Subject: Free TRS-80 Programs! (Documentation - LONG)
Posted by ronbe on Mon, 03 Jun 2013 03:16:28 GMT
View Forum Message <> Reply to Message

Message-ID:
Date: Thu, 21-Feb-85 16:49:21 EST
Article-I.D.: tekred.252
Posted: Thu Feb 21 16:49:21 1985
Date-Received: Sat, 23-Feb-85 02:45:59 EST
Organization: Tektronix, Redmond OR
Lines: 546
Xref: utcs net.micro:9055 net.micro.trs-80:267

Here's some documentation for programs I've written for the TRS-80. If you don't have a TRS-80, stop now!!!

I'm willing to make any or all of these programs public domain and mail the source or object code (in Intel Hex) to anybody who wants them. Also, if you do want these programs, I'll USMail you the TERM and TRANSLAT programs for the price of the cassette or disk and postage. Or send me the disk with a mailer and postage.

Ron Bemis
659 SW 29th St
Redmond, OR 97756
(503) 923-0580 (h)
(503) 923-4460 (w)
tektronix!tekred!ronbe

Following is a short description of each of the 8 programs:

----TERM
This program turns the computer into a terminal with uploading and downloading capabilities.

----TRANSLAT
This program translates Intel Hex to and from TRS80 executable format.

----DBUG
Full function monitor/debugger program for Z80 code. Includes disassembled next instruction.

----DISASM
This program disassembles either memory or loadable files into re-assemblable Z80 code.

----RECONFIG
 This program tailors DISASM for your assembler and your own
needs.

----OUTLINE
 This program displays the addresses in memory that will be
loaded when a given program executes.

----MERGE
 This program merges two or more executable programs together to
form another, larger executable program.

----EXTRACT
 This program creates a loadable program from a larger
executable program.

SMART TERMINAL PROGRAM

 This program turns the computer into a terminal with uploading
and downloading capabilities.

```
Syntax: TERM  to login
 TERM *  to continue
```

 Upon execution, TERM will try to find and open a file called
LOGIN/BLD. If it finds it, it will read in one record of 256 bytes
and use this as a login procedure. The format for the LOGIN/BLD file
is lines beginning with "S" (for send) or "R" (for receive), followed
by data, and ending with a carriage return. This file can easily be
created by typing BUILD LOGIN at the DOS READY line. The data in a
"S" line will be sent including the carriage return. The data in a
"R" line will be waited for, excluding the carriage return.

Example:

```
 S   sends carriage return
 R;login (dialup):
 Sdialup
 RPassword:
 Sdialup
 Rlogin:
 Smylogin
 RPassword:
 Smypass
```

 If you exit the program while still connected to the host,
you can type TERM * to re-start it without attempting to login again.

If the login fails, press BREAK to begin manual login procedure.

While connected, pressing CLEAR puts you into control mode. Pressing certain keys at this time allows you to send special characters to the host (characters not on the TRS80 keyboard). Pressing CLEAR a second time puts you into command mode.

Control Mode (entered by pressing CLEAR):
Pressing: < > : ( ) 6 ! / ' -   CLEAR
Gives: [ ] ~ { } ^ | \ ` _ command mode

Example:
To send "~", press CLEAR, then ":".

Command Mode (entered by pressing CLEAR twice):
C - Clear the RAM Buffer
P - Printer ON/OFF Switch
D - Download a File on TRS80 Disk
W - Write the Buffer's Contents to TRS80 Disk
Q - Quit TERM - Return to DOS
CLEAR - Return to Interaction

Example:
To clear the buffer, press CLEAR (control mode), CLEAR (command mode), "C" (to clear), and CLEAR again (to return).

The buffer is for storage of data as it arrives from the host. At start-up, the buffer is OFF (nothing stored as it comes from the host). Pressing BREAK will turn the buffer ON (everything coming from the host is buffered). Pressing BREAK again will turn the buffer back OFF. This is a circular buffer, in other words, if there's no more room at the end of the buffer, it fills in at the beginning. However, it will not over-write good data already stored. If the buffer becomes full, a message will appear and buffering will be turned OFF.

The buffer is for storing information for saving or printing. Pressing "P" from command mode will turn the printer on, and use the buffer for a print spooler (everything coming from the host will be printed when buffering is ON). In this case, writing to the buffer is just like printing. The only difference is that the spooler sends data to the printer only when the printer is ready for it. Pressing "P" from command mode a second time will turn the printer off. If buffering is on now, the data in the buffer can be used to write a file to the TRS80 disk. Press "W" from command mode, enter a filename, and the buffer will be written out to disk. Using "W" will also clear the buffer if it completes successfully.

To download a file, first get the host ready to accept the

data. Under UNIX, type something like "cat >> file" and return. Now
enter command mode, and press "D", and enter the filename. If the
file is found, you will be returned to command mode and the file will
be sent. At the end of sending, an EOT (cntl-D) will be sent and the
printer will beep. Downloading is done line-by-line, that is, an
entire line is sent with carriage return, then the echoed carriage
return is awaited before sending the next line. For this reason, you
may not see everything that is sent to the host. This method is
much faster that waiting for each individual character to echo. If
the download fails, press BREAK to abort.

X-off is sent automatically to the host when you enter command
mode. X-on is sent when you leave command mode.

TERM handles the display of 4 characters differently than the
TRS80. CR (return) moves the cursor to the left side of the current
line. LF (line feed) moves the cursor down one line and leaves it in
the same horizontal position. This is really just standardizing these
two characters. The TRS80 handles them differently than most computers.
Also, the TRS80 doesn't know how to handle HT (tabs), so TERM fixes
that. The TRS80 is not capable of producing a bell, so if a BEL is
received, TERM sends it directly to the printer, causing the printer to
make the noise if possible.

TRANSLATOR

This program translates Intel Hex to and from TRS80 executable
format.

Syntax: TRANSLAT

/CMD suffix assumed for TRS80 executable file
/HEX suffix assumed for Intel Hex file

Upon execution, a small menu will appear asking the direction
of the conversion. Press 1 for Intel Hex to TRS80 executable or 2 for
TRS80 executable to Intel Hex.

The program will then prompt you for the two file names.
"/CMD" and "/HEX" need not be typed. If the destination file exists, a
message to that effect will appear, and you will be asked if you want
to overwrite the file. Press "Y" or "N".

If the translation is successful,"Translation complete." will
appear. If there are any errors, they will be displayed and the process
will be aborted.

DEBUGGER

Full function monitor/debugger program for Z80 code. Includes disassembled next instruction.

Syntax: DBUG file params
Examples: DBUG
  DBUG TESTPROG
  DBUG EDITOR:1 INPUT OUTPUT

/CMD suffix assumed for file.

The first example above simply loads and starts the debugger.
All register values are stored as DEFS's, so they can be restored from a previous DBUG session if memory hasn't been altered in DBUG's area.

DBUG uses memory from E000-FFFF (hex), so it is not possible to DBUG programs that use memory in this range. (OUTLINE can show where programs load, but the program may not even reference this area - not even the stack!)

DBUG will allow up to 8 breakpoints to be set. Breakpoints are not allowed in ROM, and it is up to the user to not set breakpoints in DBUG memory locations (E000-FFFF). Breakpoints are done with a RST 8 instruction which is put into the memory at specified breakpoints only at JUMP time. CALL sets no breakpoints.

DBUG also has the ability to move, fill, and search memory.
Again, caution must be used in the E000-FFFF range. Also, do not try to execute the debugger with itself. (i.e. no PC values between E000 and FFFF.)

Following is a summary of DBUG commands:

B breakpoint -- set or clear a breakpoint
If the entered breakpoint is already set, it will be cleared.

C call -- call a subroutine
This should be used with extra caution. If the statement to be executed next is not a CALL, this will have the same effect as I (instruction trace). However, if a CALL is about to be executed, DBUG simply does the call and then returns to the user. For this reason, you should not use this command if the subroutine does not return normally or takes advantage of data at its return address. (i.e. checking the contents of SP.)

D display data -- set screen display pointer
Any locations in the addressable 64K may be displayed.

E edit memory -- similar to M (modify) with RS's DEBUG
 Enter the address to edit, then use the 4 arrows to move the cursor and change data. Data may be entered in hex or ASCII; pressing SHIFT and the UP ARROW together will toggle this function. Pressing ENTER will save your changes and return to main display, BREAK will return to main display without saving changes.

F fill memory -- fill range of memory with certain value
 Enter start and end addresses, then data value to fill this range with. Again, avoid E000-FFFF.

I instruction trace single-step -- execute one instruction
 This will also allow you to single-step through ROM, although this is not advised, as you could cause timing problems. If you find yourself in ROM, set a breakpoint you're sure you will hit and JUMP. The processor accepting interrupts will cause the debugger to lose control if the handlers do not return to the point they were called from.

J jump -- real execution at the current PC
 Debugger has no control until a breakpoint is hit (RST 8). Breakpoints are put in memory only on this command, and restored only after one is encounted and control returns to DBUG. You must press ENTER after this command.

K kill all breakpoints -- simply clears all 8 breakpoints

L load file -- load a file into memory
 Enter the file name with optional parameters. (Like command line, see Syntax, above.) Display pointer is set to the first location loaded, PC register is set to the entry address, and HL register is set to point to the parameters. (In other words, use K, then L, then J to execute a program.) Be careful the program you're loading doesn't overlap DBUG, or use any of its memory.

M move memory -- move consecutive memory to another place
 Source and destination may overlap, the move determines whether to use a forward (LDIR) or backward (LDDR) move. Again, avoid E000-FFFF.

N next search -- find the next occurance
 This command should be used only after a S (search) command. It is used to find a second possible occurance of a search string.

P port access -- read or write the I/O system

This command allows you to communicate with the I/O system. Reads and writes can both be done. Port Write to port EC (hex) with data 10 (hex) may be neccessary to use ports 00-7F. (Enable I/O bus.)

Q quit DBUG -- terminate the debugger
This command is used to terminated the debugger. You must press Y or ENTER to complete the command. N or BREAK will bring you back to the main display.

R register change -- change register contents
Enter the register name and the new data for the register. Primes, 8-bit, and 16-bit registers may all be modified. The only registers that may not be changed are the I and R control registers. Register names are entered by typing the name, then pressing ENTER or SPACE or COMMA. The exception to this is if the register name has three characters (i.e. AF'), then the name is automatically entered when the 3rd character is typed.

S search memory -- find a pattern in memory
Enter search start and end addresses, and then the data format (H for hex, A for ASCII). Enter the data in the format specified and press ENTER. When the string is found, it will be located at the first location of the data display. If not found, a message will inform you of such. To find a second occurance of the same string, see the N (next) command.

W write disk file -- similar to DOS's DUMP instruction
A disk file may be created starting anywhere in memory, and comprising any number of 256-byte blocks. Entry address is set to DOS (402D hex).

X execute -- execute a DOS command
Any of the standard DOS commands may be executed from within the debugger. Other programs may be executed, but will not return control to DBUG.

RIGHT ARROW -- increment PC to next instruction
The PC will move to the next instruction (skip current instruction).

LEFT ARROW -- decrement PC
The PC will decrement by 1.

UP ARROW -- move display pointer back
Display previous page (back 80 hex).

DOWN ARROW -- move display pointer forward

Display next page (forward 80 hex).

+ or ; -- move display pointer forward
Display next line (forward 10 hex).

- -- move display pointer back
Display previous line (back 10 hex).

DBUG error messages are displayed on the last line of the screen with a hex error number. This number must be converted to decimal (DD) to do X, then ERROR DD command.

Some commands display a message on the last line of the screen upon completion. You can execute commands at this time, or press any other key (such as SPACE or CLEAR) to return to the COMMAND--> prompt. Also, pressing SPACE or CLEAR or any other unknown key will re-display the entire screen, so internal memory changes such as the real-time clock interrupt locations may be observed as they change.

Holding down a shift key as you enter a command will cause DBUG to wait until the shift key is released before going back to the main display. This is useful for checking routines that put messages on the screen or Loads that load anything onto the screen.

2- and 4-digit hexidecimal numbers are entered into DBUG by typing the significant digit(s), then pressing SPACE or ENTER. The exception to this is when both or all 4 digits are typed, in which case the number is automatically entered when you type the last digit.

Pressing BREAK at any time DBUG has control will abort the current command and return to the main display with the COMMAND--> prompt.

DISASSEMBLER

This program disassembles either memory or loadable files into re-assemblable Z80 code.

```
Syntax: DISASM org(,end) (SRCfile) for memory
 DISASM CMDfile (SRCfile) for files
Examples: DISASM 9BC0,0A000
  DISASM 0,3FF ROM1K:1
  DISASM GAME
  DISASM GAME GAME
```

/CMD suffix assumed for CMDfile.
/SRC suffix assumed for SRCfile.

The SRCfile created can be edited and re-assembled by your assembler. (See RECONFIG to configure DISASM to your own needs). If your assembler loads the entire source file into RAM to assemble, you may need to break the CMD file down into smaller pieces before disassembling. (See EXTRACT.) Memory start and end addresses must be preceeded by 0 if beginning with a letter (A-F).

As the program executes, you can press BREAK to exit the program. Pressing @ will freeze the display. Press any key but BREAK to re-start.

Note: for printer output use the DOS's DUAL command before invoking the disassembler.

RECONFIGURING THE DISASSEMBLER

This program tailors DISASM for your assembler and your own needs.

Syntax: RECONFIG

The screen will fill with a selection menu, using the graphics hand as a pointer. To move the pointer to the next selection, press the SPACE bar. Following is an explanation of each selection:

Line Number Format
 0 None -- no line numbers will be created.
 1 ASCII -- line numbers will be put on the disk in ASCII format.
 2 EDTASM -- line numbers will be put on the disk in EDTASM
  format (BX). If this option is used, the line numbers
  will still be displayed on the screen in ASCII.

Labels
 0 No -- labels will not be created.
 1 Yes -- labels will be created with the form of LXXXX, where
  XXXX is the current PC. This is helpful in locating
  JP, CALL, JR, and DJNZ destinations without assembling.

Label Delimiter
 This option is ignored if no labels are created.
 0 No -- used for EDTASM
 1 Yes -- this option suffixes a colon (:) to the end of each
  label.

Main Delimiter
 0 Spaces -- disk output fill will have spaces in the line to
  separate labels from mneumonics from operands, from
  remarks, etc.

1 Tabs -- disk output will contain a single tab character (09)
 instead of spaces. This is standard for EDTASM.

Comments
 0 No -- no comments will be generated, even from '05' sections
 in the CMD file.
 1 Yes -- each instruction line will contain a comment, showing
 what bytes were used to generate that particular
 instruction. '05' sections encountered in the CMD file
 will also cause comments to be generated.

Comment Format
 This option is ignored if no comments are generated.
 0 ASCII -- comments are displayed as ASCII, non-ASCII are not
 displayed. This option is helpful in creating DEFM
 instructions.
 1 Hex -- each byte is displayed as two hexadecimal digits.

Hex Format
 0 0XX -- This option is not used by most assemblers.
 1 0XXH -- This option suffixes the letter H to any number
 designating a hexadecimal value.

EOF Character
 Any value may be used with the exception of SPACE (see above).
 For EDTASM, use 1A (press SHIFT, DOWN ARROW, and Z together).
 For some editors, use a RETURN.

Save Changes
 Position the pointer here and press ENTER to update DISASM with
 the displayed options.

Abort (Don't Save Changes)
 Position the pointer here and press ENTER to leave DISASM the
 way it was before this utility was entered.

 Of course, to use this utility, DISASM/CMD must be available
and non-write-protected.

 FILE OUTLINER

 This program displays the addresses in memory that will be
loaded when a given program executes.

Syntax: OUTLINE File
Examples: OUTLINE BIGPROG
 OUTLINE BASIC.PASSWORD:1

/CMD suffix assumed for filename.

As the program executes, you can press BREAK to exit the program. Pressing @ will freeze the display. Press any key but BREAK to re-start.

Any non-loaded remarks in the file are also displayed.

The last line printed will show the entry address (where execution will start after loading).

/CMD FILE MERGER

This program merges two or more executable programs together to form another, larger executable program.

```
Syntax:  MERGE File1 File2 (File3...FileN)
Examples: MERGE PARTA PARTB
  MERGE MAINPROG:1,DATA1,DATA2
```

Deliminaters between filenames can be spaces or commas.
/CMD suffix assumed for all filenames.

The program will prompt for an output filename. Type it in and press ENTER or press BREAK to exit the program. The output file MUST NOT be the same as any of the input files from the command line. /CMD is assumed.

If the output filename you entered is already in the directory, the program will prompt "File already exists, use it?". Press Y to overwrite, N to allow another filename to be entered, or BREAK to exit the program.

As the program executes, the display will show which file is being read. This is helpful in determining the cause of any read errors.

"Merge complete." will display when the program has successfully finished.

Note: Files are copied completely into the output file with the exeption of the entry address. Only one entry address may be specified per program. This program uses the entry address from the first input filename in the command line. All others are ignored.

FILE EXTRACTOR

This program creates a loadable program from a larger executable program.

Syntax: EXTRACT Address File
Example: EXTRACT A034 PROG:1

/CMD suffix assumed for file.

The Address must be the first loaded location of a loadable section. (i.e. the first number in a line from OUTLINE's output.) Leading 0's are not needed for the address, although they can be included.

The program will prompt for an output filename. Type it in and press ENTER or press BREAK to exit the program.

If the output filename you entered is already in the directory, the program will prompt 'File already exists, use it?'. Press Y to overwrite, N to allow another filename to be entered, or break to exit the program.

If the program finds your Address in the input file, a new loadable file will be created with entry address at DOS (402DH), and 'Extraction complete.' will be displayed.

If your Address is not found, 'Section not found.' will be displayed.

```
--------------------------------------------------------------
--
tektronix!tekred!ronbe   _____      Support Bacteria -
Ron Bemis               / o o \     It's the only
Tektronix              | \___/ |    culture some
Redmond, OR             \_____/     people have!
```