themselves. Thus any eight-bit byte can be sent.

A selection of command subroutines are made available via escaped names.

All the remaining escaped names are escape operators, so that they are effectively ignored. For example, the sequence hex 1B, 00, 6E will produce identical results to the sequence hex 1B, 6E (the command subroutine <DXA> will be executed). Thus the spurious NUL in the first sequence makes no difference.

Any specific demands can thereafter be met in two ways:

(i) by overlaying the tables with further entries (by execution of <CET>, <TXT>, <TX8>, <VDX>, <EMP> command subroutines)

(ii) by changing individual entries (by execution of <DLS>, <DES>, <DLC>, <DEC> command subroutines).

A full description of these command subroutines is given in section 2.3.

The following tables show the entries in the name tables after they have been initialised. This state of the name tables describes the default format used by the RTF decoder.

(i) Name Tables - Default Entries (Hex 00 - 3F)

Value (Hex)	ASCII Char	Meanings Lone	Escaped	Value (Hex)	ASCII Char	Meanings Lone	Escaped
00	NUL	00	<ESC>	20	SP	20	<ESC>
01	SOH	01	<ESC>	21	!	21	<ESC>
02	STX	02	<ESC>	22	"	22	<ESC>
03	ETX	03	<ESC>	23	#	23	<ESC>
04	EOT	04	<ESC>	24	$	24	<ESC>
05	ENQ	05	<ESC>	25	%	25	<ESC>
06	ACK	06	<ESC>	26	&	26	<ESC>
07	BEL	07	<ESC>	27	'	27	<ESC>
08	BS	08	<ESC>	28	(	28	<ESC>
09	HT	09	<ESC>	29	)	29	<ESC>
0A	LF	0A	<ESC>	2A	*	2A	<ESC>
0B	VT	0B	<ESC>	2B	+	2B	<ESC>
0C	FF	0C	<ESC>	2C	,	2C	<ESC>
0D	CR	0D	<ESC>	2D	-	2D	<ESC>
0E	SO	0E	<ESC>	2E	.	2E	<ESC>
0F	SI	0F	<ESC>	2F	/	2F	<ESC>
10	DLE	10	<ESC>	30	0	30	<ESC>
11	DC1	11	<ESC>	31	1	31	<ESC>
12	DC2	12	<ESC>	32	2	32	<ESC>
13	DC3	13	<ESC>	33	3	33	<ESC>
14	DC4	14	<ESC>	34	4	34	<ESC>
15	NAK	15	<ESC>	35	5	35	<ESC>
16	SYN	16	<ESC>	36	6	36	<ESC>
17	ETB	17	<ESC>	37	7	37	<ESC>
18	CAN	18	<ESC>	38	8	38	<ESC>
19	EM	19	<ESC>	39	9	39	<ESC>
1A	SUB	1A	<ESC>	3A	:	3A	<ESC>
1B	ESC	<ESC>	1B	3B	;	3B	<ESC>
1C	FS	1C	<ESC>	3C	<	3C	<ESC>
1D	GS	1D	<ESC>	3D	=	3D	<ESC>
1E	RS	1E	<ESC>	3E	>	3E	<ESC>
1F	US	1F	<ESC>	3F	?	3F	<ESC>

(ii) Name Tables - Default Entries (Hex 40 - 7F)

Value (Hex)	ASCII Char	Meanings Lone	Meanings Escaped	Value (Hex)	ASCII Char	Meanings Lone	Meanings Escaped
40	@	40	<ESC>	60	-	60	<ESC>
41	A	41	<CET>	61	a	61	<DTL>
42	B	42	<TXT>	62	b	62	<DSB>
43	C	43	<TX8>	63	c	63	<DEB>
44	D	44	<VDX>	64	d	64	<DET>
45	E	45	<EMP>	65	e	65	<DSL>
46	F	46	<ESC>	66	f	66	<DSN>
47	G	47	<ESC>	67	g	67	<DNC>
48	H	48	<ESC>	68	h	68	<DST>
49	I	49	<ESC>	69	i	69	<DDT>
4A	J	4A	<ESC>	6A	j	6A	<DCO>
4B	K	4B	<ESC>	6B	k	6B	<DIG>
4C	L	4C	<ESC>	6C	l	6C	<DLA>
4D	M	4D	<ESC>	6D	m	6D	<DLR>
4E	N	4E	<ESC>	6E	n	6E	<DXA>
4F	O	4F	<ESC>	6F	o	6F	<DXR>
50	P	50	<ESC>	70	p	70	<DIR>
51	Q	51	<ESC>	71	q	71	<DES>
52	R	52	<ESC>	72	r	72	<DEC>
53	S	53	<ESC>	73	s	73	<DLS>
54	T	54	<ESC>	74	t	74	<DLC>
55	U	55	<ESC>	75	u	75	<DSA>
56	V	56	<ESC>	76	v	76	<DAC>
57	W	57	<ESC>	77	w	77	<ESC>
58	X	58	<ESC>	78	x	78	<DND>
59	Y	59	<ESC>	79	y	79	<DEF>
5A	Z	5A	<ESC>	7A	z	7A	<ESC>
5B	[	5B	<ESC>	7B	{	7B	<ESC>
5C	\	5C	<ESC>	7C	|	<ESC>	7C
5D	]	5D	<ESC>	7D	}	7D	<ESC>
5E	^	5E	<ESC>	7E	~	7E	<ESC>
5F	_	5F	<ESC>	7F	DEL	7F	<ESC>

(iii) Name Tables - Default Entries (Hex 80 - BF)

Value (Hex)	ASCII Char	Meanings Lone	Escaped	Value (Hex)	ASCII Char	Meanings Lone	Escaped
80	NUL	80	<ESC>	A0	SP	A0	<ESC>
81	SOH	81	<ESC>	A1	!	A1	<ESC>
82	STX	82	<ESC>	A2	"	A2	<ESC>
83	ETX	83	<ESC>	A3	#	A3	<ESC>
84	EOT	84	<ESC>	A4	$	A4	<ESC>
85	ENQ	85	<ESC>	A5	%	A5	<ESC>
86	ACK	86	<ESC>	A6	&	A6	<ESC>
87	BEL	87	<ESC>	A7	'	A7	<ESC>
88	BS	88	<ESC>	A8	(	A8	<ESC>
89	HT	89	<ESC>	A9	)	A9	<ESC>
8A	LF	8A	<ESC>	AA	*	AA	<ESC>
8B	VT	8B	<ESC>	AB	+	AB	<ESC>
8C	FF	8C	<ESC>	AC	,	AC	<ESC>
8D	CR	8D	<ESC>	AD	-	AD	<ESC>
8E	SO	8E	<DE0>	AE	.	AE	<ESC>
8F	SI	8F	<ESC>	AF	/	AF	<ESC>
90	DLE	90	<ESC>	B0	0	B0	<ESC>
91	DC0	91	<ESC>	B1	1	B1	<ESC>
92	DC2	92	<ESC>	B2	2	B2	<ESC>
93	DC3	93	<ESC>	B3	3	B3	<ESC>
94	DC4	94	<ESC>	B4	4	B4	<ESC>
95	NAK	95	<ESC>	B5	5	B5	<ESC>
96	SYN	96	<ESC>	B6	6	B6	<ESC>
97	ETB	97	<ESC>	B7	7	B7	<ESC>
98	CAN	98	<ESC>	B8	8	B8	<ESC>
99	EM	99	<ESC>	B9	9	B9	<ESC>
9A	SUB	9A	<ESC>	BA	:	BA	<ESC>
9B	ESC	<ESC>	9B	BB	;	BB	<ESC>
9C	FS	9C	<ESC>	BC	<	BC	<ESC>
9D	GS	9D	<ESC>	BD	=	BD	<ESC>
9E	RS	9E	<ESC>	BE	>	BE	<ESC>
9F	US	9F	<ESC>	BF	?	BF	<ESC>

(iv) Name Tables - Default Entries (Hex C0 - FF)

Value (Hex)	ASCII Char	Meanings Lone	Escaped	Value (Hex)	ASCII Char	Meanings Lone	Escaped
C0	@	C0	<ESC>	E0	-	E0	<ESC>
C1	A	C1	<CET>	E1	a	E1	<DTL>
C2	B	C2	<TXT>	E2	b	E2	<DSB>
C3	C	C3	<TX8>	E3	c	E3	<DEB>
C4	D	C4	<VDX>	E4	d	E4	<DET>
C5	E	C5	<EMP>	E5	e	E5	<DSL>
C6	F	C6	<ESC>	E6	f	E6	<DSN>
C7	G	C7	<ESC>	E7	g	E7	<DNC>
C8	H	C8	<ESC>	E8	h	E8	<DST>
C9	I	C9	<ESC>	E9	i	E9	<DDT>
CA	J	CA	<ESC>	EA	j	EA	<DCO>
CB	K	CB	<ESC>	EB	k	EB	<DIG>
CC	L	CC	<ESC>	EC	l	EC	<DLA>
CD	M	CD	<ESC>	ED	m	ED	<DLR>
CE	N	CE	<ESC>	EE	n	EE	<DXA>
CF	O	CF	<ESC>	EF	o	EF	<DXR>
D0	P	D0	<ESC>	F0	p	F0	<DIR>
D1	Q	D1	<ESC>	F1	q	F1	<DES>
D2	R	D2	<ESC>	F2	r	F2	<DEC>
D3	S	D3	<ESC>	F3	s	F3	<DLS>
D4	T	D4	<ESC>	F4	t	F4	<DLC>
D5	U	D5	<ESC>	F5	u	F5	<DSA>
D6	V	D6	<ESC>	F6	v	F6	<DAC>
D7	W	D7	<ESC>	F7	w	F7	<ESC>
D8	X	D8	<ESC>	F8	x	F8	<DND>
D9	Y	D9	<ESC>	F9	y	F9	<DEF>
DA	Z	DA	<ESC>	FA	z	FA	<ESC>
DB	[	DB	<ESC>	FB	{	FB	<ESC>
DC	\	DC	<ESC>	FC	|	<ESC>	FC
DD	]	DD	<ESC>	FD	}	FD	<ESC>
DE	^	DE	<ESC>	FE	~	FE	<ESC>
DF	_	DF	<ESC>	FF	DEL	FF	<ESC>

2.3 The Library of Command Subroutines

2.3.1 Alphabetical List of Commands

Code	Section	Command Subroutine
(i) C.E.T. Format		
<CEB>	2.3.3.2(i)	End Block
<CET>	2.3.2.1(ii)	C.E.T. Format
<CSB>	2.3.3.1(i)	Start Block
<CE0>- <CE5>	2.3.3.3(iii)	Eight-bit Byte Adjustments
(ii) Default Format		
<DAC>	2.3.2.4(viii)	Redefine Code to Request That Block is Sent Again
<DCO>	2.3.2.3(vi)	Comment
<DDT>	2.3.2.3(v)	Datatype and Hardware
<DEB>	2.3.2.2(i)	End Block
<DEC>	2.3.2.4(ii)	Change Escaped Name's Meaning (Command)
<DEF>	2.3.2.4(vi)	Revert to Default Format
<DES>	2.3.2.4(i)	Change Escaped Name's Meaning (Decoded String)
<DET>	2.3.2.3(i)	End of File (transmission)
<DIG>	2.3.2.3(vii)	Ignore
<DIR>	2.3.2.3(xii)	Inhibit Run
<DLA>	2.3.2.3(viii)	Load Address (Absolute)
<DLC>	2.3.2.4(iv)	Change Lone Name's Meaning (Command)
<DLR>	2.3.2.3(ix)	Load Address (Relative)
<DLS>	2.3.2.4(iii)	Change Lone Name's Meaning (Decoded String)
<DNC>	2.3.2.4(vii)	Redefine Code to Request That Next Block is Sent
<DND>	2.3.2.4(v)	No Reversion to Default Format Before Next File
<DSA>	2.3.2.6(ii)	Send Block Again
<DSB>	2.3.2.1(i)	Start Block
<DSL>	2.3.2.3(ii)	Start Logical Record
<DSN>	2.3.2.6(i)	Send Next Block
<DST>	2.3.2.3(iv)	Subtitle (of Segment)
<DTL>	2.3.2.3(iii)	Title (of File)
<DXA>	2.3.2.3(x)	Execution Address (Absolute)
<DXR>	2.3.2.3(xi)	Execution Address (Relative)
(iii) The Escape Operator		
<ESC>	2.3.2.5	Escape Operator
(iv) Overlays and Utilities		
<EMP>	2.3.2.1(vi)	Electronic Mail Protocol
<TXT>	2.3.2.1(iii)	Teletext
<TX8>	2.3.2.1(iv)	Teletext (8 bit)
<UER>	2.3.3.4	Error in Transmission
<ULB>	2.3.3.3(i)	Eight-bit Byte Adjustment (Lower)
<URB>	2.3.3.3(ii)	Eight-bit Byte Adjustment (Raise)
<VDX>	2.3.2.1(v)	Videotex

2.3.2 The Command Subroutines Named by Default

All the subroutines in the library of command subroutines may be used for decoding telesoftware after naming. However those listed in this section have links made to them in the name tables by default.

2.3.2.1 Start Block and Medium Description Commands

(i) Start Block <DSB>

Operands: 0, 1 or 2; each of variable length.

<DSB> marks the start of the block of data. The first operand is the number of this block. The second operand (which only needs to be present for one of the blocks in the file) is the number of blocks in the file. The blocks need not be numbered if the <DET> command is used.

Example: <DSB> 02 01 03 01 05
(3rd block in a file of 5 blocks).

(ii) C.E.T. Format <CET>

Operands: None.

<CET> makes the changes to the name tables necessary to decode C.E.T. format telesoftware, and then executes the <CSB> subroutine to mark the start of the block. The changes involved are as follows:

The name tables are prepared for data with even parity (see <VDX> command).

The following commands are given these escaped names:

<CSB>: hex 41 <CEB>: hex 5A <CE0>: hex 30
<CE1>: hex B1 <CE2>: hex B2 <CE3>: hex 33
<CE4>: hex B4 <CE5>: hex 35 <DET>: hex C6

The escaped name hex C9 is set to decode to nothing.
The escaped name hex C5 is set to decode to hex 7C.
The escaped name hex CC is set to decode to hex 0D,0A.
The lone name hex 7D is set to decode to hex 20.
The escaped name hex 7D is set to decode to hex 7D.
The request code for the next block is set to hex 5F (see <DNC> command).

A note is made that this (first) block of the C.E.T. format file is the header block.

(iii) Teletext <TXT>

Operands: 0, 1 or 2; each of variable length.

<TXT> makes the changes to the name tables necessary for data with odd parity, streamlines the tables for teletext and then executes the <DSB> subroutine to mark the start of the block.

The changes involved for odd parity are as follows:

In the name tables, all names that would be a parity error if received are set to be names of the <UER> command subroutine. If they are subsequently received, the <UER> subroutine will alert the decoder of a transmission error. All lone names with correct parity are set to decode to themselves AND-ed with hex 7F.

The command subroutine <ULB> is given the lone name hex E0 and the command subroutine <URB> is given the lone name hex FE. The escaped names hex E0, FE are set to decode to themselves.

The decoder is set to expect numerical information (numbers describing variable operands, load addresses, etc) to be given in ASCII hexadecimal digits (see section 1.3.1)..

As teletext has transmission error checking facilities outside the scope of the decoder, no CRC calculations are made on the incoming data and no second pass is made of it.

A full description of the proposed form that telesoftware will be available in on Ceefax is given in D.J. Rayers' "Teletext Telesoftware".

Example: <TXT> '2' '1' '1' '1' '4'
(Teletext: this is the 1st block of 4).

(iv) Teletext (8 bit) <TX8>

Operands: 0, 1 or 2; each of variable length.

<TX8> sets the name tables for teletext with no parity (ie all eight bits in the byte significant), and then executes the <DSB> subroutine to mark the start of a block.

The changes involved are as follows:

The duplicated lone names of the escape operator are removed from the tables (ie the lone names hex 7C, 9B, FC are set to decode to themselves). The equivalent escaped names (hex 7C, 9B, FC) are set to be names of the escape operator.

As teletext has transmission error checking facilities outside the scope of the decoder, no CRC calculations are made on the incoming data and no second pass is made of it.

Example: <TX8> 00
(8-bit teletext: no block number).

(v) Videotex <VDX>

Operands: 0, 1 or 2; each of variable length

<VDX> makes the changes to the name tables necessary for data with even parity and then executes the <DSB> subroutine to mark the start of the block.

The changes for parity are similar to those made by the <TXT> subroutine, but the command subroutine <ULB> is given the lone name hex 60 and the command subroutine <URB> is given the lone name hex 7E; the escaped names hex 60, 7E are set to decode to themselves.

The request code for the next block is set to hex 30 (see <DNC> command).

Example: <VDX> '1' '3' 'X' '1' '0'
(Videotex: this is the 16th block).

(vi) Electronic Mail Protocol <EMP>

Operands: 0, 1 or 2; each of variable length.

The default setting of the name tables are suitable both for data with all bits in the byte significant and with parity. However the tables may be optimised for eight-bit data for use in electronic mail and inter-computer communication by use of the <EMP> command.

This removes the duplicated lone names of the escape operator (ie sets the lone names of hex 7C, 9B, FC to decode to themselves). The equivalent escaped names (hex 7C, 9B, FC) are set to be names of the escape operator.

The commands <DSN> and <DSA> are given the lone names hex 06 and 15 respectively. The escaped names hex 06 and 15 are set to decode to themselves.

The <DSB> subroutine is then executed to mark the start of the block.

2.3.2.2 First Pass Type Commands

(i) End Block <DEB>

Operands: 0 or 1; of variable length.

<DEB> marks the end of a block of data. If there is an operand, it is a cyclic redundancy check on the contents of a block (see below).

Telesoftware requires an additional check to parity, if it is used, on the integrity of reception. In teletext a cyclic redundancy check (CRC) for each page is to be available. Replacing this in other media, the <DEB> command allows a CRC to be used to ensure data is received correctly before it is decoded.

The CRC associated with the <DEB> command uses the same algorithm as the BBC teletext CRC.

The CRC "sum" is zeroed at the start of a block. The algorithm is applied for each byte received thereafter up to, and including, the names of the operand of the <DEB> command. The "sum" is then compared to the decoded operand. If they differ, then there has been an error in transmission.

The "sum" is a sixteen-bit shift register which has as input the modulo-2 sum of the external input (received bits, including any parity) and the contents of the 7th, 9th, 12th and 16th stages of the register. The shift register is clocked for each bit that arrives. The "sum" can be thought of as a binary number with 16th stage as MSB, 1st stage as LSB.

Note: if the medium allows other methods of marking the end of a block (eg the end of a teletext page is the end of the block) then the <DEB> command is optional.

Example: <DEB> 01 02 56 AD
(CRC on block is hex 56AD).

2.3.2.3 File and Segment Information Commands (Second Pass Type)

(i) End of Transmission of File <DET>

Operands: None.

<DET> indicates that this is the last block in a file. It should be used by media that always send the blocks in order, when the blocks are not individually numbered.

Note: if the medium allows other ways of marking the end of a file, then these may be used instead.

(ii) Start Logical Record <DSL>

Operands: 1 of variable length.

<DSL> marks the start of a logical record of a segment. The operand is the number of this record.

Example: <DSL> 01 02
(This is the second record).

(iii) Title (of File) <DTL>

Operands: 1, 2, or 3; each of variable length.

The operands of a <DTL> command describe the title of the telesoftware file (first operand), its version number (optional second operand), and its date of issue (optional third operand).

Example: <DFT> 02 07 'P' 'r' 'o' 'g' 'r' 'a' 'm' 01 05
(File title is 'Program'; version 5).

(iv) Subtitle (of Segment) <DST>

Operands: 1 of variable length.

The start of a segment is marked by the <DST> command. The operand is the name of this segment.

Example: <DST> 07 's' 'e' 'g' 'm' 'e' 'n' 't'
(Segment title is 'segment').

(v) Datatype and Hardware <DDT>

Operands: 1 or 2; each of variable length.

The operands of a <DDT> command are computer-readable. The first operand gives information on the datatype of the telesoftware (eg the language that it is written in). The optional second operand contain details of the configurations of computers that it is suitable to be run on.

Example: <DDT> 01 04 'B' 'B' 'B' 'C'
(BBC BASIC - full specification).

(vi) Comment <DCO>

Operands: Variable number; each of variable length.

Comments are indicated by the <DCO> command, and may be displayed by the decoder whilst the telesoftware is decoded. The recommended layout is:

1st operand - Name of program
2nd operand - Language
3rd operand - Hardware
4th operand - Date of Publication
5th operand - Source of program
6th operand - Other comments

Example: <DCO> 01 09 'A' ' ' 'p' 'r' 'o' 'g' 'r' 'a' 'm'
(The name of the program is 'A program')

(vii) Ignore <DIG>

Operands: Variable number; each of variable length.

The decoder will ignore the operands of a <DIG> command.

(viii) Load Address (Absolute) <DLA>

Operands: 1 of variable length.

The operand of a <DLA> command is the first absolute address in memory that the decoded telesoftware should be stored at. (This may be over-ruled by the user of the decoder).

The first decoded byte of the operand represents the most significant byte of the address; the fourth decoded byte, the least significant byte.

Example: <DML> 02 20 00
(Store the decoded telesoftware in memory starting from location hex 2000).

(ix) Load Address (Relative) <DLR>

Operands: 0 or 1; of variable length.

The operand of a <DLR> command represents an offset from the first address that decoded telesoftware is being stored at. Decoded telesoftware should now be stored from this offset onwards.

The <DLR> command allows blocks to be received and decoded out of order.

If there are no operands, then the data should not be stored in a contiguous block of memory, but should be passed to a command line interpreter in whatever order the blocks are received and decoded. Typically BASIC programs could be loaded this way.

Example: <DLR> 01 02 01 00
(Store any further decoded telesoftware in memory starting hex 2100 - assuming that the base address is hex 2000).

(x) Execution Address (Absolute) <DXA>

Operands: 1 of variable length.

The operand of a <DXA> command gives the execution address of the file of telesoftware.

Example: <DXA> 02 20 F0
(The execution address is hex 20F0).

(xi) Execution Address (Relative) <DXR>

Operands: 1 of variable length.

The operand of a <DXR> command gives the execution address relative to the load address of the file.

Example: <DXR> 01 F0
(the execution address is hex 20F0 - assuming that the base address is hex 2000).

(xii) Inhibit Run <DIR>

Operands: None.

The <DIR> command inhibits execution of the program immediately after its reception.

2.3.2.4 Format Redefinition Commands (Second Pass Type)

(i) Change Escaped Name's Meaning (Decoded String) <DES>

Operands: 2; 1st of length 1 decoded byte, 2nd of variable length.

The <DES> command changes the entry in the name tables for the escaped name given (1st operand) to decode to the string of bytes given (2nd operand).

Example: <DES> 00 04 'W' 'O' 'R' 'D'
(The escaped name 00 will decode to 'WORD').

(ii) Change Escaped Name's Meaning (Command) <DEC>

Operands: 2; 1st of length 1 decoded byte, 2nd of length 3 decoded bytes.

The <DEC> command changes the entry in the name tables for the escaped name given (1st operand) to represent the command subroutine given (2nd operand).

Example: <DCE> FF 'U' 'E' 'R'
(Reception of the escaped name hex FF indicates that there has been a transmission error - the <UER> command subroutine is executed).

(iii) Change Lone Name's Meaning (Decode String) <DLS>

Operands: 2; 1st of length 1 decoded byte, 2nd of variable length.

The <DLS> command changes the entry in the name tables for the lone name given (1st operand) to decode to the string of bytes given (2nd operand).

Example: <DSL> 00 04 'w' 'o' 'r' 'd'
(The lone name 00 will decode to 'word').

(iv) Change Lone Name's Meaning (Command) <DLC>

Operands: 2; 1st of length 2 decoded bytes, 2nd of length 3 decoded bytes.

The <DLC> command changes the entry in the name tables for the lone name given (1st operand) to represent the command subroutine given (2nd operand).

Example: <DLC> FF 'E' 'S' 'C'
(The lone name hex FF becomes a name of the escape operator).

(v) No Reversion to Default Format Before The Next File <DND>

Operands: None.

<DND> stops the decoder from resetting its name tables to their default values prior to decoding the next file.

(vi) Revert to Default Format <DEF>

Operands: None.

<DEF> resets the name tables to their default values (see section 2.2.2).

(vii) Redefine Code to Request That Next Block is Sent <DNC>

Operands: 1 of variable length.

The medium in use will have some default series of bytes to be sent by the receiver of the telesoftware to inform the sender that the last block was received correctly and to receive the next block. On Prestel this is the byte hex 30; on teletext the receiver must just wait for it; on other media it is usually the byte hex 06 (ACK).

The <DNC> command allows this series of bytes to be changed. It comes into action after the block in which the command resides has been received correctly.

Example: <DNC> 01 5F
(Change the request code for the next block to the byte hex 5F).

(viii) Redefine Code to Request That Block is Sent Again <DAC>

Operands: 1 of variable length.

The <DAC> command changes the series of bytes to be sent by the receiver to request retransmission of the same block (cf <DNC>).

By default, the request code is hex AA, 30, 30 on Prestel and hex 15 (NAK) for most other media.

Example: <DAC> 02 15 15
(Change the request code for retransmission of the same block to hex 15, 15).

2.3.2.5 The Escape Operator <ESC>

The function of the escape operator (<ESC>) is to indicate that an escaped name follows, rather than a lone name. Thus it permits 512 distinct names instead of 256. The escape operator must have at least one lone name and may have escaped names.

2.3.2.6 Commands to Encoders

(i) Send Next Block <DSN>

Operands: None.

The <DSN> command is a request by a distant decoder for transmission of the next block of the file.

(ii) Send Block Again <DSA>

Operands: None.

The <DSA> command is a request by a distant decoder for retransmission of the same block of the file.

2.3.3 Other Command Subroutines

The following command subroutines are not named by default, but may be named by execution of an overlay command, the <DLC> command or the <DEC> command.

2.3.3.1 Start Block Commands

(i) C.E.T. Format Start Block <CSB>

Operands: None.

<CSB> marks the start of a block of telesoftware in C.E.T. format.

2.3.3.2 First Pass Type Commands

(i) C.E.T. Format End Block <CEB>

Operands: 1 of length 3 decoded bytes.

<CEB> marks the end of a block of telesoftware in C.E.T. format. The operand is a check-sum on the bytes in the block (see C.E.T.'s "Prestel Telesoftware Format Recommendations", 1981). It consists of three bytes (which encode to three names).

Example: <CEB> ' ' '4' '5'
(The check-sum is 45).

2.3.3.3 Second Pass Type Commands

(i) Eight-bit Byte Adjustment (Lower) <ULB>

Operands: 1 of decoded length 1 byte.

For media capable only of sending seven-bit bytes with parity, eight-bit data can be sent using the <ULB> and <URB> commands.

<ULB> indicates subtract hex 58 from the next decoded byte (to give a value in the range hex C8 to FF or hex 00 to 27, if only bytes in the range hex 20 to 7F plus parity can be transmitted, as in Prestel).

Example: <ULB> 20
(Decodes to hex C8).

(ii) Eight-bit Byte Adjustment (Raise) <URB>

Operands: 1 of decoded length 1 byte.

<URB> indicates add hex 58 to the next decoded byte (to give a value in the range hex 78 to D7).

Example: <DPL> 20
(Decodes to hex 78).

(iii) C.E.T. Eight-bit Byte Adjustments <CE0> to <CE5>

<CE0> to <CE5> are used in C.E.T. format telesoftware to send eight-bit data bytes via Prestel. They correspond to |0, |1, |2, |3, |4, |5 in the C.E.T. telesoftware format specification.

2.3.3.4 Error in Transmission <UER>

<UER> indicates that there has been an error in transmission (eg there has been a parity error).

THIS SECTION NOW OUT OF DATE See Working Notes on the RTF Decoder V 1.1

Part Three: The BBC Microcomputer's Telesoftware Decoder

As an example of an RTF decoder, this section describes the BBC Microcomputer's telesoftware decoder. It should be read in conjunction with the MOS (machine operating system) specification for the BBC Microcomputer.

Names of variables used by the decoder are given in <angled-brackets>. The 6502 microprocessor has one accumulator (A), two index registers (X and Y) and a status register which contains a carry flag (C).

3.1 User Interface

3.1.1 The TELESOFT Filing System

To enter the TELESOFT filing system, the user types:

*TELESOFT

When using this filing system, the commands listed below have effects as described.

3.1.2 Loading

To decode and load a telesoftware file, the user types:

*LOAD <title> nnnnnn

Decoding will only happen when the telesoftware indicates that its title is the same as that requested. If the title is given as a null-string ("") in the *LOAD command, then the first file that arrives will be decoded and loaded.

If the load address is given (nnnnnn), then the telesoftware will be stored in memory starting at that address. If no load address is given, then the telesoftware will be stored starting at the address indicated by the telesoftware (<DLA> command), or if no such address is indicated, then an error condition will occur.

*LOAD commands (together with BASIC's LOAD commands) go to a section of the TELESOFT filing system (via MOS), where they are converted to a series of low-level calls to the telesoftware decoder.

3.1.3 Running

To decode, load and run a telesoftware file, the user types:

*RUN <title>

The telesoftware file bearing the requested title will be loaded to the address indicated by the telesoftware, and will be executed from the start address also indicated by the telesoftware. If this is not available, then the start address will be assumed to be the same as the load address.

If the title is given as a null-string, then the first file to arrive will be decoded and executed.

As with *LOAD commands, a section of the TELESOFT filing system converts *RUN commands into a series of low-level commands to the telesoftware decoder.

3.1.4 ASCII Commands

To treat a decoded telesoftware file as a series of ASCII commands, the user types:

*EXEC <title>

The telesoftware file bearing the title requested (or the first file that arrives if the title is given as a null-string) will be decoded, and passed to the command line interpreter.

The MOS converts *EXEC commands to a series of low-level commands to the telesoftware decoder directly.

3.1.5 Catalogues

To obtain a catalogue of files available, the user types:

*CAT

The file titles, along with comments decoded to the level set by <commlevel> will be decoded, and displayed on the screen (see sections 2.3.2.3 (vi) and 3.1.6 on <commlevel>). If <commlevel>=0 then a section of the TELESOFT filing system will temporarily set it to 1. The deoder is instructed to decode everything (in so doing the titles and comments will be displayed; any resulting decoded telesoftware produced will be ignored).

3.1.6 Level of Comments

To set the level of comments decoded, the user types:

*OPT 1,xx

<commlevel> is set equal to xx, where <commlevel> is the level of comments to be decoded and displayed on the screen whilst a telesoftware file is being decoded and loaded. It is set to hex FF on power-up. <commlevel>=0 gives no comments and no file titles. <commlevel>=>1 gives file titles and that number of levels of comments.

3.1.7 Initialisation

To stop the decoder resetting itself prior to decoding the next file, the user types:

*OPT 4,yy

yy=0 (default) means the decoder will reset itself prior to decoding the next file (unless instructed otherwise by the last telesoftware file it decoded). yy>0 means that the decoder will not reset itself.

3.2 Internal Organisation of the Decoder

3.2.1 Memory

The decoder program resides in read only memory and runs under the supervision of the TELESOFT filing system. In turn, this filing system is paged with languages and other filing systems (see MOS specification).

The TELESOFT filing system will secure sufficient random access memory workspace for the decoder to run in (2.5 kbyte). Use is made of this memory by the decoder as follows:

6 pages of name tables.
2 pages of heap/queue space.
1 page of soft stack.
1 page of variables.

3.2.2 Communications

All communication to and from the decoder after it has been initialised is by operating system calls. A full description of calls made to the decoder is given in section 3.4.

The decoder makes the following calls itself:

OSASCI - to display comments and file titles.

OSBGET - to get bytes of coded telesoftware from the buffer run by the TELESOFT filing system.

OSARGS - to control aspects of the buffer.

Full details of these calls are available in the MOS and TELESOFT filing system specifications.

3.3 Initialisation of the Decoder

3.3.1 Start-up

On start-up of the TELESOFT filing system, the decoder will be initialised by a call of <startup>.

This will:

(i) initialise the name tables,

(ii) flush all buffers, etc.,

(iii) initialise all variables, including <commlevel>=255 (the level of comments to be decoded), <noreset>=hex FF (a reset will occur prior to decoding the next file).

3.3.2 Prior to Decoding a File

On being instructed to open a file (see section 3.4.1), and provided <noreset> is non-zero, the decoder will:

(i) reinitialise the name tables

(ii) flush all buffers, etc.,

(iii) initialise all variables except <commlevel>, <stackfreep>.

If <noreset> is zero, on opening a file, the decoder will initialise all variables except <commlevel>, <blockreq>, <blockretrans>, <heapfreep>, <stackfreep>. The name tables will remain unaffected; only the queue will be flushed.

Note: <stackfreep> and <heapfreep> are pointers to the next free locations on the soft stack and the heap respectively. <blockreq> and <blockretrans> point into the heap to the sequences of bytes to be sent to request the transmission of the next block and the same block again.

3.4 Operating System Calls to the Decoder

Decoded telesoftware may be obtained from the decoder at a low-level in two ways:

(i) Ordered telesoftware files

(ii) Disordered telesoftware files.

The telesoftware will be made available via the OSBGET channel (see section 3.4.3) in both cases.

In the former (ordered) case, the bytes will be made available in strict order. This is the mechanism used by *EXEC commands.

In the latter (disordered) case, the bytes will be made available in a series of sections. Each section will be ordered within itself, but the sections will be made available in an undefined order. This is the mechanism used by *LOAD and *RUN commands (supervised by sections of the TELESOFT filing system).

3.4.1 Opening Telesoftware Files

(i) Opening an Ordered File

Files opened in an ordered file will appear to be the same as any other sequential file (eg a file opened for sequential reading under the cassette filing system). However catalogue and offset pointer information will not be available.

An ordered file is opened by calling OSFIND with A=hex C0 or A=hex 40. The registers X and Y should point to the title of the file (X lo-byte, Y hi-byte) - a string ending in hex 0D.

The handle hex F8 will be returned in Y.

(ii) Opening a Disordered File

A disordered file is opened by calling OSFILE with A=hex 02. The registers X and Y should point to a control block.

XY+0 to XY+1 point to the title of the file (a string ending in hex 0D).
XY+2 to XY+5 is the preferred load address (if available).
XY+6=0 if preferred load address is available; XY+6<>0 otherwise.

The handle hex F8 will be returned in A.

In either wasy of opening a file, a null-string given as the title will mean that the decoder will decode the first file made available to it.

3.4.2 Obtaining the Load Address (disordered files)

For disordered files, an OSFILE call with A=hex 05 and the registers X and Y pointing to a control block will give the following information:

On return,
XY+0 to XY+1 point to the title of the file (a string ending in hex 0D).
XY+2 to XY+5 is the load address for the next section of decoded telesoftware.
XY+6 to XY+9 is the execution address (if available).
XY+(hex A)=0 if the execution address is available; XY+(hex A)<>0 otherwise.

3.4.3 Obtaining Decoded Bytes of Telesoftware

Decoded bytes are available via the normal OSBGET channel, with Y=hex F8 (the handle).

On exit, C=0 indicates a successful transfer of data (the decoded byte is passed in A).

If the file is ordered, and C=1, then the end of the telesoftware file has been reached.

If the file is disordered, and C=1, then the reason code in A should be examined:
A=hex 00 indicates the end of file has been reached.
A=hex 01 indicates the end of a section has been reached; a new comment is available (it will remain available only until the first byte from the new section is requested).
A=hex 02 indicates the end of a section; no new comment is available.

3.4.4 Obtaining Supplementary Information

Supplementary Information about the file of telesoftware is available by calling OSWORD with A=hex FB.

On calling, the registers X and Y point to a control block.

XY+0 indicates the type of data requested, according to the following code:

hex 00:
hex 01: get new comment (according to the setting of <commelvel>).
hex 02: get file title and version number.
hex 03: get tongue information
hex 04: get the bytes to be sent to request the next block.
hex 05: get the bytes to be sent to request the same block again.

On returning,
XY+0 to XY+1 point to the data requested (they are both zero if the data was not available)
XY+2 to XY+3 indicate the length of the data.

3.4.5 Other OS Calls to the Decoder

There are OSBYTE calls to set the level of comments to be decoded and displayed, and to prevent the decoder resetting itself prior to decoding the next file. See sections 3.1.6 and 3.1.7.

WORKING
NOTES ON:

THE
REDEFINABLE TELESOFTWARE FORMAT
DECODER

(Part of the Teletext Filing System for the BBC Microcompu

Version 1.1

Contents:

C.J. Dell

30th June '82

updated 14th July '82

- No Known Bugs
- Decoder implemented as per RTF document version 1.2 with all command subroutines except: <CEB>, <CET>, <CSB>, <CE∅> - <C <DDT>, <DSA>, <DSL>, <DSN>, <DST>, <EMP>, <TXB>, <VDX> and without CRC algorithm
- Some of these will have to be ad when the decoder is used as part o the Prestel Filing System
- The decoder as it stands more th covers teletext telesoftware require

Notes on The RTF Decoder

(A) Variables

(i) Key: Initialisation { (S) – on soft reset of the decoder
(H) – on hard reset of the decoder

The number of bytes each variable requires is given in brackets after its name

(ii) Zero page variables (transient section):

ztype (2)	}	set to point to current name tabl
zlo (2)		
zhi (2)		
ztempaddr (2)	}	general use zero page location
ztemploc0 (1)		

(iii) Zero page variables (dedicated section):

NONE

(iv) Other variables (all stored in the Global Vector):

stackfreep (2) – stack pointer to the soft stack
(S) (H) = stackbase

P.T.O.

Variables/cont.

extralibindex (2) – pointer to the continuation of the command subroutine library's inde
Ⓗ = ∅

heapfreep (2) – heap pointer
Ⓗ = lospace

queue (2) – tail of queue of decoded bytes po
Ⓢ Ⓗ = hispace

headqueue (2) – head of queue pointer
Ⓢ Ⓗ = hispace

blockreq (2) – pointer into the heap to the code to be sent to request the next block of telesoftware
Ⓢ Ⓗ = ∅

blockretrans (2) – pointer into the heap to the code to be sent to request the same block of telesoftware again
Ⓢ Ⓗ = ∅

comment (2) – pointer into the heap to the latest comment
Ⓢ Ⓗ = ∅

tandv (2) – pointer into the heap to the title and version number of the file of telesoftware being decoded (cf title)
Ⓢ Ⓗ = ∅

blocknumgot (1) – indicates the number of blocks already decoded
Ⓢ Ⓗ = ∅

blockgot (32) – 255 flags indicating which actual blocks have already been decoded
Ⓢ Ⓗ = ∅ except the very least significant bit (set) representing the non-existant block

Variables / cont.

thisblock (1) – the number of the present block being decoded
(S) (H) = 1

blocks (1) – the total number of blocks in the file
(S) (H) = 255

lastblock (1) – flag indicating that the presen block is the last block by numb in the file (note there may still be earlier blocks by number to decode)
(S) (H) = FALSE

sevenbitmode (1) – flag indicating that the decod is operating in seven bit mode (ie operand information will be expected as ASCII hexadecima digits rather than a binary count
(S) (H) = FALSE

operand (1) – flag indicating that the operan of a command subroutine is presently being decoded
(S) (H) = FALSE

ignoredla (1) – flag indicating that a forced load address is being used, so that any <DLA> commands should be ignored
(S) (H) = FALSE

noreset (1) – flag indicating that the decod should not execute a hard reset of itself before the next file (instead only a soft reset should be executed)
(S) (H) = FALSE

norun (1) – flag indicating that the teleoftw should not be run

Variables/cont.

endofsection (1) — flag indicating that the end of a section of a disordered fil has been reached (should be set by <DCO>, <DLA>, <DLR <DXA>, <DXR> command subroutines)
(S) (H) = FALSE

ordernomatter (1) — flag indicating that the ord that the blocks are received i does not matter (should be set by the <DLR> command subroutine)
(S) (H) = FALSE

disordered file (1) — flag indicating that the present file was opened as a disordered file
(S) (H) = FALSE

pass (1) — the pass that the decoder is presently on of the telesoftware (0, 1 or 2)
(S) (H) = 0

commlevel (1) — the level of comments to be decoded
(S) (H) = 255

crc (2) — the Cyclic redundancy check code

checkcrc (2) — temporary locations used by the <DEC> command subroutin

oploc0 (1) — temporary location used by opinfo subroutine

rawloc (1) — temporary location used b the rawbyte subroutine

bytepending (1) — temporary location used by the getteletebyte subroutin

Variables /cont.

buffloc (1) — temporary location used by the buffer control subroutine

memory (4) — the base load address, set either to the forced load addr if there is one, or by the <DLA> command subroutine.
(5) (H) = ∅

relocate (4) — the offset address from memo incremented when each decod byte is given by getdebyte. Set when the <DLR> comma subroutine is execute).
(5) (H) = ∅

title (2) — a pointer into the heap to the title of the file requested (may contain wild cards); zero if no file open
(5) (H) = ∅

calcaddr (4) — the load address of the next section of a file; calculated from memory + relocate
(5) (H) = ∅

execute (4) — the execution address of the file; calculated by <DXA> or <DXR> command subroutines
(5) (H) = ∅

CONSECUTIVE LOCATIONS

noexec (1) — flag indicating that the execution address is not available; cleared by <DX or <DXR> command subroutines
(5) (H) = TRUE

Variables /cont.

(v) The name tables :

These are six one-page tables (see datastructures) which represent the current telesoftware format.

Entries in the type sections are as follows :

& ØØ	ignore byte	
& Ø1	direct decode to 1 byte	(to be found in lo se
& Ø2	direct decode to 2 bytes	(to be found in hi section & lo sec
& ØF	lookup decode (address to be found in hi-lo s	
&1Ø	zeroth pass command	(execution address to
&11	first pass command	be found in hi-lo
&12	second pass command	section)
&1B	escape operator	
&1F	reception error	

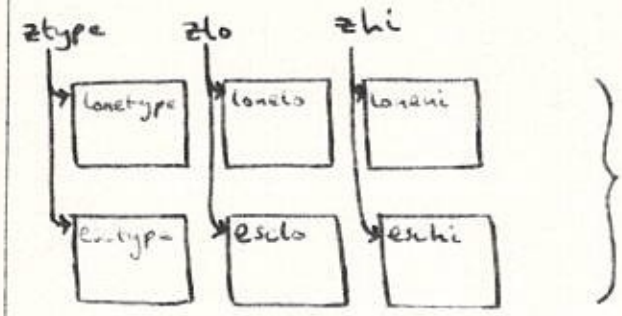
On hard reset (ⓗ) the name tables are set to their default values.

(B) Data Structures

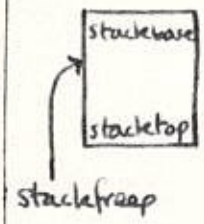
(i) In R.A.M. (static section):

If the TELESOFT filing system hands control over to another filing system temporarily in the middle of decoding a file of telesoftware (or between files if noreset = TRUE), then these data structu must be preserved for a successful restart of the decoder.

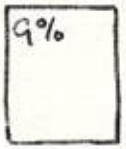
The data structures run from 8% to T% inclusive (provisionally ten pages).



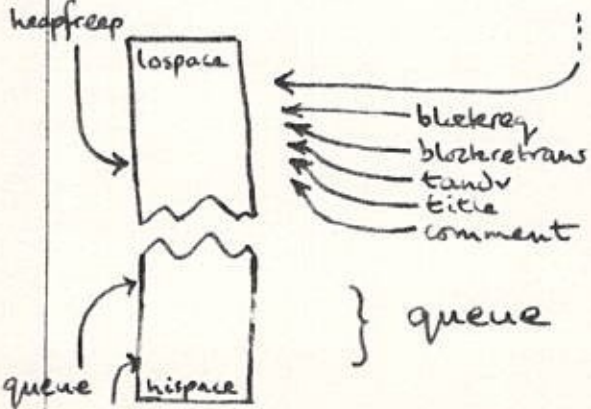
name tables (6 pages)



soft stack (½ page)



global vector (½ page)



heap (jointly 3 pages)

queue

Adding Command Subro

Note:
The index to the library of Comman subroutines may be extended into RA by setting (extralibin to point to the extr index. It must ta the same form as lib index.

(ii) In R.O.M. :

} The index to the library of command subroutines (see below)

The RTF dec
(including H
Library's in
runs from
U% to V%

Entries in the libindex are seven bytes long each (one entry per command subroutine), with those appearing in the default na tables having a one byte entry each in the second part of the index (contli

Byte:	1 2 3	4	5 6	
	name of subroutine (byte 1 is set to zero if this is the end of the index)	type of command (&10, &11 or &12) OR-ed with &80 if command appears in default name tables (see contlibx entry in these cases)	execution address	first entry in escaped name tables (command will also be put at that entry OR-ed with &80)

} entry in contli

© Error Codes (Provisional)

Decimal:

200	Out of Soft Stack Workspace
201	Unexpected End of File
202	Unexpected entry in Name Tables
203	Command Subroutine not available
204	Illegal operand information
205	Out of Heap/Queue Workspace
206	No more data available from source
207	Cannot find start of block
208	No file open

(D) The Interface to the RTF Decoder

The interfacing of the RTF decoder both to the remainder of the TELESOFT filing system and to the outside world is described in these notes.

It has previously been described in:

"A Redefinable Telesoftware Format"
Document Version 1.2 (C.J.Oswald)

and in:

"The BBC Microcomputer Teletext & Videotex User Interface"
4th Draft (A.R.Gordon)

These notes combine the information from both of the documents (with minor modifications) and supercede them on this subject.

After an initialisation of the decoder (CALL initall), interface is by means of tube-compatible operating system calls:

FILES	OSFIND	&40	= CALL openordered
	OSFILE	&05	= CALL getaddresses
	OSFILE	&07	= CALL opendisordered
	OSBGET	&FB	= CALL getlebyte
	OSBGET	&FD	} calls to the remainder of the TELESOFT filing system
OSWORD	OSWORD	&7A	}
	OSWORD	&7A	= CALL getinfo
*OPT	*OPT	&01	= CALL messagelevel
	*OPT	&04	= CALL resetchoice

Thus there are eight entry points to the decoder & the decoder makes two different types of calls itself (both with regard to the page buffer supervised by the remainder of the TELESOFT filing system).

The Interface to the RTF Decoder/unit.

(i) FILES

When the TELESOFT filing system is selected, all file commands are routed to it. Five of these (one OSFIN two OSFILE and two OSBGET calls) concern the RTF decoder directly.

- OSFIND &40 (calls openordered) — OPEN FILE (ORDERED)

  On entry: A = &40
  XY points to the name of the file to be opened (terminated in &0D)

  On exit: A = &FB (handle)

- OSFILE &05 (calls getaddresses) — TELESOFTWAR LOAD ADDRESS

  On entry: A = &05

  On exit: XY points to a control block:
  XY+0, XY+1 } points to the title of the file (a string ending in hex &0
  XY+2, XY+3, XY+4, XY+5 } load address for next secti of telesoftware
  XY+6, XY+7, XY+8, XY+9 } execution address of telesoft file
  XY+&A = 0 if execution address is kn
  <>0 otherwise

- OSFILE &07 (calls opendisordered) — OPEN FILE (DISORDERED

  On entry: A = &07
  XY points to a control block:
  XY+0, XY+1 } points to the title of the file (a string ending in hex

(PTO)

The Interface to the RTF Decoder /cont.

OSFILE &Ø7 /cont.

XY +2
XY +3
XY +4
XY +5
} load address (forced)

XY +6 = Ø if load address is to be for
<>Ø otherwise

On exit: A = &FB (handle)

- OSBGET&FB (calls get telebyte) — DECODED TELESOFT

On entry: Y = &FB (handle)

On exit: C = Ø indicates successful acquisition of a decoded byte (passed in A)
C = 1 indicates some form of EOF condition
A contains the reason code:
A = &ØØ : the whole file has been loaded
A = &Ø1 : end of a section of a disordere file; new comment is available
A = &Ø2 : end of a section of a disorder file; no comment available

- OSBGET &FD (calls ARG's code) — PAGE BUFFER

On entry: Y = &FD (handle)

On exit: C = Ø indicates successful acquisition of a byte from the page buffer (passed in
C = 1 indicates some form of EOF conditio
A contains the reason code:
A = &ØØ : no more data available from sour
A = &Ø1 : end of file & block markers have been pass
A = &Ø2 : end of block marker has been pass
A = &Ø3 : end of buffer has been passe
(a call of OSWORD is needed to replenis

The Interface to the RTF Decoder/cont.

(ii) OSWORD

The two OSWORD calls would logically be more suited to OSBYTE calls; however all OSBYTE calls have been allocated for MOS functions.

- OSWORD &7A (calls ARG's code) — PAGE BUFFER

On entry: A = &7A
XY points to control block:
XY+Ø = &80 Load next page into page buf
XY+Ø = &81 Reload same page
XY+Ø = &82 Mark start of block
XY+Ø = &83 Return to start of block
XY+Ø = &84 Move back one position
XY+Ø = &85 Get absolute page specifie
from XY+1 onwards
(terminated by &ØD)
XY+Ø = &86 Get linked page specified
at XY+1
XY+Ø = &87 Get previous page
XY+Ø = &88 Mark end of block
XY+Ø = &89 Mark start of name.

- OSWORD &7A (calls get info) — TELESOFTWARE INFORMATION

On entry: A = &7A
XY points to control block:
XY+Ø = &ØØ Request file title & version
XY+Ø = &Ø1 Request new comment
XY+Ø = &Ø2 Request code to be sen
to get next block
XY+Ø = &Ø3 Request code to be sent
to get same block ag

On exit: XY points to requested data
XY+Ø is length of data
XY+1 onwards is the data
X = Y = Ø if data requested is not
available.

The Interface to the RTF Decoder/cont.

(iii) *OPT

- *OPT &Ø1 (calls messagelevel) – MESSAGES

  *OPT 1, xx

  The level of comments decoded and made available for display (via OSASCI) is set to 255 by defau The *OPT command resets the level to the value passed by it (xx).

- *OPT &Ø4 (calls resetchoice) – RESET

  *OPT 4, yy

  By default the decoder will reset itself at the end of a telesoftware file.
  The *OPT command with yy ≠ Ø will stop the decoder from resetting itself; with yy = Ø will make it reset itself at the end of a telesoftwar file).

(iv) Initialisation

- CALL initall – INITIALISATION

  A call of initall performs a power-up initialisatio of the RTF decoder.

(E) Command Subroutines

The library of command subroutines may be expanded. An index of all the subroutines is kept (see ROM data structures). This library index may be extended by another page (ie 36 more subroutines) by setting extralibindex to point to the user supplemented index. This second index must be of the same form as the ROM index (libindex)*

A variety of functions are available to command subroutines implemented in RAM:

saveall restoreall	} save and restore A,X,Y, ztempaddr, ztemploc∅
decoded	– decoded byte of teletext in A (load address unalt
opinfo	– operand information returned in A
error	– cause a fatal error (ERR code passed in A)
qbyte	– add the byte in A to the tail of the queu
chopqueue	– empty the queue
heapbytes	– decode no. bytes in A & store them on the hea

(but, of course, any of the functions in the RTF decoder may be used)

The variables (as outlined above) may be used by command subroutines – they should only be altered with care ztempaddr and ztemploc∅ may be altered providing that a call to saveall is made first and a call to restoreall is made last.
ztype, zlo and zhi may be destroyed provided the command subroutine is called via the name tables (as is usual).

The command subroutine should return with carry set only if it has detected the end of the file.

*However no subroutines mentioned in it can appear in the name tables by default.

(F) User Interface

(i) To enter the TELESOFT filing system, the user types:

*TELESOFT

When using this filing system, the commands listed below have effects as described.

(ii) To decode and load a telesoftware file, the user types

* LOAD title nnnnnn

The title may be given as a null-string (" "), in which case the first file that arrives will be decoded and loaded.

If the load address is given (nnnnnn), then the file will be stored in memory starting at that address.

(iii) To decode, load and run a telesoftware file, the user typ

* RUN title

(iv) To treat a decoded telesoftware file as a series of ASCII commands, the user types

* EXEC title

(BASIC programs could be loaded this way)

(v) To obtain a catalogue of files available, the user types

*CAT

The file titles, along with comments decoded to the level set by commlevel will be decoded and displayed on the screen.

(vi) *OPT commands are available, as described above in the interface to the RTF decoder

(vii) BASIC's LOAD and CHAIN commands are converted by BASIC into operating system calls and will work similarly to the *LOAD and *RUN commands described above (but for BASIC rather than machine code programs).

⑥ Assembly arrangement (subject to change!)

PAGE ⟶ TOP(a) — Linking Program ("GO

TOP(a) ⟶ TOP(b) — Assembler programs
+
&100 — ("rtf0" to "rtfn")

&4600 — LOMEM

&65FF — HIMEM

RTF Decoder:

- S% ⟶ T% — RAM workspace (name tables, etc)
- U% ⟶ V% — Assembled program + Library's index
- Z% ⟶ Z%+8 — Zero page workspac

(By default: S% = &6600, U% = &7000, Z% = &

(H) Running Structure