



T2P.EXE

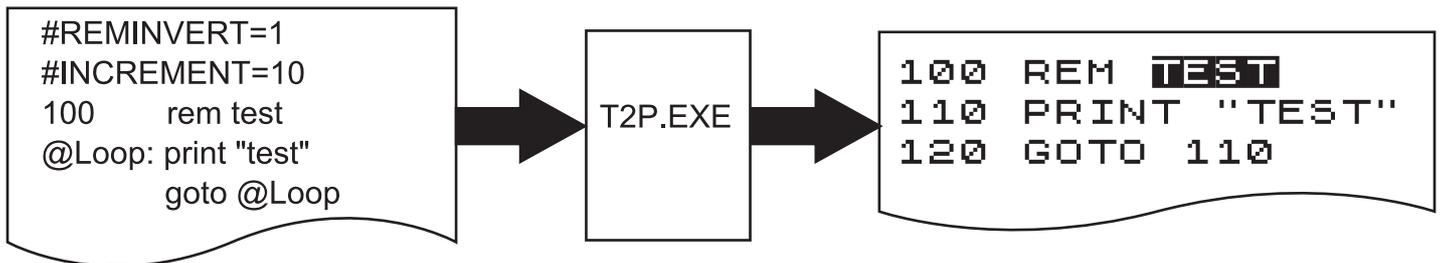
Version 1.7

for Windows



T2P ist ein Programm, das aus einer Textdatei eine lauffähige P-Datei für den ZX81 oder TS100 erzeugt. Damit kann ein Basic Programm am PC mit einem beliebigen Texteditor geschrieben werden. Die erzeugte P-Datei kann entweder auf einen ZX81 übertragen und geladen werden oder in einem Emulator wie z.B. EightyOne benutzt werden.

T2P is a program that generates a valid P-file for the ZX81 or TS1000 from a textfile. You can now write Basic programs on your PC with any texteditor. The resulting P-file can either be transferred to a ZX81 or be used in an emulator like EightyOne.



Eigenschaften

Das Programm entstand aus dem Programm ZXTXT2P.EXE, aus dem Jahr 2002. Es ist dazu weitgehend kompatibel. Folgende Eigenschaften sind ergänzt worden:

- Quelltext als Batchdatei incl. t2p.exe Aufruf möglich
- Basic-Zeilen mit oder ohne Zeilennummern
- automatischer Increment der Zeilennummern
- Increment bereichsweise einstellbar
- feste Zeilennummern vorgebar
- Label und Zeilennummern gleichzeitig
- Labelreferenz als Zeilennummer oder Speicheradresse
- inverse REM Zeilen
- Autostart aktivierbar
- FAST voreinstellbar
- Variablen vordefinierbar (auch Arrays)
- Grafikzeichen, inverse Zeichen und binäre Zeichen

Aufrufparameter

Das Programm hat die folgenden Aufrufparameter:

t2p [-v] [-a] [-r] [-d] [-o Dateiname] Dateiname

- v Verbose
- a Autostart
- r REM invertiert
- d DFILE minimiert
- o anderer Ausgabe-Dateiname

Wenn kein Ausgabe-Dateinamen angegeben wurde, dann wird passend zu der Quelldatei eine Datei mit der Erweiterung ".p" für die Ausgabe erzeugt.

Wenn die Quelldatei vom Typ ".bat" ist, dann werdsn bis einschließlich der Zeile "exit" alle Zeilen in der Quelldatei überlesen.

Die Parameter "-L" und "-I" werden nicht benötigt, weil Label-Mode automatisch aktiv ist und der Increment per Pseudobefehl bestimmt werden kann.

Features

This program is a successor of ZXTXT2P.EXE from 2002. It is largely compatible in usage.

The following features are added:

- Basic-text can be written as Batchfile calling t2p.exe
- Basic lines with or without linenumbers
- automatic increment of linenumbers
- increment value adjustable at any line
- fixed linenumbers also
- labels and linenumbers at same line
- reference to labels as linenummer or as memory-address
- inverse REM lines
- autostart possible
- FAST presetable
- predefined variables (even arrays)
- graphics, inverse and binary characters supported

Calling parameter

You can call the program with the following parameters:

t2p [-v] [-a] [-r] [-d] [-o filename] filename

- v verbose
- a autostart
- r REM inverse
- d DFILE collapsed
- o different output filename

If there is no output filename given, then it will be made like the sourcefile but with extension ".p".

If the sourcefile has extension ".bat", then all lines until after "exit" will be discarded.

Parameters "-L" and "-I" are not needed because Label-mode is automatically enabled and increment can be adjusted with pseudo-commands.

Kommentare und Pseudobefehle

Kommentarzeilen müssen mit "#" beginnen. Sie werden nicht in das Basic-Programm übernommen.

Eine Besonderheit stellen die Pseudobefehle dar. Sie sind Kommentarzeilen, die mit einer der hier abgebildeten Zeichenfolgen beginnen.

(Großbuchstaben beachten!)

#INCREMENT=, #VERBOSE= und #REMINVERT= können im Quelltext auch mehrfach

benutzt werden. Dabei ist der neue Wert ab der jeweiligen Zeile wirksam.

Die mit #VARS erzeugten Variablen entsprechen der Syntax eines LET Befehls. Bei Arrays müssen hier in Klammern die Dimensionen angegeben werden. Nach dem Gleichheitszeichen folgt dann eine Liste der Elemente, die jeweils mit Komma getrennt sind.

```
#AUTOSTART= 0 or 1 (default=0)
#INCREMENT= Number (default=10)
#REMINVERT= 0 or 1 (default=0)
#FAST=      0 or 1 (default=0)
#VERBOSE=   0 or 1 (default=0)
#VARS n=1
#VARS number=2
#VARS u$="123"
#VARS a(2,3)=11,12,13,21,22,23
#VARS a$(2,3,4)="abcd","efgh","ijkl","0123","4567","8901"
```

Comments and pseudo-instructions

Comments have to start with a "#". These lines will not be copied to the program.

Pseudo-instructions are special comments. They all have to start with the listed phrases. (Only uppercase!)

#INCREMENT=, #VERBOSE= and #REMINVERT= can be used multiple in a source-text. Each new value is then valid for the following lines.

With #VARS you can create variables similar to a LET.

When used for an array you have to define the dimensions in brackets. After the equal sign all the values have to follow separated by comma.

Zeilennummern und Label

Zeilennummern können an den Anfang jeder Programmzeile eingefügt werden. Wenn eine Programmzeile keine Zeilennummer hat, dann wird bei der Übersetzung eine Zeilennummer automatisch generiert, die um den Wert größer als die vorangehende Zeile ist, der mit dem letzten vorausgehenden #INCREMENT= eingestellt wurde. Damit lassen sich feststehende Einsprung-Zeilennummern realisieren und gleichzeitig ein bequemes Einfügen von Basic-Zeilen.

Zeilennummern dürfen auch in einer eigenen Zeile weit vor der jeweiligen Programmzeile stehen.

Eine Zeilennummer darf allerdings nie kleiner als die Zeilennummer der vorangehenden Basic-Zeile sein.

Zeilennummern dürfen mit 0 beginnen und einen maximalen Wert von 16383 (hex 3FFF) besitzen.

Ein Label muß vor einer Basic-Zeile stehen. Es kann in einer eigenen Zeile oder in derselben Zeile des Basic-Befehls stehen. Es kann mit oder ohne Zeilennummer zusammen stehen. Es kann vor oder hinter der Zeilennummer stehen.

Eine Labeldefinition beginnt mit dem Zeichen "@" gefolgt von dem Namen des Labels und wird mit dem Zeichen ":" abgeschlossen.

Eine Referenz auf ein Label kann mit seiner Zeilennummer oder der Speicheradresse der Zeile geschehen. Für die Zeilennummer wird das Zeichen "@" mit dem Labelnamen benutzt. Für die Zeilenadresse wird das Zeichen "&" mit dem Labelnamen benutzt. In beiden Fällen darf hier hinter dem Labelnamen kein Doppelpunkt stehen.

Der Inhalt einer REM-Zeile könne z.B. so ausgelesen werden:

```
print "remline ";@RemLine

for x = &RemLine +5 to &RemLine +8
  print peek x
next x

@RemLine: rem \01\39\30\c9
```

Linenumbers and labels

Linenumbers can be entered at the beginning of a Basic-line. If a Basic-line has no line number then it will be generated automatically at translation. This line number will be that much bigger than the previous one as defined by a preceding #INCREMENT=. With this you can get lines to have a fixed line number and have lots of comfort in adding lines.

Line numbers may be written in separate lines much ahead of the lines they belong to.

A linnumber must not be smaller than the preceding linnumber.

Linenumbers may start with 0 and may rise up to 16383 (hex 3FFF).

A label has to be written before a Basic-line it refers to. It may be combined with a linnumber. It then may be written before or after the linnumber.

A label definition starts with a "@" followed by the name of the label and ends with a ":".

A reference to a label can result in its linnumber or its memory address.

If the linnumber is referenced then the character "@" is used before the name. If the address is referenced then the character "&" has to be used. In either way there must not be a colon after the name.

The following example shows such a reference in both ways.

Aufruf mit Batchdatei

Um den Aufruf von T2P.EXE und das Basic-Quellprogramm zusammen in einer Batch-Datei zu schreiben, sollte man so vorgehen.

Die Textdatei mit dem Quell-Programm (hier nur die Zeilen 100 und 110) in eine ".bat" Datei umbenennen. Die im Beispiel angegebenen Zeilen davor einfügen.

Hinweise:

"@echo off" ist nur dazu da, dass der Batch weniger Text in das Textfenster schreibt. Dieses Textfenster ist normalerweise auch schnell wieder verschwunden.

"t2p %0" ist die Zeile, die T2P.EXE aufruft und mit "%0" seinen eigenen Dateinamen mit Extension als Aufrufparameter, also als Quelldatei, mit angibt. Da die Extension ".bat" ist, wird T2P bis zur Zeile "exit" alles überlesen und dann mit den beiden Basic-Zeilen beginnen.

"if errorlevel..." wird immer dann ausgeführt, wenn T2P einen Fehler gemeldet hat. Die beiden Befehle "...pause & goto END" lassen das Textfenster auf einen Tastendruck warten und danach mit dem Sprung zu END und dem folgenden "exit" das Textfenster schließen. Dadurch hat man Zeit, sich die Fehlermeldung im Textfenster anzusehen.

Wenn T2P fehlerfrei lief, dann wird die Zeile "start..." ausgeführt. Durch diesen Befehl wird der Befehl "...%~n0.p" aufgerufen ohne auf dessen Ende abzuwarten. Das Fenster schließt sich also sofort wieder.

Mit "~n" wird aus dem Name der eigenen Batchdatei "%0" nur der vordere Teil ohne Extension ausgefiltert. Die Extension wird mit den Zeichen ".p" wieder ergänzt. Hiermit wird also die gerade erstellte P-Datei aufgerufen. Hier ist es empfehlenswert, wenn man z.B. P-Dateien mit EightyOne verknüpft hat. Damit wird dann nämlich EightyOne automatisch mit der neuen P-Datei gestartet.

Zusammenfassend haben wir damit eine Datei erzeugt, die sich selber übersetzen wird und dann mit dieser Datei eine Aktion startet (hier EightyOne). Diese Aktion kann natürlich auch etwas anderes sein, wie z.B. ein Copy-Befehl.

Weitere Informationen

Weitere Informationen, Beispiele und Aktualisierungen...

Calling from a batchfile

To write a call to T2P.EXE and your Basic-Source in one batch-file follow the following guidelines.

```
@echo off
t2p %0
if errorlevel 1 pause & goto END
start %~n0.p
:END
exit

100 print "hello world"
110 run
```

Rename the file containing the Source-program to become a ".bat" file. In this example with program lines 100 and 110 just enter the lines above.

Hints:

"@echo off" helps to minimize the text appearing in the text-window of the batch. At normal situations this window will close immediately.

"t2p %0" is the command that calls T2P.EXE with "%0" as parameter. This is the name of your batchfile itself. T2P will recognize its extension as ".bat" and therefore will overread all lines until after "exit".

"if errorlevel..." will handle the case that T2P reports an error. The following commands "...pause & goto END" makes the batch to wait for a keyboard entry and after this jump to the line with "exit". With this you get the option to read the error messages before the text window closes.

If T2P found no error then the line containing "start..." will be executed. By this the following command "...%~n0.p" starts executing but the batch continues without waiting for the result. So the text window will close instantly.

By the filter "~n" the extension will be cut from the name of the batch. The new extension ".p" is therefore added. So the new generated P-file will be opened. If P.files are associated with EightyOne, it will start EightyOne then.

Now you got a file that starts its own translation followed by an action like EightyOne. You can easily change its behavior to start something different like a Copy command.

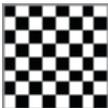
Further reading

More infos, examples and new versions...

<http://www.swatosch.de/zx81>

Sonderzeichen

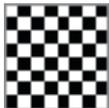
Die folgenden Grafikzeichen lassen sich mit "\" eingeben

"\ " "	
"\ ' " "	
"\ ' ' " "	
"\ ' ' ' " "	
"\ . " "	
"\ : " "	
"\ . ' " "	
"\ : ' " "	
"\ # # " "	
"\ , , " "	
"\ ~ ~ " "	

Das Doppel-Anführungszeichen wird mit "\" eingeben.
 Am Ende einer Zeile kann mit "\" das Weiterschreiben in der Folgezeile erzwungen werden.
 Beliebige Zahlenwerte können in hexadezimaler Form mit "\00", "\01", "\02", ... "\FD", "\FE", "\FF" eingegeben werden.
 Damit können alle Zeichen oder auch Token eingegeben werden.

Special characters

The following blockgraphic characters can be entered using "\"

"\ : : " "	
"\ . : " "	
"\ : . " "	
"\ . . " "	
"\ ' : " "	
"\ : " "	
"\ ' . " "	
"\ . " "	
"\ @ @ " "	
"\ ; ; " "	
"\ ! ! " "	

The double quotation mark can be entered using "\".
 At each end of line a continuation in the following line can be forced with "\".
 Any Value can be entered with hexadecimal notation like "\00", "\01", "\02", ... "\FD", "\FE", "\FF"
 In this way any character or even token can be entered.

Beispielprogramm

Exampleprogram

```
@echo off
t2p %0
if errorlevel 1 pause & goto END
start %~n0.p
:END
exit

#AUTOSTART=1
#REMINVERT=1
#INCREMENT=20

#VARS a$="hallo"
#VARS b=55
#VARS ab=4711
#VARS c(2,3)= 0,1,2,3,4,5
#VARS d$(2,3,4)="1234","0011","abcd","9876","1100","wxyz"

1000
    rem test
@Loop:
    print a$
    print b
    print ab

2000
    for x=1 to 2
        for y=1 to 3
            print c(x,y);".";
        next y
    print
    next x

3000
    for x=1 to 2
        for y=1 to 3
            print "\"";d$(x,y); "\"";
        next y
    print
    next x

goto @Loop
```