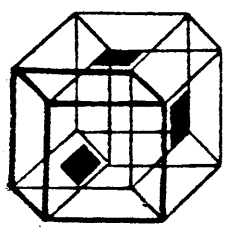
XYZT COMPUTER DIMENSIONS, INC.

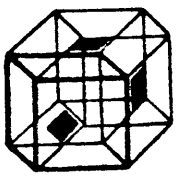
Ram Sledges

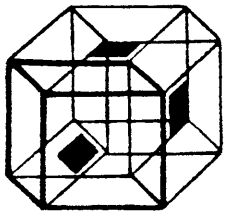
The utility producing a self-relocatable programs for the TRS-80 mod I/III computers.



C O N T E N T S

XYZT COMPUTER DIMENSIONS, INC.





XYZT COMPUTER DIMENSIONS, INC.

2 PENN PLAZA, SUITE 1500, NEW YORK, N.Y. 10121
(212) 244-3100

I N T R O D U C T I O N

Many TRS-80 programmers write their own machine language programs to enhance some of their software. For example various keyboard, display and printer drivers, USR functions for BASIC programs, etc. Such programs typically reside somewhere in high memory and typically there are conflicts with other software also residing in high memory. To avoid such memory conflicts the programs are made relocatable. Since assemblers and linkage editors for the TRS-80 do not provide any facilities for it this work is done manually - programmer must locate all relocatable addresses, write code that will relocate program to the new position and adjust all relocatable addresses, etc. This is quite a tedious and time consuming task, and every time the program is being modified the work must be repeated again, since addresses will be changed.

The RAM SLEDGES will assist you in preparing the self relocatable programs. You can write your assembly language programs without any consideration of future relocation. When the coding is finished and program is tested RAM SLEDGES will produce appendage to it so that the program upon execution will be relocated to the current top of memory.

RAM SLEDGES works with the TRS-80 models I/III computers (minimal configuration - 16K 1 disk system) and most major TRSDOS-like operating systems. Due to the method used - processing object code, it will work with all assemblers and linkage editors producing standard executable code for the mentioned operating systems. Two separate versions are included for each hardware model. A sample program - KEYCOPY/CMD is also included in the package.

About this manual. To ease understanding of the product it is organized in such way that at first you will get the general information - terminology and principles of operation, then installation procedure and finally operational instructions along with related technical information. Those of you who have the INTERACTIVE CONTROL LANGUAGE package will find some useful tips in appendix describing an ICL procedure to be used with RAM SLEDGES.

All comments and suggestions are welcomed!

2. G L O S S A R Y

Though the need for this utility will be recognized only by the experienced programmers and therefore it will be used by such, the documentation will be rather simple. First of all to avoid ambiguities here are some fundamental notions.

RELOCATION ADDRESS - an address reference in program that is subject to change if program is moved to a different location. It can be:
an address constant such as - DEFW START+10
or an intruction operand - LD HL,BUFFER

RELOCATION DICTIONARY - a table containing all relocation addresses within the program (for a given program).

SELF-RELOCATABLE PROGRAM - 1. A program that can be executed without changes at any address in memory;
2. A program that will be loaded somewhere in memory and then will relocate itself to another location. RAM SLEDGES will produce second type of self relocatable code, to be specific the code that will move itself to the current top of memory and then readjust all relocatable addresses within the program according to new location.

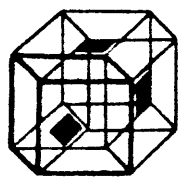
ABSOLUTE ADDRESS - an address reference within program that is not subject to change when the program is moved to another location in memory.

PROGRAM ORIGIN - location in memory where the program starts. Not necessarily same as entry point.

ENTRY POINT - address of the first instruction to be executed when program execution begins. Sometimes called transfer address.

HIGH MEMORY POINTER - a word in DOS area pointing to the high boundary of memory that can be used by DOS, BASIC and other programs. A memory above that boundary is considered protected and will not be used by mentioned software.
(The location is 4049H for mod I and 4411h for mod III)

This covers terminology used in this manual.



3. PRINCIPLES OF OPERATION

RAM SLEDGES produces an appendage to the program which will receive control upon execution of the program. This appendage contains a relocation dictionary and a code which will relocate the program.The program text will be moved to the current top of memory, then all relocatable addresses will be recalculated and adjusted, and then control will be transfered to the program entry point. The current top of memory pointer will be adjusted and will point below the program origin.

The only complication is determining the relocatable addresses within the program. Most assemblers as well as linkage editors available for the TRS-80 do not produce relocation dictionary. Disassembly of the program will not produce reliable information, since certain data may appear as instructions. In addition some address references outside the program text may be absolute or relative, and there is no way of determining which is which.

RAM SLEDGES bypasses this obstacle by using two files for input. The program should be assembled two times with different origin. Then comparing two files it is possible to locate all relocatable addresses within the program, since the only difference will be caused by difference in origin addresses.

RAM SLEDGES performs the following processing:

A. Compares two program files to locate the relocation addresses within the program.

B. Constructs the relocation dictionary.

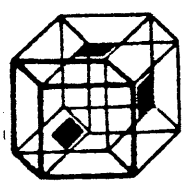
C. Creates new program file and adds appendage to it containing relocation dictionary.

D. Replaces program entry point address in the newly created file with the address of appendage entry point. Program entry point address is stored.

Notes:

a). Assembling or link-editing the program two times(in order to produce two input files for RAM SLEDGES) requires additional time and effort. However this is negligible comparing to what it takes to produce a self-relocatable module manually.

b). An INTERACTIVE CONTROL LANGUAGE (another product from XYZT Computer Dimensions, Inc.) can be used to automate creation of two program files. An example of procedure for the L80 linkage editor is given in appendix.



4. I N S T A L L A T I O N

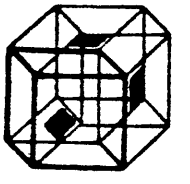
The diskette included in the package is in model I format. It can be copied WITHOUT ANY SPECIAL processing to the following model I operating systems - TRSDOS, NEWDOS/80, DOSPLUS, LDOS, MULTIDOS; and following model III operating systems - DOSPLUS, LDOS, MULTIDOS. Model III users of the TRSDOS 1.3 should use CONVERT*utility, model III users of the NEWDOS/80 should set appropriate model I PDRIVE specification and then boot the system before copying files.

The diskette contains the following files:

File name	Description	Size
RAMSLD1/CMD	Model I version of RAM SLEDGES	2g
RAMSLD3/CMD	Model III version of RAM SLEDGES	2g
KEYCOPY/ASM	Sample program KEYCOPY - source	1g
KEY6000/CMD	First input file	1g
KEY6001/CMD	Second input file	1g

You should copy a version of RAM SLEDGES according to your hardware model(either RAMSLD1 or RAMSLD3), and for your first experiments a sample programs - KEYCOPY/ASM, KEY6000/CMD and KEY6001/CMD.

Please note: this manual refers to RAMSLD1 in all cases for consistency. If you use model III then the right name is RAMSLD3.



5. USING RAM SLEDGES

Producing a self-relocatable programs with RAM SLEDGES is simple. As an example we will process the KEYCOPY program. The text of this program is given on the next page. The source code for that program is included on diskette, it is in EDTASM format.

At first we must assemble it two times. The program origin is specified in line 80. For the first assembly we'll take origin 6000H, for the second - 6001H. The names for the files produced are - KEY6000/CMD and KEY6001/CMD. (You don't actually have to do the assemblies, since these two files are provided on diskette.)

After that invoke RAM SLEDGES to generate a self-relocatable program -

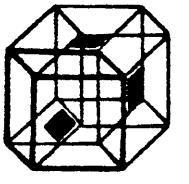
```
RAMSLD1 KEYCOPY/CMD,KEY6000/CMD,KEY6001/CMD
```

As a result of that a new file - KEYCOPY/CMD will be created. It is ready for use and you can start it right now. After that just press both shift key and a space bar and see how the characters from the line above are being copied to your input line. That function is convenient when you enter or edit BASIC programs. Consider for example the following piece of program -

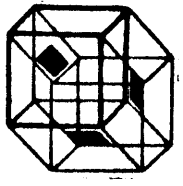
```
. . . . . .
  100 FOR I=1 TO 10
  110 FOR J=1 TO 10
  120 S = S + A(I,J)
  130 NEXT
  140 NEXT
. . . . . .
```

When you enter such text it is quite easy to copy line 110 from line 100 (which will be exactly above during entry in AUTO mode) with minor change of I to J, and copy line 140 from line 130. Similarly KEYCOPY can be used when entering assembler programs, etc. And of course it is quite easy to modify it to allow copying from the second line above your input line, third line and so on.

The main advantage here is that KEYCOPY is self-relocatable. Therefore it will not have conflicts with other programs residing in high memory. Upon startup it will relocate itself to the current top of memory. You can check the high memory pointer before starting KEYCOPY and after that to see the difference.



```
                    00010 ;-------------------------------------------------------;
                    00020 ; *** KEYCOPY *** - COPY CHARACTER FROM THE UPPER LINE. ;
                    00030 ;   CHECK THAT SHIFT KEY AND SPACE BAR ARE PRESSED,     ;
                    00040 ;   IF SO - RETURN CHARACTER FROM THE LINE ABOVE,       ;
                    00050 ;   OTHERWISE CONTINUE KEYBOARD SCAN.                   ;
                    00060 ;      COPYRIGHT (C) 1982 BY EUGENE SHKLYAR             ;
                    00070 ;-------------------------------------------------------;
6000                00080          ORG 6000H
6000 2A1640         00090 KEYCPY   LD HL,(4016H)          ;PLUG INTO KEYBOARD
6003 221560         00100          LD (OUT+1),HL          ;SCAN ROUTINE
6006 210F60         00110          LD HL,START
6009 221640         00120          LD (4016H),HL
600C C32D40         00130          JP 402DH               ;RETURN TO DOS
                    00140 ;
600F 3A8038         00150 START    LD A,(3880H)           ;CHECK SHIFT KEY
6012 E601           00160          AND 01H
6014 CA0000         00170 OUT      JP Z,0000H             ;RESUME SCAN ROUTINE
                    00180 ;
6017 1600           00190          LD D,0                 ;SET D=0
6019 01B80B         00200          LD BC,3000             ;REPEAT 3000 TIMES
601C 3A4038         00210 LOOP     LD A,(3840H)           ;CHECK SPACE BAR
601F E680           00220          AND 80H                ;AND KEEP RESULT IN D
6021 B2             00230          OR D
6022 57             00240          LD D,A
6023 0B             00250          DEC BC                 ;DECREASE COUNTER
6024 79             00260          LD A,C                 ;EXIT IF EXHAUSTED
6025 B0             00270          OR B
6026 20F4           00280          JR NZ,LOOP
                    00290 ;
6028 7A             00300          LD A,D                 ;IF STILL D=0 THEN
6029 FE00           00310          CP 0                   ;GOTO SCAN ROUTINE
602B 28E7           00320          JR Z,OUT
602D 2A2040         00330          LD HL,(4020H)          ;GET CURSOR POSITION
6030 114000         00340          LD DE,64               ;SUBTRACT 64 THUS POSI-
6033 ED52           00350          SBC HL,DE              ;TIONING TO PREVIOUS LINE
6035 7E             00360          LD A,(HL)              ;RETURN CHARACTER FROM
6036 C9             00370          RET                    ;THE PREVIOUS LINE
6000                00380          END KEYCPY
```

nur für kleinere Programme von einfacher Struktur

Now the technical information.

5.1 The format of RAMSLD1 invocation is:

RAMSLD1 <relo-file>,<input-file1>,<input-file2>

Where: relo-file - filespec for the resulting self-relocatable module produced by the RAM SLEDGES;
input-file1 - filespec for the first input file;
input-file2 - filespec for the second input file;

Filespec stands for a regular TRS-80 file specification - filename/ext:drive

5.2 The following should be taken into consideration -

- The origin of the input files must be at or above 6000H address.

- The difference in origin addresses should be equal to 0001H.

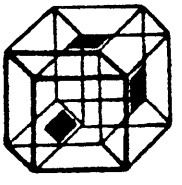
- The output file is produced from the first input file. It will be loaded at the same address in memory as the first input file, and then relocated to high memory.

- If I/O error occurs while processing RAMSLD1 exits to DOS and error message is displayed.

- The input files should have only regular 01h (text block) and 02h (end of program) block headers. Other types of block headers will not be recognized.

- The program text must include all used data areas. For example if your program uses storage immediately after the end of program text you should either define that area via DEFM instruction, or at least put a byte constant at the end of that area. In that case RAM SLEDGES will know that this area belongs to the program, and it will be considered when computing size of the program. Otherwise errors will occur.

This utility was originally developed by XYZT Computer Dimensions for the internal use. The LIBRARY SUPPORT OPTION and a relocatable version of ICL were produced using RAM SLEDGES. We hope you will find it useful too.



6. A P P E N D I X

If you use EDTASM or a similar assembler generating an executable code the overhead caused by the need to produce two input files is minimal - just extra assembly. However if your assembler (such as M80) subsequently requires link-editing it may be time consuming to do it manually.

This extra burden can be eased if you have the INTERACTIVE CONTROL LANGUAGE. The whole process can be automated using ICL facilities. As an example a procedure to be used with the L80 linkage editor is described below.

The following ICL procedure builds two input files(origins 6000H and 6001H) and then executes the RAM SLEDGES. In order to use it you must prepare a file containing names of the modules to be link edited together. The extension of that file must be /BLD. The procedure takes data from that file and passes it to L80, supplying appropriate origin addresses for creation of input files.

```
10  RELO &1
20  &QUEUE -P:6000
30  &DOS QLOAD &1/BLD
40  &QUEUE &1A-N-E
50  &DOS L80
60  &TYPE ==> first module built "&1A/CMD"
70  &QUEUE -P:6001
80  &DOS QLOAD &1/BLD
90  &QUEUE &1B-N-E
100 &DOS L80
110 &TYPE ==> second module built "&1B/CMD"
120 &DOS RAMSLD1 &1/CMD,&1A/CMD,&1B/CMD
130 &TYPE ==> relocatable module built "&1/CMD"
140 &KILL &1A/CMD
150 &KILL &1B/CMD
160 &EXIT
```

For example you create a TEST/BLD file containing names of all modules to be link-edited together. The contents of that file is something like -

```
TEST,SUBR1,SUBR2
SUBR3,SUBR4,SUBR5
```

To produce a self-relocatable file using RELO enter -

```
RELO TEST
```

The procedure will build two input files - TESTA/CMD and TESTB/CMD, then it will produce a relocatable version TEST/CMD and after that it will kill both input files.