

Toolkit Reference

TPS for Windows

Toolkit Reference
Issue: 2.00
TPS for windows 2018
Copyright Micro SciTech Ltd. 1991-2018

Information in this document is subject to change without notice and does not represent a commitment on the part of Micro SciTech Ltd. The software described in this document is furnished under a licence agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the licence or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by means electronic or mechanical, including photocopying and recording, for any purpose without the express permission of Micro SciTech.

Copyright Micro SciTech Ltd., 1991-2018. All rights reserved.

TPS is a trademark of Micro SciTech Ltd.

Windows, MS-DOS, Windows NT, Windows for Workgroups and Windows 95 are registered trademarks of Microsoft Corporation.

Introduction	3
Installation	4
General Information	5
Command line Options	5
Input filenames	7
Output filenames	7
Packet Format, TPS System Settings	8
Premature Termination	9
Error Returns	9
Toolkit Utilities	10
PKTREDU	10
FTOPKT	12
PKTTOF	14
COLTOPKT	15
PASPLICE	16
EXPERR	17
EXPTIME	18
FILEREDU	19
Syntax Summary	20

1. Introduction

The TPS Toolkit comprises TPS specific utilities to aid in the post-flight analysis of data. For instance, the packet reduce utility 'PKTREDU' can reduce a TPS packet data file to a manageable size for fast replay and quick data analysis. The Toolkit also comes with some utilities to enhance TPS's capability such as the file-to-packet 'FTOPKT' converter which converts an ordinary file into data packets, TPS can then replay them for analysis. The packet-to-file utility 'PKTTOF' performs the reverse process and converts packets back to files.

The full suite of utilities is as follows:

PKTREDU	Packet file reduction and format conversion
FTOPCK	File to packet conversion
PKTTOF	Packet to file conversion
COLTOPKT	Tabular data file to packet conversion
EXPTIME	Parameter record and error file time conversion
EXPERR	Error code descriptive expansion
PASPLICE	Parameter record file splicer
FILEREDU	Text file reduction

1.1. Installation

These utilities are command line executables (Windows 'CMD' command - used to be the 'DOS' Window) and are provided in the TOOLKIT sub-folder of TPS, e.g. folder TPSDIST\TOOLKIT or TPSOPEN\TOOLKIT or, indeed, any folder you unzip the TPS files to.

2. General Information

All programs are run from the Windows command-line.

The general form to run a utility is

utility option:value option:value

Where

utility is the utility name, such as FTOPKT

option is one of the set of options such *infile*

value is any number or text, normally a filename

Example

To run the file-to-packet conversion program, you would enter the following

FTOPKT infile:myfile

where myfile is absolutely any disk file.

2.1. Command line Options

All utilities have one or more command line options. For most utilities, an input filename is required. Output filenames are optional.

Options can be entered in one of several equivalent ways

/option:value
option:value
option=value
/option=value

Throughout this manual, only the */option:value* form will be used. The other forms are equally valid but, when passing the options to a batch file (file extension .BAT), the equate sign is not interpreted correctly.

There are only a small set of options, some are common to most utilities, e.g. the input filename option *infile* and the output file option *outfile*.

The full list of options is detailed below:

```
infile
infile1 PASPLICE only
infile2 PASPLICE only
outfile
informat      PKTREDU only
outformat     PKTREDU only
start
end
every
page
```

Example

Suppose you use the below batch file 'FRE.BAT' to run fileredu. This batch file passes the input filename as the first parameter.

```
rem FRE.BAT: Executes the FILEREDU toolkit program
fileredu %1 every:1
```

Then typing the command using the equate symbol as in:

```
FRE infile=p11.12
```

will return the error:

```
Unrecognised option: 'p11.12'
```

whereas typing the command using a colon separator, i.e. `FRE infile:p11.12` is acceptable.

It may seem superfluous to even mention the use of the equate separator when not the documented or preferred method. However, not everyone will want to use batch files to embed execution of toolkit programs and, furthermore, the specification of optional values using an equate symbol is more readable and also common to other applications.

Abbreviated Options

All options can be abbreviated to only one or two leading characters. For example, the `every` option can be replaced by `ev`. Note, `every` cannot be abbreviated to the single character `e` because then it would become indistinguishable from the `e` abbreviation of the `end` option and, hence, ambiguous.

Example

The file reduction utility 'FILEREDU' extracts lines from an ordinary text file (as opposed to a packet file - see PKTREDU for the latter). TPS generates plain text files when recording parameters, often these files can contain thousands of lines of which you may only require, say, every other line. You may also wish to view, say, only the first 100 or so lines.

This example shows FILEREDU being used to extract lines every other line from lines 10 to 100 in the file myfile.txt. The full command would read as follows:

```
fileredu infile:myfile.in outfile:myfile.out start:10 end:100 every:2
```

Where the `every` option instructs FILEREDU to read a line, skip the next line (skip 1), then read another line etc, i.e. every two lines are processed.

The abbreviated form of this command could be written as:

```
fileredu in:p11.12 out:p1112.new st:10 en:100 sk:1
```

Furthermore, to really cut-down on the typing effort, you could write a batch file (file extension .BAT) to save typing in the option names. An example file FRE.BAT is shown below

```
rem FRE.BAT: Executes the FILEREDU toolkit program
fileredu in:%1 out:%2 st:%3 en:%4 sk:1
```

Which would then be executed by typing:

```
FRE p11.12 p1112.new 10 100 1
```

This does, however, require you to always enter all the options. For instance, if you simply typed the following, missing off the end and every option values:

```
FRE p11.12 p1112.new 10
```

you would get the error

```
Unrecognised option: 'every:'
```

2.2. Input filenames

All utilities requiring a file as input, must be supplied with an input filename using the `infile` option, as there are no default filenames. The exception to this rule is the EXPERR utility (expands TPS error codes) which takes its input from the standard TPS error file TPSEERR.TXT.

Example

To convert the file TEXTFILE.TXT to packets you would use the FTOPKT utility as follows:

```
ftopkt infile:textfile.txt
```

By default, the output of FTOPKT is to file FTOPKT.DAT.

2.3. Output filenames

All utilities which generate output data will write the data to a default file if the user does not enter an outfile filename. E.g. PKTTOF will write its data to PKTTOF.DAT if the user does not specify the *filename* as in `outfile:filename`. The default filename is always the same as the root but with an extension of either TXT, DAT or BIN dependant upon the output file format.

If an outfile already exists, you will be prompted as to whether you wish to overwrite the existing file. Any response other than 'y' or 'Y' for yes is interpreted as a negative response and the utility will terminate immediately.

Example

To convert the file TEXTFILE.TXT to packets and save the packet data in the file PKTDATA.DAT, you would use the FTOPKT utility as follows:

```
ftopkt infile:textfile.txt outfile:pktdata.dat
```

Note, FTOPKT writes TPS 'raw bin' format packets according to the currently configured TPS packet format - the packet file can be subsequently be replayed on TPS.

2.4. Packet Format, TPS System Settings

The following utilities,

```
PKTREDU
FTOPKT
PKTTOF
COLTOPKT
```

either read or write packet data. Therefore, each utility requires the packet format information. When using TPS, this is configured online using the *Transfer, Pkt* format menu item. The packet format is a TPS system setting and is optionally saved when you quit TPS and respond *Yes* to the *Save System settings?* prompt. TPS actually saves the information in the file PAGE0000.BIN resident in its home directory. However, a default copy, as supplied, is placed in the TOOLKIT folder. Nevertheless, if you change page 0, you should always ensure the toolkit utility always use the PAGE0000.BIN pertinent to the packet format of the data you are processing with that utility. Commonly, you may use packets with a different synchronisation pattern and packet length.

Example

If you were running the utility FTOPKT from a project data directory C:\TPSDIST\PROJECT which housed the file MYFILE.TXT and had a copy of the FTOPKT.EXE also resident in this directory, then you would explicitly tell FTOPKT to look in the TPS home directory TPSDIST for its packet format information, by adding the page option as follows:

```
FTOPKT infile:myfile.txt outfile:myfile.pkt
page:c:\tpsdist\page0000.bin
```

This example would convert MTFILE.TXT to a packet data file MYFILE.PKT, using the packet format stored in C:\TPSDIST\PAGE0000.BIN.

Alternatively, you could just copy PAGE0000.BIN to your current working directory. However, if you change the packet format online using TPS, only the master version of PAGE0000.BIN in the TPS home directory will be changed, not the copy. Therefore, you would need to make a copy every time you change the packet format.

2.5. Premature Termination

Most, utilities can be prematurely terminated by depression of the Q or q keys denoting quit. The standard <CTRL>+<C> abort key combination will also perform the same function.

2.6. Error Returns

If a utility terminates before completion, either due to an error or triggered by an operator (see above), then the utility will return a non-zero error code which can be tested for within batch files by use of the `if errorlevel` statement. E.g., the below batch file will display the error message `!! Error encountered running FILEREDU !!`

```
rem FRE.BAT: Executes the FILEREDU toolkit program
fileredu in:%1 out:%2 st:%3 en:%4 sk:1
if not errorlevel 1 got end
echo !! Error encountered running FILEREDU
:end
```

3. Toolkit Utilities

3.1. PKTREDU

```
PKTREDU  infile:filename informat:format [outfile:filename] [outformat:format] [page:number] [start:number]
        [end:number] [every:number]
```

Extracts packets from a file and writes them to a new packet file, optionally converting the file format in the process from, say, raw to ASCII. This program is useful both in its own right and as a preprocessing tool for several of the other toolkit programs that take a packet file as input such as PKTTOF which can only accept packet data files in raw format.

It is commonly used to reduce the size of files by, for instance, only extracting every 10th packet. You can also select all packets within a start and end packet number range.

For text files, the toolkit program FILEREDU performs the equivalent function to PKTREDU.

Because PKTREDU can optionally perform format conversion, you must specify at least the input file format using option `informat`. The output format defaults to that of the input format if you do not supply the `outformat` option.

Example

To convert every 3rd packet from packets 10 to 20 in the TPS samples file 5.DAT, from raw format to compressed binary, and put the resulting data in 5.BIN, you would type the following:

```
PKTREDU infile:5.dat outfile:5.bin start:10 end:20 every:2 informat:raw
outformat:compressed page:c:\tpsdist\page0000.bin
```

In the example above, FTOPKT will interpret the packet according to the current TPS system settings as stored in PAGE0000.BIN in the TPS folder C:\TPSDIST.

Options

Option	Description	Default
infile	packet data file to be reduced	none
outfile	reduced copy of the input file	PKTREDU.DAT
start	first packet in the input file to be copied to the output file	1
end	last packet in the input file to be copied to the output file	last packet in the file, maximum 4000 million

every	only process every nth packet	maximum 4000 million
informat	Packet format of input file - raw, ASCII or compressed	none
outformat	Packet format of output file - raw, ASCII or compressed	informat

3.2. FTOPKT

FTOPKT infile:filename [outfile:filename] [page:number]

Converts a file to TPS 'raw-bin' packet format such that the file can be read by TPS as a packet data file. Since TPS can re-transmit data using the Transfer, Echo menu item, the replayed packet file can also be simultaneously transmitted on the TPS network. The PKTTOF program performs the reverse operation, i.e. converts a file of packets back to an ordinary file.

Use the page option if you wish FTOPKT to use a PAGE0000.BIN file located in another directory other than the current working directory.

Common uses for this utility are for transfer of files and viewing file contents in headecimal, i.e. replaying the file whilst viewing the hexadecimal packet display on TPS page 13 (256 byte packet dump).

Example

```
C:\TPSDIST\FTOPKT infile:myfile.txt outfile:myfile.dat
page:c:\tpsdist\page0000.bin
```

```
FTOPKT File-to-TPS standard Packet conversion tool.
Copyright (C) Micro SciTech Ltd. All rights reserved. 1991-1996
```

```
Writing 346 bytes...Bytes written = 384
```

```
Program terminating...
```

Where 346 is the total number of bytes read from the input file and 384 is the number of bytes written. The difference is due to packetisation overhead and explained in the below Technical Details. In the example above, FTOPKT will format the packets according to the current TPS system settings as stored in PAGE0000.BIN in the TPS folder C:\TSPDIST.

Options

Option	Description	Default
infile	file to be converted to data packets	none
page	page 0 file (TPS packet format system settings)	page0000.bin (current directory)

Technical details

All files are treated as binary, i.e., every byte in the file is read with no conversion of control codes. The reverse program PKTTOF treats files similarly so any file converted to packets using FTOPKT can be converted back to an identical copy using PKTTOF.

FTOPKT divides the file into data blocks of a fixed length as specified within TPS using the `Transfer, Pkt Format` menu item. This information is actually stored as a TPS 'System Setting' in the file `PAGE0000.BIN` and so a copy of `PAGE0000.BIN` must be resident in the directory in which FTOPKT was installed.

Since, it is rare that the file is exactly divisible into whole, fixed length packets, any unused space in the last packet is padded with the period character '.'. Every packet starts with the synchronisation pattern and ends with a two byte data length.

In the example given above, the numbers are derived as follows:

Extra bytes are allocated for packetisation overhead. 2 bytes per packet are allocated for the synchronisation pattern (TPS default hex 12 34) and an additional 2 bytes per packet reserved for storing the data length value. This latter 2-byte value denotes the actual length of file data stored in the packet itself is stored in the last two bytes of the packet, i.e. 127 and 128 when using the default 128 byte TPS packet length. With 4 bytes overhead per packet (2 sync., 2 data length), 124 bytes of file information is stored per packet. Using the default TPS packet length of 128 bytes, it can be seen that the file was split into three 128-byte packets ($384 = 3 \times 128$) of which the first two packets contained 124 bytes each, leaving 98 in the last packet ($346 = 2 \times 124 + 98$). Consequently, the last packet was padded with 26 period characters ($124 = 98 + 26$). These padding characters are automatically stripped off by PKTTOF - see the Technical Notes

Suggestions

We recommend you use the same [packet] format as most commonly used in your TPS system. This is because the formats obviously have to be compatible between receiving and sending TPS stations.

How to send the file

- Ensure the receiver uses the same packet format and serial protocols (baud, parity, etc)
- Inform the receiver to set up data recording in raw format to the filename desired

3.3. PKTTOF

PKTTOF *infile:filename* [*outfile:filename*]

Converts a TPS raw binary packet file to a file by stripping off all the TPS header information.

This utility can be used for file transfer and is usually used in conjunction with FTOPKT which performs the reverse operation, i.e. converts files to packets. For instance, a file can be converted to packets, transmitted across a TPS network to another PC where TPS is recording the data. The recorded file can then be stripped of its packet header to give an identical copy of the original file.

The input packet file must be in raw format (the file format generated by FTOPKT). PKTTOF cannot use compressed or ASCII. However, this is not a limitation since the utility PKTREDU can be used prior to PKTTOF to convert compressed or ASCII format files to raw format.

Use the page option if you wish FTOPKT to use a PAGE0000.BIN file located in another directory other than the current working directory.

Example

To convert a raw format packet file MYFILE.DAT to a file, MYFILE.TXT, without any packet header information, type the following:

```
C:\TPSDIST\PKTTOF infile:myfile.dat outfile:myfile.txt
page:c:\tpsdist\page0000.bin
```

In the example above, PKTTOF will format the packets according to the current TPS system settings as stored in PAGE0000.BIN in the TPS folder C:\TSPM4.

Options

Option	Description	Default
infile	The packet file to be converted	none
outfile	The converted packet file	PKTTOF.DAT
page	page 0 file (TPS packet format system settings)	PAGE0000.BIN (current directory)

Technical details

All files are treated as binary, i.e., every byte in the file is read with no conversion of control codes. The reverse program PKTTOF treats files similarly so any file converted to packets using FTOPKT can be converted back to an identical copy using PKTTOF.

3.4. COLTOPKT

COLTOPKT *infile:filename* [*outfile:filename*]

Converts a text file of numeric columns (tabular file) to packets such that the packets can be replayed by TPS for graphical display of the data. This enables TPS to be used as a simple plotting utility similar to a spreadsheet analysis program.

The files generated by the TPS parameter record facility are prime candidates for this application. In particular, the toolkit program PASPLICE can be used to splice several parameter record files together to form a single file of multiple columns, one column for each parameter. By converting this file to a packet file, each of the parameters can, for instance, be plotted.

Example

```
C:\TPSDIST\COLKTOPCK infile:colfile.txt outfile:colfile.dat
page:c:\tpsdist\page0000.bin
```

In the example above, PKTTOF will format the packets according to the current TPS system settings as stored in PAGE0000.BIN in the TPS folder C:\TSPDIST.

Options

Option	Description	Default
infile	The tabular file to be converted	none
outfile	The packet data file generated from the input file in TPS raw format	COLTOPKT.DAT
page	page 0 file (TPS packet format system settings)	page0000.bin (current directory)

Technical details

COLTOPKT builds one raw format packet per line in the input file. For each column, it generates a single precision (4-byte) floating point number and inserts this in the data packet. Thus, the Column 1 value will occupy the first four bytes, column 2 will occupy the next four bytes etc.

Note, the files must contain numerical data only.

3.5. PASPLICE

PASPLICE *infile:filename* [*outfile:filename*]

Splices together two TPS parameter record files to form a single tabular file. By repeating the operation, several parameter record files can be merged. Such a single tabular file is useful as a mission summary and is in a form that can be imported into spreadsheets and database programs for detailed analysis. Using the toolkit COLTOPKT utility it can also be converted to packets for analysis by TPS using the TPS file playback facility.

Example

```
C:\TPSDIST\pasplice infile1:p11.12 infile2:p11.13 outfile:summary.txt
```

Options

Option	Description	Default
infile1	The first parameter record file to be spliced with the second input file	none
infile2	The second parameter record file to be spliced with the first input file	none
outfile	The spliced output formed from the two input files	0

Technical details

Each parameter record file has a first column containing the time (by default - page 0)

Asterisks where no valid value exists in one file

3.6. EXPERR

EXPERR [infile:*filename* outfile:*filename*]

Descriptively expands the numerical error codes in the TPS error file (usually TPSERR.TXT) with a brief text message as seen in the error window when running TPS. It is also useful to run the converted error file through the toolkit utility EXPTIME to expand the times given in column 1 of the file.

Example

```
EXPERR outfile:tpserr.new
```

Expands the error codes in the TPS error file TPSXERR.TXT and puts the descriptive information in the file TPSERR.NEW.

Options

Option	Description	Default
infile	TPS error file	TPSERR.TXT
outfile	Descriptive version of the input error file	EXPERR.TXT

3.7. EXPTIME

EXPTIME infile:filename [outfile:filename]

Expands the TPS elapsed millisecond (the TPS default 'tag' parameter) time into an absolute date and time. The input file is either a parameter record file or a TPS error file which has, as its first column, the 'tag' parameter value, the [valueport] for which is specified on page 0. The tag parameter, unless changed by the user, is the TPS elapsed time in milliseconds since the start of the TPS session, i.e., when TPS was last loaded. However, to the user, this is a rather meaningless number and is far more readable when converted to an absolute format.

Example

EXPTIME infile:TPSERR.TXT outfile:tpserr.new

Options

Option	Description	Default
infile	TPS error file or parameter record file	none
outfile	Expanded version of the input file	EXPTIME.TXT

3.8. FILEREDU

FILEREDU *infile:filename [outfile:filename outformat:format start:number end:number every:number]*

Selectively extracts lines from a text file and writes them to a new file. For instance, parameter record files generated by TPS can grow very large and, before passing to the toolkit splice utility PASPLICE for merging with other recorded files, it is useful to reduce them with this FILEREDU utility to a readable file size, say, < 500 lines.

Example

```
FILEREDU infile:p1l.15 outfile:p1l15.new every:4
```

Reads every line of the parameter record file P1L.15 (page 1, valueport 15), and generates a new file comprising every fourth line of the input file.

Options

Option	Description	Default
infile	The text file to be reduced	none
outfile	The reduced copy of the input	FILEREDU.TXT
start	The first line in the input file to be copied to the output file	1
end	The last line in the input file to be copied to the output file	the last line in the file, maximum 4000 million lines
every	only process every nth line	maximum 4000 million

Technical details

All files are treated as text, i.e., they can be read with a standard text editor such as Windows Notepad.

4. Syntax Summary

TPS File Processing Utilities

PKTREDU *infile:filename* [*outfile:filename*] [*page:number*] [*informat:format* *outformat:format*] [*start:number*]
[*end:number*] [*every:number*] [*page:number*]
FTOPKT *infile:filename* [*outfile:filename*] *page:number*
PKTTOF *infile:filename* [*outfile:filename*] *page:number*
COLTOPCK *infile:filename* [*outfile:filename*] [*page:number*]
PASPLICE *infile1:filename* *infile2:filename* [*outfile:filename*]
EXPTIME *infile:filename* [*outfile:filename*]
EXPERR [*infile:filename*] [*outfile:filename*]
FILEREDU *infile:filename* [*outfile:filename*] [*start:number*] [*end:number*] [*every:number*]