**=============================================================**

**### ###### ####**

**##### ########## ######**

**################## #######**

**#################### ######**

**##### ##################**

**#### ################**

**#### ############# #####**

**##### ###### #####**

**##### ####### #####**

**###### ####### #####**

**###### ######## #####**

**#### ######## #####**

**######## #####**

**####### #########################**

**####### #########################**

**####### #########################**

**####### #####**

**###### #####**

**####### #####**

**###### #####**

**###### #####**

**####### #####**

**####### #####**

**#######**

**#######**

**######**

**#### 7PLUS v2.10 - (c) 1991,92,93 by DG1BBQ**

**updated 2/4/15 by NA7KR to Word format and fixed spelling.**

An intelligent ASCII to binary encoder with enhanced correcting capabilities.

=============================================================

---------------------------------------------

Manual for all system versions of 7PLUS v2.10

---------------------------------------------

7PLUS is hamware. It may only be passed on free of charge and including manual.

7PLUS may be used without limit of duration for non-commercial purposes.

=============================================================

| Axel Bauda |

| DG1BBQ @DB0CL.#HB.DEU.EU |

| Potsdamer Str.43 |

| D-w27576 Bremerhaven |

| Germany |

=============================================================

Contents

-------------------------------------------------------------

CONTENTS

========

The reading-the-manual-phobia (DOCphobia) .................1

Preface .............................................................................1

The development of 7PLUS .............................................2

The ASCII-encoder 7PLUS ..............................................3

Basic steps of a 7PLUS file transfer ................................3

7PLUS-theory ...................................................................5

Encoding algorithm .......................................................5

Character set ................................................................6

Decoding algorithm .......................................................6

Automatic single byte correction ...................................7

The structure of 7PLUS code files ................................7

Header .......................................................................8

Code lines ..................................................................9

Foot line .....................................................................9

Single and multipart 7PLUS code files .......................10

Splitting .......................................................................11

The metafile ................................................................11

The error report ...........................................................12

The correction file ........................................................14

'Saving' damaged error reports and correction files ….15

Compatibility of 7PLUS versions .....................................16

Filename conventions ......................................................17

Usage ...............................................................................18

Encoding ......................................................................18

Extracting .....................................................................19

Decoding ......................................................................20

Compiling a correction file ............................................20

Correcting .....................................................................21

Joining error reports .....................................................22

Recreating an error report from a metafile ...................23

Quick help ....................................................................23

Additional options .........................................................23

-J (Output into single file) .........................................23

-K (Automatic erase) ................................................23

-N (Don't wait for keystroke at termination) ..............23

-P (Use CR as line delimiter) ....................................23

-R (Re-encode a single part) ....................................24

-SYSOP (Decode, even if parts are missing) ...........24

-T (Define file termination string) ..............................24

-TB (Add head and foot text when encoding) ............24

-Y (Assume 'YES' on queries) ..................................26

-# (Produce server info) ............................................26

Contents

-------------------------------------------------------------

Tips for configuring devices and programs for 7PLUS ......26

General conditions .........................................................26

Downloading/uploading in binary mode .........................27

Downloading ...............................................................27

Uploading ...................................................................28

7PLUS on the AMIGA ....................................................28

Tips for down- and uploading of 7PLUS files .....................28

Downloading ..................................................................28

Automatic 'BBS milking' .................................................30

'Listening in' on someone else .......................................31

Uploading .......................................................................32

Uploading correction files ...............................................33

General notes .....................................................................34

Is there a future without errors? .........................................34

II

Preface

-------------------------------------------------------------

The reading-the-manual-phobia (DOCphobia)

=========================================

Despite the fact, I have gone through great lengths in making this manual as complete as possible, I am frequently asked about things that are precisely described in this manual.

I know reading a manual can be cumbersome, but it would save me, as the 7PLUS author, a lot of unnecessary work, if every 7PLUS user who is having trouble, would take THIS MANUAL in hand and read it again thoroughly, before asking me.

Of course, since nobody is perfect, I may have forgotten a few things here and there. Also some parts may not be very intelligible to beginners, although I tried hard to keep it simple.

I cannot supply a MS\_DOS/ATARI/AMIGA or UNIX course in this manual. Therefore you should be familiar with the basics of your computer and terminal program.

Preface

=======

When sending data through the Packet Radio network, errors may occur in spite of the safety mechanisms incorporated into the AX25 protocol. One must bear in mind, that data may pass a lot of unprotocolled interfaces, where a simple impulse can alter it. Also upon transfer the data is passed through quite a lot of memory cells. On computers that don't use memory parity checking, memory errors may go undetected.

Plain text with errors is still well legible for humans. We can correct or ignore errors (as long as numbers aren't falsified) and thus still understand what it meant. Even at an error rate of 10%, we can still get the meaning of the text.

Things are very different when it comes to machine readable data as programs for instance. One error in 100000 bytes may cause the data to be completely worthless. A program affected

By this would probably not run at all or at least not correctly, a picture may be damaged beyond recognition or may not be displayable at all.

If a program was sent in the form of its source text, it may be repaired, but only by those who have a profound knowledge of that programming language. For the mass of users, this is no help.

Another problem is: How do you transfer a program or picture through a network of bulletin board systems (BBS) that are usually just meant to handle plain text?

1

Preface

-------------------------------------------------------------

The answer is to convert it to a format compatible with the BBSs. This task is done by so-called ASCII-encoders (short: encoder). The encoders used in the past had one common problem:

- Did a file get corrupted, it was useless and had to be transferred again.

- Some encoders wouldn't even recognize errors, so the files produces could actually do unexpected things! This ranged from simple crashes to destruction of data on the hard disk.

- Some did recognize that there was an error, but wouldn't tell you where (in what file).

- The overhead produced by those encoders was quite large and ranged between 27% up to 100% (!). To transfer a program of 10K in length took up between 12,7K and 20K, depending on the encoder used.

This situation was not very satisfying, so it was high time for an encoder that could handle the problem

- by using checksums on each and every line of code, thus being able to recognize the exact position of errors,

- by supplying a method of correcting errors without having to resend entire files over and over again,

- by correcting minor errors on the spot, and

- by producing less overhead (ca.15%).

The encoder that does all this - and more - is >> 7PLUS <<.

The development of 7PLUS

========================

7PLUS has been developed on the basis of my previous encoder SEVEN. 7PLUS however has the ability of detecting and correcting errors. These abilities had the price of incompatibility to SEVEN, however. This isn't so grave, since 7PLUS has a lot of advantages compared to SEVEN and now the repeated upload of files that were corrupted in store & forward, is a thing of the past.

Since 7PLUS was incompatible to SEVEN anyway, I implemented a RADIX216 encoding which allows for very compact files.

In the beginning, there was heavy opposition against 7PLUS. Some people were convinced that an encoder that uses characters above 7Fh would create more trouble than do good.

2

The development of 7PLUS

-------------------------------------------------------------

7PLUS established itself as THE standard encoder in Germany very quickly. This proves the critics to be wrong. By now,7PLUS is not only used in Germany, but worldwide. There even are BBSs that use 7PLUS for file serving purposes.

An important aspect is the fact, that 7PLUS is available for the most commonly used computer systems in Packet Radio: IBM\_PC, ATARI\_ST, AMIGA and UNIX systems.

(There also are versions for the Apple Macintosh and Acorn Archimedes. I don't have those versions, so ask around for them in the BBSs)

The ASCII-encoder 7PLUS

=======================

From its conception, the Packet Radio protocol (AX.25) is an8 bit system. When only using 7 bits, the protocol is only used to 87.5% of its capacity. When encoding data only using a 6 or 7 bit algorithm, an unnecessarily high overhead is produced.

7PLUS (all versions)

- exploits the capacity of the network fully by using all 8 bits,

- creates the 7PLUS code lines in a way, that when decoding them, it can determine if a line was transferred correctly,

- detects and protocols what lines were damaged or missing,

- simplifies the exact reconstitution of corrupted files, by using a mainly automated method of correcting (with minimal cooperation between sender and recipient).

Basic steps of a 7PLUS file transfer

===================================

1. OM A encodes the file TEST.TXT into TEST.7PL and uploads it to the Packet Radio network.

2. OM B downloads TEST.7PL and tries to decode it....Unfortunately, the file has been damaged on its travel through the network....

3. OM B's 7PLUS detects the errors in TEST.7PL and produces a so-called metafile (TEST.7MF) where all the good lines are stored in their already decoded form and in addition an error report TEST.ERR. (TEST.7PL may be deleted at this point, since it is not needed for correcting)

4. OM B sends TEST.ERR to OM A.

3

The ASCII-encoder 7PLUS

-------------------------------------------------------------

5. With the help of 7PLUS, the original TEST.TXT and the error report TEST.ERR, OM A creates the correction file TEST.COR. He then sends the correction file to OM B or he uploads it as a bulletin, if it can be useful to others.

6. OM B's 7PLUS then uses TEST.COR to fill in the gaps in the metafile TEST.7MF. If no gaps remain, 7PLUS frees the metafile of the index data contained in it, renames it toTEST.TXT and reconstitutes the file's timestamp.

In case the file TEST.COR has been damaged as well, OM A has to send it again. Since the files TEST.ERR and TEST.COR are usually rather short, this way of correcting damaged files (points 3 through 6 above) is much more effective than resending entire parts again and again until they luckily get through undamaged.

The whole concept of exchanging error report and correction files may sound complicated at first, but it isn't really.

The 7PLUS correcting method is basically comprised of 5 simple, but important steps:

1. "At OM B: Decode and recognize faulty lines and memorize them in an error report TEST.ERR (automated),

2. "Send error report TEST.ERR from OM B to OM A",

3. "At OM A: Compile correction file TEST.COR" (automated),

4. "Send correction file TEST.COR from OM A to OM B",

5. "At OM B: Do the correction" (automated).

If the correction file TEST.COR (for OM B) is downloaded by OM B2 (someone who also downloaded TEST.7PL) does not cover all faulty lines, he just has to run through the described

Steps 3-5, since his own special TEST.ERR has been generated when unsuccessfully tried to correct his TEST.7MF.

Once OM A and OM B have successfully transferred a program or picture file this way, they will see that the whole concept is quite fool-proof.

A helpful OM C, who has successfully decoded TEST.7PL can take the place of OM A by supplying correction files to his friends or to someone looking for it. Also, an automated station (e.g. an F6FBB-BBS running 7PSERV by SM0NCS) may well take over the job of supplying correction files!

4

7PLUS-theory

-------------------------------------------------------------

7PLUS-theory

============

The encoding scheme 7PLUS uses, could well be called a 7- three-quarter-bit encoding, because on the smallest scale, 31 bits of the original data are encoded into 32 bits (4 characters) of the 7PLUS code. That is one of the reason, why 7PLUS is called 7PLUS (Simple, isn't it?)

7PLUS uses a RADIX216-algorithm for encoding.

For 31 bytes of the original data, 32 7PLUS-bytes are produced. This causes an overhead of 1/32 = 3.125%.

Because of the required line delimiters, checksums, head and foot lines, the overhead increases to about 13% to 15%.

Simple encoding algorithms as RADIX64 (UUE), RADIX128 (SEVEN) are done by simple bit shifting operations (multiplications with 2^n and divisions by 2^n). Something, microprocessors can do with great speed.

RADIX216 (7PLUS) is not based on multiples of 2, thus integer divisions and multiplications are necessary which take considerably more time than simple bit shifting.

Some older microprocessors do not have direct commands for multiplication and division of integers, so these operations have to be implemented by using add-shift-loops, which run relatively slowly. Divisions done that way are even slower.

The microprocessors commonly used today (80X86, 680X0) how-ever all have direct commands for multiplication and division of integers. That's why the 7PLUS algorithm will run at acceptable speeds on them.

Encoding algorithm

------------------

When encoding data, 7PLUS interprets 31 bits of the original data as the integer number N. This number is then divided through 216 for three times.

Die divisional rests (A1, A2, A3) and the third quotient (A4)

- can take one of 216 possible values (0 bis 215) and

- represent 4 basic characters (= 4 bytes = 32 bits) of the 7PLUS code.

After encoding of a group of 31 bytes of the original data, a 7PLUS group of 32 basic characters is the result.

The basic characters from the RADIX216 algorithm can however not be used directly as they still contain certain characters they are used for control purposes in the ASCII code. This is why each basic character is assigned a new one that does not collide with any control character of the ASCII code.

5

7PLUS-theory

-------------------------------------------------------------

Character set

-------------

Characters used by the 7PLUS algorithm: (in HEX notation)

21 - 29, 2B - 7E, 80 - 90, 92, 94 - FC (ranges including)

Characters not used:

00 to 20 : Control codes and the space character.

2A : The asterisk (\*) must not be used, because

- by chance - the string '\*\*\*end' could be produced, thus causing the receiving The Box BBS to end storage. (To my shame, I must admit, I did not take the bulletin board systems used outside of Germany into account.. The file termination '/EX' may be produced by chance! I haven't received any complaints about this, however. Axel)

7F : DEL, another control code.

91 and 93 : Some telephone terminal programs mis-interpret these characters as XON and XOFF (11h and 13h).

When having problems storing all characters of the 7PLUS code, see 'Tips for configuring ...'.

Decoding algorithm

------------------

When decoding,

- 7PLUS reads 4 characters of the 7PLUS code ( = 32 bits) and

- regroups them into the four values A1, A2, A3 and A4.

The value N is then rebuilt:

N = A1 + A2\*216^1 + A3\*216^2 + A4\*216^3

The value N now represents 31 bits of the original data (almost 4 bytes).

After reading and decoding a full 7PLUS group of 32 characters, 31 bytes of the original data are reconstituted.

6

7PLUS-theory

-------------------------------------------------------------

Automatic single byte correction

--------------------------------

Since version 1.60, 7PLUS uses a simple, but effective way to correct single byte errors on the spot, when

- one byte per line is wrong, or

- one byte is missing, or

- one byte is superfluous.

Upon decoding of a line, 7PLUS checks its validity by using the 'inner' checksum, which has been there since 7PLUS v1.0 .

In case of an error, 7PLUS tries to rebuild the line by testing all possible combinations (up to more than 14.000 combinations). If a combination is found, where the 'outer' checksum (this checksum has been added since version 1.60 of 7PLUS) tests ok, the result is checked again with the inner checksum, because at this point there may be more than one possible result. If both checksums test ok, then the line has been successfully rebuilt. If not, testing of combinations continues until either a valid combination is found or all combinations have been tested unsuccessfully.

A situation where the inner AND outer checksum test ok and the line is still incorrect, is EXTREMELY unlikely. To add some more safety, the outer checksum is calculated inverse to the inner one (back to front).

The single byte correction may take quite some time (up to three seconds) on slow computers, but if this feature can save the trouble of ordering a correction files, it's well worth the waiting. Also, single byte errors are not that frequent.

I am well aware of the fact that there are better methods.

The trouble is, they either have a much greater overhead or need extensive tables, wich is a problem in situations where there isn't much memory available (e.g. when called from within a terminal program or a computer running a bulletin board system).

The structure of 7PLUS code files

=================================

7PLUS code files are generally comprised of

- a header,

- up to 512 lines of code and

- a foot line.

Each line is concluded with a CR/LF combination (on MSDOS and ATARI; on other systems a line delimiter complying with that system is used, e.g. LF on AMIGA/UNIX).

During transfer via Packet Radio, the line delimiter is always and only the CR character.

7

7PLUS-theory

-------------------------------------------------------------

Transformation of CR/LF (LF with AMIGA/UNIX) into CR is done automatically by the terminal programs.

Exception: When receiving or sending in the binary mode of some terminal programs.

See 'Tips for configuring ...'.

Header

------

(Presentation split due to lack of space)

go\_7+. 005 of 010 TEST.TXT 0010453 ->

+-----+ +-+ +-+ +----------+ +-----+

1 2 3 4 5

-> 6488 166 (7PLUS v2.1) XXXXYZZ

+--+ +-+ +----------+ +-----+

6 7 8 9

1) Start indicator, beginning in the second column.

2) Number of the part, decimal (part 5)

3) Number of parts, decimal (10 parts)

4) Name of original file (TEST.TXT), filled with spaces to a length of 12 characters. Even on system that allow longer filenames, an MSDOS-compatible filename is supplied here! This is to insure the exchange-ability of files between different systems.

5) Length of the unencoded file in bytes (decimal).

6) Number of ASCII bytes in this part. Not used any more since 7PLUS v2.0. It's there only to keep compatibility with older versions.

7) Number of lines per file (hex).

8) Name and version of encoder used.

9) XXXX: Test characters to detect stripping of eight bit. Is also used to indicate that the full filename follows in the next line (since 7PLUS v1.4).

Y: Minichecksum for the header (since 7PLUS v1.5).

ZZ: Additional checksum for single byte correction.

(since 7PLUS v1.60).

(Calculation of checksums, see 7PLUS source code)

8

7PLUS-theory

-------------------------------------------------------------

Code lines

----------

(Presentation split due to lack of space)

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA ->

+------------------------------>

1

-> BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBPPPQQ

>------------------------------++---+

2

1) A: First 7PLUS group (32 7PLUS bytes representing 31 bytes of the original data).

B: Second 7PLUS group.

2) P: Checksum (14 bit) and line numbering (9 bit).

Q: (Since 7PLUS v2.0) Additional checksum for single byte correction (15 bit).

A 7PLUS code file may contain up to a maximum of 512 code lines.

(Calculation of checksums, see 7PLUS source code)

Foot line

---------

stop\_7+. (TEST.P05/0A) [18398BE4] XXXXYZZ

+-------+ +-----------+ +--------+ +---------------+ +-----+

1 2 3 4 5

1) End indicator beginning in second column.

2) Name of the encoded file and overall number of parts.

3) Date and time (timestamp) of the unencoded file in MSDOS format:

32bit-value, bits 0 -4 -> seconds / 2

bits 5 -10 -> minutes

bits 11-15 -> hours

bits 16-20 -> days

bits 21-24 -> moths

bits 25-31 -> years since 1980

4) Reserved.

5) See position 9 in 'Header'.

(Calculation of checksums, see 7PLUS source code)

9

7PLUS-theory

-------------------------------------------------------------

Single and multipart 7PLUS code files

=====================================

7PLUS uses a 9 bit line number for the numbering of the code lines. This allows for a maximum of 2^9 (=512) lines per code file. Since a code line represents 62 bytes of the original data, the maximum size of a file that can be encoded into a single 7PLUS code file is 512 \* 62 = 31744 bytes.

Bigger files are automatically encoded into several 7PLUS code files because of this restriction.

However, 7PLUS already starts splitting at a size of 8557 bytes, so that the 7PLUS code file have a size of about 10K. By using the appropriate split options, 7PLUS can still be ordered to produce bigger files, if required.

Note:

-----

The reason for the restriction of the size of code file to about 10k lies in the not very smart method used in the store& forward traffic between BBSs.

An example:

A message of 40,000 bytes is to be transferred from BBS A to BBS B. If the transfer is interrupted by a link failure (happens quite frequently), the entire transfer has to be repeated from the top, until the link holds through the entire transfer.

As long as the authors of bulletin board systems don’t improve the store & forward method to allow resuming an interrupted transfer without starting all over, it is best to keep the size of files low.

Single part code files are issued a filename consisting of the prefix of the filename of the original file and the extension '.7PL'.

Example: 'FUNKTION.EXE' or 'FUNKTION.PAS' becomes 'FUNKTION.7PL'.

Multipart code files are issued a different extension than single part code files. The extension '.PXX' marks all code files that belong together. 'XX' stands for a hexadecimal number in the range of 01 to FF (1 to 255 dec).

Example: 'XMIT.EXE' may be encoded into 'XMIT.P01', 7PLUS can produce up to 255 parts. Thus, 255 \* 31,744 = 8,094,720 bytes can be encoded with 7PLUS.

For Ham purposes, this is more than sufficient.

10

7PLUS-theory

-------------------------------------------------------------

Splitting

=========

7PLUS can split files in different manners.

It can be ordered to

- split into a specified number of files, where the size of the produced files depends on

- the specified number of files and

- the size of the original file, or

- split into files of a specified size, where the number of files produced depends on

- the specified size of the parts and

- the size of the original file.

How 7PLUS splits can be greatly influenced by the user.

See description of the split options in 'Usage/Encoding'.

The metafile

============

A new feature of 7PLUS are the so-called metafiles. They eliminate a number of processing steps when correcting files.

With older versions of 7PLUS, the user had to keep the unsuccessfully decoded files until he received a correction file.

When the correction file was received, the code files had to be corrected and then decoded. Were there a good number of single byte errors, this could take a considerable amount of time and if the correction was not successful, the procedure had to be repeated. In principal, the code files were decoded three times: once when trying to decode, when correcting and again when finally decoding. This wasn't very effective, as anyone can imagine.

Now, 7PLUS will produce a metafile, when first decoding. The metafile contains all successfully decoded lines (in their already decoded form) and where lines are missing or damaged, the metafile is filled in with NUL bytes.

At this point, the 7PLUS code files may be erased, because they are not needed for correcting.

After decoding, the metafile has the same length as the original file, only where lines are missing, it has gaps where the missing lines can be inserted, when a correction file is available.

11

7PLUS file formats

-------------------------------------------------------------

Some important index information is then appended to the end of the metafile (basically the same information as in the error report), so 7PLUS can look up, where there still are gaps in the metafile.

When a correction file is available, the gaps are simply filled in. The metafile does not even have to be moved for this process, because this is done by direct positioning of the write pointer. The previously inserted NUL bytes are simply overwritten.

Since the metafile does not have to be moved, the process of correcting is now extremely fast. You may get the feeling, nothing has happened!

If all gaps in the metafile are filled in, it is renamed to the name of the original file, truncated to its correct length (to rid it of the additional info 7PLUS put at the end of it; that is also done without moving the file) and its original timestamp is restored.

Metafiles have filenames that consist of the prefix of the original file's name plus the extension '.7MF'. Metafiles must be kept, because without them, correcting errors is not possible!

Note:

On operating systems that do not support the manipulation of file sizes, the index info is not appended to the metafile but written into an extra file with the extension '.7IX'. At least under MS\_DOS, DR\_DOS, OS/2 and TOS (Atari), manipulation of file sizes is possible, so no extra index file is produced.

The error report

================

When 7PLUS code files have been damaged in transfer, 7PLUS does not actually recognize the specific errors, but

- it recognizes which lines are missing or damaged and therefore have to be transferred again,

- 'memorizes' the uncorrupted and decoded lines in the metafile and

- produces an error report, where all necessary information is contained to allow the uploader of the damaged files the compilation of a correction file.

Exception:

If only one byte per line is falsified, superfluous or missing, 7PLUS will correct the error on the spot (so-called 'single byte correction').

12

7PLUS file formats

-------------------------------------------------------------

Example of an error report:

go\_text. TEST.ERR

7PLUS error report: TEST.TXT 113 /Test.TXT/ 34028

01

005 006 007 03D FFF

02

10C 10D 10E FFF

[18398BE4]

00

CRC A17D

stop\_text.

The error report from OM B is practically the request for a correction file from OM A to thus resend the damaged lines, so they can be added to OM B's metafile.

The lines ' go\_text.' and ' stop\_text.' mark the beginning and end of the error report. These markings allow 7PLUS to find and extract the error report from a capture file.

Note:

The markings ' go\_text.' and ' stop\_text.' are also used by my program FFR (Mailbox File Fractionize) to mark regular text for automatic extraction. Only when the extension of the filename stated in the ' go\_text.' line is either '.ERR' or '.COR', it is an error report or a correction file! (Axel)

The header '7PLUS error report...' contains information 7PLUS needs to compile a correction file:

- Name of the original file,

- number of lines per file (split size),

- the full filename (only AMIGA/UNIX since v1.4) and

- the length of the original file (since 7PLUS v1.5).

The two-digit hex values define the part (01 - FF) in which the following lines were corrupted.

An unsplit 7PLUS file is handled as 'part 1' here.

One or more lines containing 3-digit hex values follow (the first line in a code file is regarded as line 000 and the512th as 1FF).

A line number of 'FFF' concludes the list of damaged lines for that part.

This can be repeated for all parts having missing or damaged lines.

Since 7PLUS v2.0 the error report is concluded by the original file’s timestamp enclosed in braces '[]'. The old conclusion

13

7PLUS file formats

-------------------------------------------------------------

'00' is still there for reasons of compatibility to older versions.

After this, the checksum for the error report follows (since7PLUS v1.5) to enable 7PLUS to check the validity of the error report before using it.

The correction file

===================

Upon reception of the error report from OM B, OM A (hopefully) compiles a correction file with it. The correction file contains all the lines that were requested by the error report. OM A then either sends the correction file to OM B or posts it as a bulletin, when he has been receiving the same error reports from more than one OM. In case the error reports differ, OM A can have 7PLUS join them ('-J' option) to one error report and then produce a collective correction file with it.

When using a correction file, 7PLUS will only replace lines that aren't already decoded successfully. Because of this, OM B2 can use the correction file that was intended for OM B as well and without risk. The correction will then be successful, if all the lines that are damaged with OM B2 are successfully replaced.

Example of a correction file:

go\_text. TEST.COR

7PLUS correction: TEST.TXT 34028 113 [18398BE4]

P01:

L005:

sdWKLRjwEK...(line 5 of part 1)

L006:

"rt4rWER7Z...(line 6 of part 1)

L007:

eeERG35ERG...(line 7 of part 1)

L03D:

$%&H$%6TZJ...(line 61 of part 1)

P02:

L10C:

AdD$%SFFDr...(line 268 of part 2)

L10D:

sFG$sFw4rf...(line 269 of part 2)

L10E:

56fZTddfDg...(line 270 of part 2)

P00:

CRC A9B6

stop\_text.

' go\_text.', ' stop\_text.' and checksum: see 'The error report'.

14

7PLUS file formats

-------------------------------------------------------------

The header '7PLUS correction...' contains the following information:

- name of the original file,

- length of the original file (decimal) (since 7PLUS v1.5), and additionally since 7PLUS v2.0:

- the block size in lines (hex) the original file was split into and

- the timestamp of the original file.

A line with an entry of the form ' PXX:' defines, for which of the encoded files the following lines are meant. The code lines are preceded by ' LXXX:' which states the line number in the file. 'XX' and 'XXX' stand for two- and three-digit hex values.

The structure of the error reports and correction files is intentionally held very simple.

'Saving' damaged error reports and correction files

===================================================

When an error report or correction file has been damaged and it has a checksum, 7PLUS will break execution and warn the user about the missing or incorrect checksum.

This does not necessarily mean, the file is unusable. Since the structure of these files is help very simple, it should not be very difficult to erase the faulty parts.

Since the checksum (CRC XXXX) is incorrect, it must also be erased or else 7PLUS will still refuse to use it.

In addition the '-F' option must be issued, when using files manipulated in this manner.

This way it is still possible to at least use the good parts of an error report or correction file, making the next try (requesting new correction file etc.) less voluminous.

If the correction file was downloaded by our fictional OM B2,he may be lucky if the correction file contained more lines than he needed to do the correction. Then all his damaged lines may now be replaced (if the right lines were in the correction file, of course).

15

Compatibility

-------------------------------------------------------------

Compatibility of 7PLUS versions

===============================

7PLUS will decode all files produced with older versions of 7PLUS. Files produced with 7PLUS v1.0 e.g. will still decode without any difficulty with 7PLUS v2.10.

Since version 1.60 the single byte correction feature was added. For this the code lines were fitted with an additional checksum. Because of this change, versions of 7PLUS before v1.4 will not decode the files from version 1.60 and above anymore! There is however the possibility of manually erasing the additional checksums from every line (the last two characters) and to then decode with a version prior to v1.4. Since the greater part of users has a newer version, this incompatibility is actually not a problem anymore. Not many other encoders (if any) are as highly an up- AND down-ward compatible as 7PLUS is.

Error reports and correction files are also up- and downward compatible.

Since 7PLUS v1.5 the error reports and correction files are fitted with a global checksum. It is only checked for validity if the files actually contains a checksum. If 7PLUS does not find a checksum however, it will warn the user:

'no CRC. File may be corrupted or from version earlier than 7PLUS v1.5'.

This only means, that 7PLUS could not figure out, if the file is ok or not. You should take a look at such files and read the chapter ''Saving' damaged ...'.

16

Filename conventions

-------------------------------------------------------------

Filename conventions

====================

Filenames handled by 7PLUS, must comply with the rules concerning filename generation under MS\_DOS, no matter what system 7PLUS is implemented on.

They are always made up of

- a name (8 characters max.) and

- an extension (3 characters max.) separated by a period.

Filenames in 7PLUS-files are always in uppercase!

Example for a filename according to above rules: TERMINAL.EXE

On disk, lower case is used for the filenames, if the file-system allows this. This is very important on systems that do not ignore the case in filenames (UNIX).

To enable transfer of full filenames on systems with name structures different than MS\_DOS, the full filenames are transferred separately (see 'The header'). At the same time, an MS\_DOS compatible filename is created to insure exchange- ability. E.g., the full filename 'Hyper\_Term\_Plus' would be changed into 'HYPER\_TE'.

When encoding a file without splitting it, the encoded file is assigned '.7PL' as the extension. (TERMINAL.7PL)

Split files are assigned '.PXX' as the extension, where XX is a hexadecimal number in the range 01 through FF.

Error reports usually have '.ERR' and correction files '.COR' for an extension. This is very important, because 7PLUS gets its information on what to do from the extension!

- '.ERR' tells the 7PLUS of the uploader to compile a correction file,

- '.COR' tells the 7PLUS of the downloader to do the correction

It is allowed to change names of error reports and correction files. However, the extensions MUST comply with above rules!

If you received a lot of error reports, there is another way of naming these files. The first error report must have '.ERR' for an extension and the following ones must have '.EXX', where XX is a two-digit hex value.

Example:

You received three error reports that all relate to the same upload (TEST.7PL). When extracting through the '-X' option, the first report will be called TEST.ERR, the second TEST.E01 and the third TEST.E02. The three reports can then easily be

17

Filename conventions

-------------------------------------------------------------

Joined into a combined report using the '-J' option for instance.

Analogous to the naming of error reports, correction files may also be numbered. If you receive several correction files that have the same name (and relate to the same upload), 7PLUS will name the first correction file 'TEST.COR', the next 'TEST.C01' etc..

7PLUS-files should always be stored with filenames corresponding to these rules. This is even more important in regard to split files and correcting, because 7PLUS may otherwise not be able to find the files belonging together!

Usage

=====

The following examples apply to an MSDOS system. If using 7PLUS on an Atari\_ST, you don't need to enter '7plus' again in the requester after you double clicked 7PLUS, but only the parameters shown in the examples.

On an Amiga, the CLI must first be started. The syntax is then the same as in the examples, except for differences in the structure of filename and paths.

Encoding

--------

7plus test.txt

7PLUS will encode the file TEST.TXT and (if necessary) automatically split it into files of about 10K in size.

7plus test.txt –s 7PLUS will encode TEST.TXT and split it into files of maximum size (up to 512 lines per file).

7plus test.txt -s 30 Encode and split TEST.TXT into files of 30 lines each.

7plus test.txt -sp 3 Encode and split TEST.TXT into 3 files of about equal size.

7plus test.txt -sb 5000 Encode and split TEST.TXT into files of about 5000 bytes length.

18

Usage

-------------------------------------------------------------

Extracting

-----------

(Extracting = copy 7PLUS files from a capture file)

One can use an editor to extract 7PLUS files from a capture file, but 7PLUS can do this much quicker, safer and more convenient.

The 7PLUS extractor looks for files in the capture file that

- begin with the markings ' go\_7+.', ' go\_info.' or ' go\_text.', and

- end with the line containing ' stop\_7+.', ' stop\_info.' or ' stop\_text.'.

The full command to extract 7PLUS files from a capture file is:

7plus <capturefile> -x [<searchtext>]

Examples:

7plus spsave.c10 -x 999.err 7PLUS extracts (and numbers) all error reports from the capture file 'spsave.c10' if their name contains ‘999.err’. This could be 'TERM999.ERR' or 'MEGA1999.ERR' etc. 7plus a:term.log -x

If no search text is issued, ALL 7PLUS files contained in a capture file are extracted. In this case, the files are extracted from 'a:term.log'.

7plus 91-05-15 -x ### When a fictional search text is issued (e.g. ###), no files are extracted, but a list is generated, showing what files are contained in the capture file. You get a sort of directory of the contents.

When extracting, no matter if the line delimiters in the capture file are CR/LF, CR or LF, the extracted files will have line delimiters in compliance with the system used:

CR/LF on IBM and Atari\_ST and LF on Amiga and UNIX.

If the capture file was created by capturing the data in binary mode, the line delimiters in the file will only be CRs, which will make editing impossible. 7PLUS however is not only able to correctly extract, but also the correct naming of the files is insured.

Along with the 7PLUS code files, error reports, correction files and info files are also extracted. Info files have to be marked manually before uploading them to a BBS by inserting ' go\_info. FILENAME.EXT' as the first line and ' stop\_info.' as the last line. FILENAME.EXT is the filename, the info file will be given, when extracted.

19

Usage

-------------------------------------------------------------

At this point, I would like to bring your attention to the existence of another program of mine that will also do the job of extracting 7PLUS files and more: FFR (Mailbox File Fractionize).

The FFR will also extract bulletins, personal mail and the files of most other encoders (RADIX95, CODEX, 7BIT etc.).

The FFR can be freely configured to work with many bulletin board system. Currently it knows the formats of TheBox,F6FBB, W0RLI, MSYS and many more.. It's almost as popular as 7PLUS around Germany.. (Axel)

Decoding

------

7plus test.7pl

7PLUS decodes TEST.7PL. If there are no errors in

TEST.7PL, 7PLUS will produce a copy of the original file using the MSDOS compatible filename (unless decoding on an Amiga or UNIX system. Then the full filename is used, if available). In case of errors, 7PLUS will produce an error report (TEST.ERR).

If 7PLUS finds a correction file 'TEST.COR' in the same directory, it will use it automatically. If additional correction files exist (TEST.C01, TEST.C02 etc.) they are also used.

See also 'The metafile'.

7plus test.p01

7PLUS will decode all parts (\*.p01, \*.p02 etc.).

See previous example.

7plus test

Automatic mode:

7PLUS will - either DECODE 'TEST.7PL' or 'TEST.PXX'

- or ENCODE the file named 'TEST' (without extension).

In case the decoding was not successful, 7PLUS will locate and use correction files and 7PLUS files thatcould be used to correct the errors.

Compiling a correction file

---------------------------

7plus test.err c:\pr\

7PLUS will compile the correction file 'TEST.COR'. It will look for the original file 'TEST.TXT' in the directory 'C:\PR\'. If no search path is issued, 7PLUS will look for 'TEST.TXT in the current directory. It is not necessary for the uploader of 'TEST.TXT' to keep the encoded files (TEST.7PL or TEST.PXX), because they are not needed to compile a correction file. The only thing, he MUST keep is the original unencoded file 'TEST.TXT'!

20

Usage

-------------------------------------------------------------

Without it, no correction files can be compiled. Also he should remember how he split 'TEST.TXT' to be able to supply entire parts, if they have been lost altogether.

7PLUS automatically splits very large correction files into chunks of about 10k. The size, where 7PLUS begins to split can be influenced with the '-SB' option.

See 'Encoding'

A special situation can arise, when a 7PLUS file was created on a system with non-MSDOS filename structures.

Example:

An OM, who has an AMIGA, encodes 'Hyper\_Term\_Plus'. 7PLUS creates an MSDOS compatible filename from the full filename: 'HYPER\_TE' (the full filename is also transferred in the 7PLUS files, but is not used on MSDOS or Atari\_ST systems).

The 'AMIGA-OM' uploads the file(s) and an 'MSDOS-OM' down-loads it/them and decodes with a version of 7PLUS older than v1.4. In case of errors, the 'MSDOS-OM' sends his error report to the 'AMIGA-OM'. He tries to compile the correction file, but his 7PLUS looks for 'HYPER\_TE' instead of 'Hyper\_Term\_Plus'! In this case, the full name including the path must be specified:

E.g.: 7plus hyper\_te.err df0:utils/Hyper\_Term\_Plus

This problem will not arise, if the 'MSDOS-OM' used 7PLUS v1.4 (or higher), because 7PLUS will add the full filename to the error report.

Correcting

----------

7plus test.cor

7PLUS will use the correction file 'TEST.COR' to fill the gaps in the metafile 'TEST.7MF'. If not all gaps could be filled, 7PLUS creates a new error report. This is the case if the correction file does not cover all gaps or not the right ones. Send the new error report to the uploader and request a new correction file. If 7PLUS finds additional correction files (TEST.C01, TEST.C02

etc.) in the same directory as 'TEST.COR' it will automatically use them.

7plus test.p03 –c 7PLUS also allows the usage of a regular 7PLUS file as a correction file. It will take the lines it needs out of 'TEST.P03' and fill the gaps in the metafile. If gaps remain, 7PLUS will create a new error report, just as stated in the previous example.

21

Usage

-------------------------------------------------------------

7plus test.7mf -c

7PLUS will located the required 7PLUS files (TEST.PXX) and uses them to correct the metafile. Compared to the previous example, 7PLUS will locate all the required7PLUS files and use them to do the correction.

7plus test

Automatic mode:

The automatic mode can also be used here. 7PLUS will locate the files it requires and use them (be it correction files or 7PLUS files that can be used to do the correction).

If the correction file was produced using a 7PLUS version prior to v1.5, 7PLUS will break execution, because it can’t find any checksum. 7PLUS can be forced to use such a file by issuing the '-F' option. You should however take a look at the file, before doing so!

See also ''Saving' damaged...'

Tip for insiders:

If 'TEST.7PL/PXX' was uploaded to a general board and correction files for it turn up from time to time, one could well wait until a fitting correction file turns up. You could as well try to obtain a good copy of the damaged parts from a distant mailbox and use them for the correction.

You may use as many correction files as you wish. Since all successfully decoded lines are stored in the metafile, you may be lucky to eventually fill all remaining gaps.

Joining error reports

---------------------

Joining of error report is useful, when an uploader has received more than one error report. He can use the joined error report to produce a collective correction file which contains all lines that were reported damaged. If all error reports are the same, joining is not necessary, of course.

7plus test.err test2.err –j 7PLUS will join the error reports 'TEST.ERR' and 'TEST2.ERR'.

7plus test.err –j 7PLUS will automatically join all error reports that were created through extraction with the '-X' option. 7PLUS looks for 'TEST.E01', 'TEST.E02' etc.

22

Usage

-------------------------------------------------------------

Recreating an error report from a metafile

------------------------------------------

In case an error report is lost or if you want to send one again (because the first one wasn't answered), it can be recreated from the metafile.

7plus test.7mf 7PLUS reads the index information from the end of the metafile and uses it to create 'TEST.ERR'.

NOTE:

On some systems, 'TEST.7IX' must be specified instead of 'TEST.7MF'. The reason for this is explained in the chapter: 'The metafile'.

Quick help

----------

7PLUS -H (or simply 7PLUS)

A short usage info is displayed. Output will pause after 25 lines and can be continued by pressing return. On some systems, the number of lines output depends on the number of lines displayable on the screen.

Additional options

------------------

-J When encoding, 7PLUS will write all produced 7PLUS parts into a single file. Such a collective file can be identified by its extension 'UPL' (upload). Together with the '-TB'-option, it is possible to produce a ready-to- send file that can be uploaded to the BBS in a single action.

-K With this option, 7PLUS will automatically clean up behind itself and erase all files, that aren't needed any more. Exception: When compiling correction files, error reports will not be erased, because they may still be needed for producing collective correction files.

-N (only ATARI\_ST version)

Usually 7PLUS prompts for a key stroke after termination.

The '-N' option will switch off this prompt. This is useful when 7PLUS is invoked by another program (shell).

-P Encoded files, error reports and correction files will be written with CR as line delimiter. This is necessary with those terminal programs, where 7PLUS-files must be sent in binary mode, because they can't handle 8 bit characters in text mode. This is even more important on the AMIGA or under UNIX , because the standard line separator there is LF! Uploading a file in binary mode, that has LF as a line delimiter, would create an unusable

23

Usage

-------------------------------------------------------------

file on the receiving side, because LFs are always stripped in Packet Radio. The standard line separator in Packet Radio is CR! The result would be a file that contains one extremely long line!

-R In case only one part of a multipart upload needs to be re-encoded (e.g. because it never reached it's destination), it is not necessary to encode the entire original file. 7PLUS is able to re-encode only one specified part.

Examplel: 7plus test.txt -sp 10 -r 5

In this example, 7PLUS only encodes part 5, but not the parts 1-4 and 6-10.

It is VERY IMPORTANT to re-encode in EXACTLY the SAME WAY as with the original upload. Be sure to use the same splitting options!

-SYSOP This option allows the decoding even if not all parts have been received. The places in the meta file corresponding to the missing parts will be left blank and the lines are marked as missing for a later correction.

This option should only be used when unavoidable (the missing parts don't turn up in the BBS even after a week).

-T This option can be used to append a file termination string to 7PLUS files, error reports and correction files to automatically signal the end of the file when uploading.

Example: 7plus test.txt -t /ex

Note: If the terminating string begins with a minus sign '-', the entire string must be put into double quotes ("-EX") because 7PLUS would otherwise try to interpret it as an option.

-TB format.def When encoding, the head and foot texts contained in 'FORMAT.DEF' are automatically added to the 7PLUS files.

Example for 'FORMAT.DEF':

@@top

sb ibm @ ww

%O %p/%q 7+

@@bottom

Send ERR-files to DG1BBQ @DB0CL.#HB.DEU.EU!

73s, Axel.

/ex

24

Usage

-------------------------------------------------------------

The variables have the following meaning:

%n -> Name of the encoded file.

%N -> dito, in capital letters.

%o -> Name of unencoded original file.

%O -> dito, in capital letters.

%p -> Number of the part.

%P -> dito, in hexadecimals.

%q -> Overall number of parts.

%Q -> dito, in hexadecimals.

%% -> The percent sign '%'.

Above example format file would create 7PLUS files looking like this:

sb ibm @ ww

TERM999.EXE 01/04 7+

... This is where the 7PLUS file is ...

Send ERR-files to DG1BBQ @DB0CL.DEU.EU!

73s, Axel.

/ex

The '-TB' option only works when encoding, but not when creating correction files or error reports.

Up to 256 characters may be specified with '@@top' and '@@bottom'.

If you add the '-J'-option the produced files are output into a single file.

A word of warning:

Files created with these options should not be used with TheBox systems. There is a bug in the command parser of TheBox wich causes problems, when uploading 7PLUS files in one go (send command and 7PLUS file together in one file). If you get the error message 'invalid command', you know the bug has struk again. The bug always manifests when the first frame sent to the BBS contains semicolons!

You should erase the file in the bbs then and upload it again by issuing the send command manually step by step.

This bug in the command parser is present in ALL versions of TheBox, even in the test releases of the new TheBox v1.9. If it will be eliminated in the official release of the v1.9, remains to be seen (DieBox v1.9a Feb 23 finally works correctly! It took long enough (Axel)).

The '-TB' option will work fine with any other bulletin board system!

25

Usage

-------------------------------------------------------------

-Y 7PLUS assumes 'YES' on all queries. This option is useful, when 7PLUS is invoked from another program (shell, server).

-# When using 7PLUS with a file server, the problem is to let the server know, how many files were produced when encoding. The server could pre-calculate, how many parts would be created, but 7PLUS can produce a small info file called 7PLUS.FLS containing the information on how many files were created and how the names of the encoded files are. The content of 7PLUS.FLS could e.g. be '10 TEST'.

Tips for configuring devices and programs for 7PLUS

===================================================

General conditions

------------------

To be able to process 7PLUS files, it is important, that the TNC and the terminal program are configured correctly.

With TNC2 (using TAPR firmware), KAM, PK232, PK88, MFJ ect. the following commands must be set to not interfere with 8 bit characters:

8BITCONV ON, AWLEN 8 and LCOK ON

- or equivalent commands accordingly -

The serial link between the computer and the terminal should be set to 8 bits, 1 stop bit, no parity (8n1).

It is advisable to set the stream switch character to $01 (CTRL-A) - on a KAM, the second stream switch should be set to$02 (CTRL-A) - because 7PLUS uses practically all displayable characters for itself.

With stream switch set to $7C (|) e.g.,

- you may experience unwanted stream changes when uploading 7PLUS files!

- the terminal program may detect wrong stream switches when downloading 7PLUS files and thus induce unexpected actions.

With the stream switch set to $01, it is not necessary to use the transparent mode to upload 7PLUS files!

All commercial TNCs and multimode controllers are able to handle 8 bit characters in non-transparent mode!

Character translation tables of terminal programs must be set to 1:1 translation (no translation). Some programs have options to strip or permanently set the MSB (most significant

26

Configuring devices and programs

-------------------------------------------------------------

bit). These options must be set, so the MSB is not stripped and not changed in any way!

Terminal emulations (VT100, ANSI etc.) should not pose any problems with 7PLUS files, because the control sequences used in those emulation usually begin with an escape character (the 7PLUS code does not use the escape character!). Should there be problems with the emulation in spite of that, try switching to a simple TTY-emulation.

If you are still having problems with 7PLUS files, the problem lies with the terminal program. In that case, you should read the chapter 'Downloading/uploading in binary mode'. If that doesn't solve the problems, try to get in contact with the author of the terminal program. In his own interest, he will see to it to bring his program up to date

(this probably won't work with commercial products).

Generally, terminal programs made for telephone use shouldn't be used for Packet Radio.

There are a lot of very sophisticated programs especially made for Packet Radio use. Most of them are free or quite inexpensive. Sometimes the vendors of TNCs supply such programs along with the TNC.

Since the first appearance of 7PLUS, updates have become available for all Packet Radio programs which are now '7plus safe'.

Inspite all critics, this didn't take long at all :-) (Axel)

Downloading/uploading in binary mode

====================================

With some older Packet Radio terminal programs there may be problems with correct downloading or uploading of 7PLUS files. In such cases, the binary mode of those programs must be used. Don't get this mixed up with modem protocols like YAPP, X-, Y, Z-moden etc. Binary mode just describes a mode, where receiving and sending of data can be done without interpretation of the data content and without the use of a special protocol.

Downloading

-----------

When downloading data in binary mode everything received is stored EXACTLY as it is. Since the line delimiter used in Packet Radio is the carriage return (CR), the downloaded files will only contain CRs as line delimiters.

When using the usual capture features of the terminal programs, the conversion of the CR into (CR/LF or LF,

27

Configuring devices and programs

-------------------------------------------------------------

depending on what computer is used) is automatically done by the terminal program.

Since the file produced in binary mode has not undergone the conversion, you won't be able to use a regular editor to cut the 7PLUS files out.

To be able to extract the files anyway, use the '-X' option of 7PLUS. See 'Usage/Extracing'.

Another way is to use a conversion program for line delimiters (e.g. CRLF. Available on request). The files that were converted can then be edited with a regular text editor.

Downloading in binary mode has a useful side-effect:

National (umlauts) and graphical characters can be downloaded even if the terminal program would normally strip them.

Uploading

---------

When uploading 7PLUS files, the '-P' option MUST be used.7PLUS will then use CRs as line delimiters.

Error reports don't need to be uploaded in binary mode, so the '-P' option is not necessary there.

7PLUS on the AMIGA

------------------

Most terminal programs available for the AMIGA are not able to handle 8 bit characters, unfortunately (there are some commendable exceptions, however).

In case of difficulties, use the binary mode of those terminal programs.

See 'Downloading/uploading in binary mode'

Tips for down- and uploading of 7PLUS files

===========================================

A lot of things can go wrong when down- or uploading 7PLUS files. To avoid unnecessary trouble, here a some useful hints.

Downloading

-----------

Before downloading huge files from the BBS, you must make sure your terminal program and tnc are correctly configured.

For this purpose you find ASC.TXT in the distribution archive

28

Tips for down- and uploading of 7PLUS files

-------------------------------------------------------------

of 7PLUS. Upload it to the BBS addressed to yourself and download it again. If it doesn't look EXACTLY like the original any more, check your configuration and read the chapter 'Tips for configuring...'.

If the test went ok, then you're set to download 7PLUS-files.

The simplest way to download 7PLUS-files is to open a capture file and continuously write the received files into it. Note that you should not call the capture file the same as the 7PLUS-files you download! Use a name like 'save file' for instance.

When you have finished downloading the files (did you think of looking for correction files that may already be in the BBS?), you must extract them from the capturefile. For this, an editor can be used or - much more convenient - 7PLUS itself! 7PLUS will not only find and extract the files from the capturefile, but also create the correct filenames for them.

Invoking 7PLUS like this:

'7plus savefile -x'

will do the job of extracting the files from the capturefile.

Now, the files are in the format that 7PLUS needs and you can begin decoding.

Let's see, what we have on disk now: (This is only an example)

TERM999.INF

TERM999.P01

TERM999.P02

TERM999.P03

TERM999.P04

TERM999.COR

TERM999.C01

Ah, fine... You thought of downloading the correction files that was already in the BBS.

The file 'TERM999.INF' was automatically extracted by 7PLUS, because the author had uploaded it with the markings ' go\_info. TERM999.INF' and ' stop\_info.'. That way, you get

the info file along with the 7PLUS files. Practical...

Decoding (including correction) is very simple to do now: '7plus term999'

29

Tips for down- and uploading of 7PLUS files

-------------------------------------------------------------

In case a file called 'TERM999' (without an extension) already exists in the current directory, 7PLUS will try to encode 'TERM999' rather then decode 'TERM999.P01'!

This is why i pointed out, that you should use a name for the capture file that has no relation to the name of the 7PLUS-

files.

If you run into this problem, call 7PLUS this way:

'7plus term999.p01'

If decoding went ok, 7PLUS will tell you: 'Decoding successful! 'term999.exe' 43210 Bytes.'

TERM999.EXE is the decoded file. What you have to do from now, depends entirely on the hints from its originator. As the author of 7PLUS, i don't have anything to do with it anymore from this point on.

Let's suppose decoding did not go allright.

7PLUS says:

'Decoding not successful.'

This just means, that you should take 'term999.err' and send it to the originator of 'term999.exe' and wait for the correction file to return from him.

The meta file 'TERM999.7MF' must be kept at all cost! Without it, you won't be able to do the correction when you receive the correction file.

The 7PLUS files 'TERM999.P01' etc are not needed any more and can be deleted.

In the past, it happened that OMs sent me error reports for programs I have never uploaded and requested the according correction files from me! Let me say this again in full clarity: The one who uploaded the files that were corrupted in Store & Forward, is the one you have to send the error reports to!!! I, as the author of the encoder used, have nothing to do with it. Comparison: A forwarding agency delivers goods that were damaged on the transport. No-one in his right mind would have the idea to hold the builder of the truck used for the transport (Me) liable, when the truck (7PLUS) is ok, but the roads (the network) are not....

Automatic 'BBS milking'

------------------------

Some terminal programs have features that allow to automatically log into a BBS and download bulletins of interest.

30

Tips for down- and uploading of 7PLUS files

-------------------------------------------------------------

When using such features, this is EXTREMELY IMPORTANT:

It happens quite frequently that files downloaded are incomplete because a link failure occurred.

You should in any case check, if all files really are complete. When decoding such incomplete files, much to big error reports are produced! You should take a look into the BBS, if the files really are damaged that much. If you send such a mammoth error report to the uploader, well knowing, most errors are due to link failures, you are stressing his good will. If this happens frequently, he could cease sending correction files at all!

'Listening in' on someone else

------------------------------

Some terminal programs have features that allow capturing everything another station is downloading from a BBS (without the AX25 headers). This is a practical way to keep the load on the BBS/digi repeater down.

7PLUS files that were captured that way, usually contain more errors than are really present in the BBS. This is due to the fact, that not every frame sent to the other station you were 'listening' to, was captured.

However, doubled frames are no problem for 7PLUS. You don’t need to edit them out.

When decoding such a file, you should be aware that it is a VERY NASTY thing to do to send the error report to the original uploader!

Firstly, this causes him unnecessary work and secondly in case he uses the error reports to create a collective correction file, it will contain a lot of lines that are not corrupted anywhere!

Better is, to either

- send the error report to the station you were listening in to, and explain why you are doing so. He will most probably be happy to help out. If he was able to decode the files successfully, he can compile a correction file right away.

If not, you'll have to wait until he got his own correction file,

or

- listen in on other OMs, who are downloading the files you need and use the files to correct the metafile. If errors remain, continue as described above. See 'Correction'

31

Tips for down- and uploading of 7PLUS files

-------------------------------------------------------------

Uploading

---------

As in the previous chapter, make sure your terminal and tnc are configured correctly to handle 7PLUS-files. If you don't take all precautions, the files may not be decodable. So, avoid frustrating the other users an think before acting.

You should write a small info in which you explain,

- what program is contained in the files you upload,

- what system the program is for,

- what hardware is required,

- what compressor (ARJ, LHARC, LHA, PKZIP) was used,

- what files are in the archive and last but not least

- what encoder was used for the upload (7PLUS in this case).

This way one can see right away, if the program is any use for him.

The title of this info should be brief but yet state all needed info, e.g.: 'TERM999.EXE Info'

When uploading to a general purpose board (ALL, SOFTWA etc.), you should also state the system the program runs on: 'TERM999.EXE Info (IBM)'

The info text should be marked, so 7PLUS can extract it along with the 7PLUS files:

go\_info. TERM999.INF

This is where the info is put..... stop\_info.

The filename with ' go\_info. ...' should be spelled in capital letters and comply with the rules for MSDOS/ Atari\_TOS! See 'Filename conventions'.

OK, now you can do the encoding.

Bear in mind that some BBSs restrict the size of bulletins. Ask your SYSOP about this. Take this into regard when splitting the file up ('-s','-sp' or '-sb' option).

By using the '-SB' option, you can directly specify the size of the encoded 7PLUS files or correction files.

Invoking 7PLUS like this: '7plus term999.exe -sb 4000'

32

Tips for down- and uploading of 7PLUS files

-------------------------------------------------------------

tells it to produce files of roughly 4000 bytes each.

When not specifying a split option, 7PLUS will produce files of about 10000 bytes in length.

After encoding is done, you should have the files TERM999.P01 through TERM999.P?? (lets suppose, 12 files were produced) on disk. The files can now be uploaded separately. Here also, you should choose an informative title:

'TERM999.P01 / 0C 7+

Means: TERM952 is encoded with 7PLUS and it's the first file of 12 (0C hex) parts.

Even better, but maybe somewhat cryptic to newcomers:

'TERM999.EXE.SFX.P01/0C.7+'

The best way to read this, is from the rear:

1) It's a 7PLUS-file,

2) of a 12 part upload,

3) the first part,

4) and the resulting program is a self-extracting archive called TERM999.EXE.

After uploading all the files, you should check them by down-loading them right away. They could have become corrupted on the way from your terminal to the BBS! In case they really were corrupted, generate a correction file and upload it. It is not really necessary to upload the entire file that was corrupted.

And now: wait for error reports...

See also 'Additional options / -TB, -J'

Uploading correction files

--------------------------

It has proven to be a good idea to upload correction files twice (in one file).

It is very unprovable that both copies of the correction file get corrupted. In case it does happen, the recipient still has the option to use an editor to put together one good file from the two corrupted ones.

33

General notes

-------------------------------------------------------------

General notes

=============

When downloading 7PLUS files from a BBS, always look for correction files the uploader may have already posted. There is a good chance that the correction file covers the errors.

This saves the uploader unnecessary work.

If you have uploaded 7PLUS files, answering error reports is a question of good manners and Ham Spirit! Files no-one can get a correction file to, are just garbage, clogging up the

BBSs. Of course, there is always the chance that all files came through undamaged. Unfortunately, that isn't very often the case.

Is there a future without errors?

=================================

Should there be one fine day where the Packet Radio network does not produce any more errors that must not really mean the end of 7PLUS.

It should be considered, that

- some BBSs have a throughput of several billion characters,

- a message passes maybe several 100 memory cells and hard disks during its travel from A to B,

- some computers don't use memory parity checking, so memory defects may go undetected for some time,

- the AX25 protocol only has a limited amount of safety against bit errors,

- an affordable network still has a lot of unprotocolled links (TNC<-->computer, TNC<-->TNC in some nodes) and last, but not least

- a simple impulse can cause the alteration of a character!

7PLUS can insure that a file is reconstituted exactly as it was, even if it passed through a lot of BBS.

7PLUS is available for a wide range of systems by now, so it represents a safe and practical way of transferring binary data between different systems.

And it's more compact than the old-style encoders like UUE or RADIX95...

73s, Axel in Bremerhaven, DG1BBQ @DB0CL.#HB.DEU.EU

34