EXEC - Command-List Processor. (Size 1000 bytes).

EXEC allows you to execute "programs" which are lists of TRSDOS commands and/or direct BASIC statements. It is particularly useful for preparing power-up command sequences and for organising files. A command list is prepared as a "BASIC" program, with one command per line, and then saved as a file with the ASCII option.

Refer to the enclosed sheet, RELOC, for instructions on how to load the tape supplied. Instead of locating EXEC in protected memory, you can load EXEC anywhere above X'7000' (decimal 28672). If you choose X'7001' then EXEC will not collide with DOS utilities, which run below this address. And you will be able bring up BASIC without difficulty. However, any large BASIC program will overwrite X'7001', so use the trick shown in Example 2 to LOAD or RUN a large BASIC program, (or locate EXEC in protected memory instead). After relocating EXEC to X'7001' (28673) under Level2, return to TRSDOS and dump it as an executable file, on disk, e.g.

DUMP EXEC/CMD (START=X'7001',END=X'73E9',TRA=X'7001')

Example 1. Bring up BASIC.

Under BASIC prepare the following program:

```
10 CLOCK (ON)              (typical DOS command)
20 BASIC                   (invoke BASIC)
30 2                       (number of file buffers)
40 48128                   (protect memory, if necessary)
50 DEFUSR=&HBC00           (typical set-up statement)

SAVE "BRINGUP",A           (save with the ASCII option)
CMD"S"                     (return to TRSDOS)
EXEC BRINGUP               (invoke "BRINGUP" command list)
```

This will now execute the commands/statements in this file, using 2 and 48128 as responses to the BASIC command. (Under NEWDOS these are supplied instead as command parameters).

1) If you type AUTO EXEC BRINGUP under DOS, then BRINGUP will be executed automatically at power-up.
2) It is a quirk (a bug?) of BASIC that you cannot reload an ASCII file with lines starting with digits.
3) DOS commands must be on separate lines. BASIC statements need not be, and may contain REMARKs etc.

Example 2. Same Program, using Parameters.

```
10 CLOCK (ON)
20 BASIC
30 @1<2>                        i.e. substitute param1, or "2" if no param1
40 @9<48128>                    dummy parameter to avoid starting line with a digit.
50 @9<10> XX$="@2":IF XX$<>"" THEN RUN XX$   build temporary program to avoid RUN XX$ overwriting active EXEC
60 RUN                          EXEC releases control on this RUN
```

1) @ is a "reserved" symbol. It must be followed by a single digit, 1 to 9, representing the parameter number.
2) If followed immediately by <string>, then string is used as the default value for that parameter.
3) On invocation the parameters follow the invoked filename, in order, separated by a single blank or comma.

```
EXEC BRINGUP 4,SORT   calls BRINGUP passing param1=4 and param2=SORT. <2> is ignored. The program SORT is RUN.
EXEC BRINGUP,4 SORT   means exactly the same thing (blanks and commas are interchangeable).
EXEC BRINGUP,,SORT    would pass param1=null, and the default, 2, would be used. SORT would be RUN.
EXEC BRINGUP          would pass both parameters as null, and no program would be invoked.
```

Example 3. File Organisation.

A typical repetitive operation is to create a new copy of a file, preserving the old copy as a back-up.

```
10 KILL OLD@1/CIM
20 RENAME @1/CIM TO OLD@1/CIM
30 DUMP @1/CIM (START=X'C000',END=X'C300',TRA=X'C000')
```

EXEC THISFILE IPL is now a convenient shorthand to create a new file called IPL/CIM, saving the old version.

1) Errors, e.g. "PROGRAM NOT FOUND", do not terminate processing. So guard against irreversible losses.
2) DIR and LIST cannot satisfactorily be used in a command list, because they need indeterminate prompts.

Exmple 4. Invoking a BASIC Program Compiled by Southern Software's ACCEL2.

Suppose you have developed a BASIC application which you have compiled under ACCEL2, and which you intend to sell. You want to minimise the operational overhead required to bring up and invoke your program. You prepare a disk containing your compiled application and the run-time component of ACCEL2, located at 47616 to 49151, say. This component is called LOADER/CIM, and your application is called REPORT/BAS. Your command list, called REPORT, invoked by EXEC REPORT, will be:

```
10 LOAD LOADER/CIM                (load the run-time component of ACCEL2)
20 BASIC
30 @9<2>
40 @9<47616>                      (protect ACCEL2)
50 SYSTEM
60 /47616                         (activate the run-time routines)
70 @9<10> /RUN "REPORT/BAS"       (run the compiled application)
80 RUN
```

Turning a Command List into a "Command".

Each command list described has been a small ASCII file, interpreted by the EXEC routine. Typically EXEC CLIST has to load 2 files, EXEC itself, and CLIST. Furthermore, in the ACCEL2 example, the sale disk would have to contain copies of the CLIST and Southern Software's EXEC processor. But provided your command list occupies only one record on disk (i.e. it is less than 256 bytes), then after EXEC has run, a copy of this single record will reside in EXEC's internal 256-byte buffer. If you reinvoke EXEC at normal start + 2 bytes, then EXEC will re-execute this internal buffer, without reloading the CLIST file. So if you DUMP bytes 2 to 1000 of EXEC, under a new name, then you will have a single self-contained file which will always execute the same set of commands. (You can still pass parameters, as before).

Under TRSDOS type

```
EXEC REPORT        (Use BREAK to interrupt in BASIC before EXEC is overwritten)
CMD"S"
DUMP REPORT/CMD (START=X'7003',END=X'73E9',TRA=X'7003')
```

Now you have a single file which runs faster, occupies less disk space, and which can be invoked simply by typing REPORT. In addition you are shipping a copy of EXEC which will only run REPORT, not a general command list, so copyright infringement will be overlooked by Southern Software. (You should however give an acknowledgement to Southern Software in your product documentation.)