Z-80
OBJECT CODE RELOCATOR
MANUAL

Written by
Joseph E. Willis
and
Helen W. Gottschalk

for
Digital Group Software

the digital group
po box 6528 denver, colorado 80206 (303) 777-7133

296-011-A-11+16

DIGITAL GROUP Z-80 OBJECT CODE RELOCATOR

So who needs an "object code relocator" and, anyway, what is it? Well, let's start with "object code", which is the machine code which feeds your marvelous Z-80 monster-pet, ZORK. There is a lot of code around, but it seems that every time you stuff some of this food down its throat, ZORK gets a bad case of indigestion! The Relocator described herein is the computerniks way of spelling R-E-L-I-E-F.

You see, the problem with machine language is that it normally will execute successfully only in the specific memory locations for which the program was originally assembled. This is no problem if you have that area of memory available, but since when has this ever happened? All that neat 8080 and Z-80 stuff in the computer magazines is always either too low or too high to fit ZORK's memory with all the other goodies you already have there.

So, what can be done to prevent mayhem? Well, if you have a computer-ready source code that ZORK can read, you have no problem--all you do is feed the tape to your assembler using the appropriate starting address. What if you don't have a tape of the source code? You can always key it in, if a listing is available, but for programs of any length this can be a bore. (Of course, if you want to make some changes, you practically must use source code.)

If the routine is satisfactory except for location, you are in luck! By learning how to use the Relocator, you can easily modify the routine to run where you need it. The Relocator is also useful when you have large programs which must be assembled in several parts; the Relocator can probably handle it in one pass. When you need to move everything up a page or two (e.g., to accommodate a 64-character TV buffer), it is much easier to do so using the Relocator rather than the Assembler.

Now that you are convinced you need to use the Relocator, let's get down to the business of what it is and how it works. The Z-80 object code Relocator is an assembly language program designed to examine each byte of 8080 or Z-80 object code in the program you want relocated, adjusting all jumps, calls, etc., as necessary to make the program fully operational (provided, of course, that the original program was a working one! You didn't expect miracles, did you?). Let us first compare the Z-80 instruction set with that of the 8080.

Z-80 vs 8080

Unlike relocators designed for the 8080 instruction set only, this program can handle the much larger Z-80 code as well as the 8080 code. The 8080 has a straightforward set of machine codes in that

(1) All 3-byte and 2-byte instructions are uniquely determined by the 1st byte;

(2) All 3-byte instructions have an address or 16 bits of data as the 2nd and 3rd bytes, and all may require change after relocation.

The Z-80 with its more powerful instruction set has 3 types of special commands beginning with DD, FD, or ED (hex) that can be either 2, 3 or 4 bytes in length. The DD and FD commands are similar, with DD beginning instructions for the index register IX and FD starting the corresponding instructions for the index register IY. (These index registers are not even present in the 8080!)

When the above special cases are encountered, the 2nd byte must be tested to determine the instruction length. For 4 byte DD and FD instructions, the 3rd and 4th bytes contain an address or 16 bit data which must be inspected for possible change. For 3 byte DD and FD instructions, the last byte contains data not requiring change. The ED commands consist of instructions such as 16 bit load, addition and subtraction, block transfer, block searches, block input and output, etc., and all that good stuff that makes the Z-80 so great. These ED commands may be either 2 bytes or 4 bytes in length, with the 2-byte instructions requiring no change, while the 4-byte instructions may or may not require change.

Overview of Relocation Algorithm

The program uses a table-lookup procedure to determine instruction lengths. Initially it tests the first byte for the 3 special cases of DD, FD, and ED instructions; if it is one of these the program further tests the 2nd byte as discussed above, and makes appropriate changes if necessary. If not, the byte is checked to see if it is the beginning of a 3-byte instruction that might need address adjustment. Otherwise it is compared to the 2-byte instruction table.

If it has not been found in any of the tables, it is assumed to be a 1-byte instruction. Then, the next byte is brought in and the search starts again.

Features of the Relocator

The Relocator operates on either a resident object file, or an object file read in from audio casette. It allows 3 different tasks: 0 for move only, 1 for fix references only, 2 for fix references followed by move. At completion of any of these, you have the option to save the relocated code out on audio cassette. After references have been fixed, the program gives a memory dump of the relocated program.

Keying in a ctrl X at any time during the dump causes it to be aborted. To get a hardcopy of the dump, use option 8 instead of 7 when selecting from the menu.

In the discussion which follows, as well as in the attached source code, several mnemonics are used which should be described at this time:

STSORC and NDSORC refer to the start and end, respectively, of the original code generated by the assembler. They are the addresses where the routines you want to relocate are presently designed to operate.

STDEST and NDDEST are the starting and ending addresses, respectively, where you want the program to operate. Using the relocator, all references will be modified (if necessary) so that routines designed originally to operate in memory locations STSORC to NDSORC will now operate between STDEST and NDDEST.

STRESD and NDRESD are the starting and ending addresses, respectively, where the object code is actually residing in memory at the time you want the Relocator to alter it. (This can be anywhere in memory not occupied by the Relocator itself.) STRESD and NDRESD need not coincide with either the corresponding STSORC and NDSORC or STDEST and NDDEST. For example, it may be convenient to place a routine into memory starting at 0100H and you want to relocate it to start at 0B00H. STRESD and NDRESD give you complete flexibility in this respect.

In the ideal program all data storage areas, message areas, etc., are located together, completely separate from the executable code. At least, that is supposedly the ideal. However, if you are the typical beginning programmer, your philosophy is to get it working whichever way you can and then leave well enough alone! (Even you experienced programmers remember the feeling, don't you? Of course, you would never be guilty of that now! Oh, yeah?)

The normal result of all this is a patchwork of main program, data areas, subroutines, messages, etc., in whatever order they were written. While we do not want to encourage this, we have enough routines like that described above to cause us to allow selective relocation of sections of code, bypassing data storage areas, messages, etc. Each section is defined by a STADDR and NDADDR pair. You can specify up to 30 STADDR and NDADDR pairs as long as these are within the boundaries of STSORC and NDSORC. Good programming practice of putting all data statements at the end of the program will eliminate all but 1 pair of STADDR and NDADDR addresses.

Now let us look at the tasks the Relocator can perform for us:

Task 0: This moves the object code from location between STRESD and NDRESD to STDEST. Any degree of overlap is permissible, e.g., the code can be moved by only one byte if you desire. The routine automatically chooses the correct move routine so that the code integrity is maintained whether the relocation is to higher memory or lower memory.

Task 1: This fixes references by calculating the number of bytes of displacement between STSORC and STDEST. Note that STSORC and/or STDEST may or may not coincide with STRESD. In other words, the object code, which is physically located at STRESD, is designed to run if located at STSORC, or at STDEST after completion of Task 1. This is useful to generate an object program that can eventually be run at memory locations where the op sys and the relocater are presently residing.

Task 2: This fixes references, then performs the move from STRESD to STDEST.

Restrictions

An unavoidable possibility of error arises with the instruction "LD rp, nn" where rp refers to a register pair BC, DE, or HL, and nn can either be an address or a 16 bit value. For example "LD BC, 1000" where 1000 is intended to be a counter or a value is unambiguous as long as 1000 lies outside the limits of STSORC and NDSORC. Otherwise it will

be interpreted as fair game to be adjusted by the displacement. You will need to unadjust it manually when the program is finished.

Another possibility of error stems from vector table addresses such as

```
VTABL     DW Route 1
          DW Route 2
          DW Route 3
```

These should be treated as data using the Relocator, then manually adjusted to reflect the displacement.

A third restriction involves relocating the Relocator itself. The Relocator <u>first</u> fixes references, <u>then</u> moves the code (if Option 2 is selected), so anytime the Relocator or a portion of it is to be relocated, use Option 0 to duplicate it into another area of memory. Then you can use Option 1 to fix the references. Another convenient approach if you have an audio tape of the Relocator is to use the features which will input the tape, perform the reference fixing, and save a new audio tape. The new audio tape can then be used to load the Relocator into the new memory area.

How to Use

1. Place the cassette labelled "Z-80 Relocator" in the cassette recorder. When the low tone begins, push reset.

2. When the data burst ends, the screen should display a menu for option select.

3. Option 7 (or 8 for hardcopy) begins the Relocator Program. It is in conversational mode with instructions to assist in its use.

The following is an example of how the relocator works. The underlined characters in Figure 1 are the user's response.

FIGURE 1

Z-80 RELOCATE PROGRAM

```
ENTER TASK NUMBER:
          0=MOVE ONLY
          1=ADJUST REFS ONLY
          2=MOVE & ADJUST REFS

2
ENTER ADDRESS MODE (O/H) H
INPUT FROM TAPE? (Y/N) Y
START CASSETTE, THEN PRESS ANY KEY

DEST START      =  3000
SOURCE START    =  0E00
STADDR          =  0E00
NDADDR          =  11EE
STADDR          =  Cr
 30 00   21   37   34   CD   9A   32
 30 06   CD   69   32   7E   E6   0F
 30 00   FE   03   30   F0   32   BD
 30 12   35   F5   21   9C   34   CD
 30 18   9A   32   CD   69   32   7E
    Etc., for the rest of the dump

DO YOU WISH TO WRITE NEW TAPE? (Y/N) N
```

For this example we chose to input an audio tape of the object code Relocator routine. The object code covers the area from 0E00H to 13D5H. The source code for the Relocator is in Appendix I.

We pressed option 7 when the Relocator menu was displayed. The routine responded by displaying the Task options. We chose option 2, and then chose the Hex address mode. The audio tape was then loaded as shown.

We wanted the new starting location (DEST START) to be 3000H. The original starting address (SOURCE START) is 0E00H. Executable code resides between 0E00H and 11EEH, and data tables, messages, and data storage areas are between 11EFH and 13DSH. We therefore entered 0E00H and 11EEH as our first (and only) STADDR and NDADDR, respectively.

When the next STADDR was requested, we pressed the carriage return key. When the task was completed a dump of the relocated code was given. (To save space only the first part of the dump is shown.) After the dump was complete or manually aborted by Ctrl X, the routine asked if we wished a new tape. In this case we did not, so we said no.

```
FILE  3000  674C
READY
LTABL
ADDLUP 1110   ADDRLD 103B   ADJUST 1032   ADRLUP 0E89
ASCCON 11A9   ASCHAR 116F   BKSPCE 107A   BYTCNT 13C7
BYTE2  0F2B   BYTE3  0F21   BYTRED 009C   CHCKND 0F36
CHKCNG 0F82   CKDDFD 0F75   CONTIN 0F11   DDFD   0F60
DELDCK 102B   DESTST 0E68   DISP   13D1   DMPLP1 1134
DMPLP2 114A   DSPLAY 1128   DSPLDN 1167   EDFIND 0FA7
ERASE  0900   ERMSG  137B   FINISH 0F49   FIXREF 0ED7
HEX    10F1   HLINC  10B5   INC1   0F34   INITPR 0E7E
KEYIN  09A5   KYBDST 1021   LDCA   110D   LDEND  0E9C
LINBUF 139B   LINEIN 1069   LINEND 1097   LNLUP  1070
LODEND 11E9   LODRIT 1017   LT10   11B3   MEMOK  11DE
MENU   0500   MOVE   0FB9   MOVED  1010   MSG1   1237
MSG2   129C   MSG3   12B6   MSG5   12C6   MSG6   12D7
MSG7   12E6   MSG8   12F5   MSGTAP 130F   MUVDUN 0FE2
MUVDWN 0FE0   MUVUP  0FD5   NDDEST 13CF   NDRESD 13C1
NDRITE 011A   NDSORC 13CB   NOTAPE 0E45   NPAIRS 001E
NTCR   10D6   NUMBER 10E2   NXDSPL 1156   NXTCHK 0F4C
NXTCHR 10EB   NXTTST 1101   OCT    0E27   OCTAL  10FF
OCTNUM 118B   OFFSET 13C3   OFFST1 13C5   OUT1   11A4
OUTCR  10DA   OUTGO  111B   OUTPUT 11B5   PAIRS  13BE
RDTAPE 11B9   RELOC  0E00   RELOC1 0EB0   RUBOUT 1082
SKIP1  0F71   SKIP2  0F6F   SPACE  10B8   SPCLUP 10B0
START  0EFC   STBLDF 1222   STBLED 1231   STDEST 13CD
STMSG  1327   STORE  108A   STRESD 13BF   STRITE 0118
STRT1  13D5   STRTBT 11C0   STSORC 13C9   SUBSTP 1103
TABLE2 1209   TABLE3 11EF   TABLOD 1054   TAPMSG 134B
TASK   13BD   TBPNTR 13BB   TDELAY 11C6   TEMP   13D3
TV     00FA   TVCALL 11E3   TVCR   0993   TVOUT  10A8
TVOUT1 0928   TVTED  109A   TVTI   10BE   TVTO   10CE
VALCHR 1123   WRITE  0134   WRTMSG 1372   ZROTST 10AD

FILE  3000  674C
READY
ASSM

674D              0100 *
674D              0110 * Z-80 OBJECT CODE RELOCATE PROGRAM
674D              0120 * WRITTEN BY   JOSEPH E. WILLIS
674D              0130 *          &   HELEN W. GOTTSCHALK
674D              0140 *              HARSHAW CHEMICAL COMPANY
674D              0150 *              CLEVELAND, OHIO
674D              0160 *
0E00              0170        ST    0E00H
0E00              0180 NPAIRS EQU   30D
0E00              0190 MENU   EQU   5000
0E00              0200 BYTRED EQU   009CH
0E00              0210 TV     EQU   000372
0E00              0220 TVOUT1 EQU   0928H
0E00              0230 TVCR   EQU   0993H
0E00              0240 KEYIN  EQU   09A5H
```

```
0E00                0250 ERASE  EQU  0900H
0E00                0260 STRITE EQU  001030
0E00                0270 NDRITE EQU  001032
0E00                0280 WRITE  EQU  001064
0E00 21 37 12       0290 RELOC  LD   HL,MSG1       DISPLAY THE
0E03 CD 9A 10       0300        CALL TVTED         TASK MESSAGE
0E06 CD 69 10       0310        CALL LINEIN        FETCH TASK
0E09 7E             0320        LD   A,(HL)        & MASK IT
0E0A E6 0F          0330        AND  0FH           FOR TESTING
0E0C FE 03          0340        CP   3             IS IT VALID NUM?
0E0E 30 F0          0350        JR   NC,RELOC      IF NOT, TRY AGAIN
0E10 32 BD 13       0360        LD   (TASK),A      STORE IF VALID
0E13 F5             0370        PUSH AF            SAVE TASK DATA
0E14 21 9C 12       0380        LD   HL,MSG2       DISPLAY
0E17 CD 9A 10       0390        CALL TVTED         FORMAT MSG
0E1A CD 69 10       0400        CALL LINEIN        FETCH FORMAT
0E1D 7E             0410        LD   A,(HL)
0E1E 2B             0420        DEC  HL
0E1F FE C8          0430        CP   'H'          IS IT HEX?
0E21 20 04          0440        JR   NZ,OCT
0E23 3E 04          0450        LD   A,4           IF HEX, MODE IS 4
0E25 18 02          0460        JR   OCT+2
0E27 3E 03          0470 OCT    LD   A,3           ELSE, MODE IS 3
0E29                0480 *
0E29                0490 *STORE MODE IN 1ST BYTE OF LINBUF
0E29                0500 *
0E29 77             0510        LD   (HL),A
0E2A 21 0F 13       0520        LD   HL,MSGTAP     INPUT FRM TAPE?
0E2D CD 9A 10       0530        CALL TVTED
0E30 CD BE 10       0540        CALL TVTI
0E33 CD DA 10       0550        CALL OUTCR
0E36 FE D9          0560        CP   'Y'
0E38 20 0B          0570        JR   NZ,NOTAPE
0E3A 21 51 14       0580        LD   HL,STRT1+124D IF SO,LD
0E3D 22 BF 13       0590        LD   (STRESD),HL   DEFAULT ADDR &
0E40 CD B9 11       0600        CALL RDTAPE        READ TAPE
0E43 18 23          0610        JR   DESTST
0E45 21 D7 12       0620 NOTAPE LD   HL,MSG6       IF NOT, GET
0E48 CD 3B 10       0630        CALL ADDRLD        STRT ADDR
0E4B 22 BF 13       0640        LD   (STRESD),HL   & STORE
0E4E 21 E6 12       0650        LD   HL,MSG7       GET END ADDR
0E51 CD 3B 10       0660        CALL ADDRLD        & STORE
0E54 22 C1 13       0670        LD   (NDRESD),HL
0E57 ED 5B BF 13    0680        LD   DE,(STRESD)
0E5B AF             0690        XOR  A
0E5C ED 52          0700        SBC  HL,DE
0E5E 30 08          0710        JR   NC,DESTST
0E60 21 7B 13       0720        LD   HL,ERMSG       ERROR MESSAGE
0E63 CD 9A 10       0730        CALL TVTED
0E66 18 DD          0740        JR   NOTAPE
0E68 21 C6 12       0750 DESTST LD   HL,MSG5       DEST ST MSG
0E6B CD 3B 10       0760        CALL ADDRLD        GET STDEST
0E6E 22 CD 13       0770        LD   (STDEST),HL   & STORE
0E71 F1             0780        POP  AF            TASK DATA
```

```
0E72 B7              0790          OR   A
0E73 28 3B           0800          JR   Z,RELOC1
0E75 21 B6 12        0810          LD   HL,MSG3        SOURCE STRT MSG
0E78 CD 3B 10        0820          CALL ADDRLD         GET SOURCE STRT
0E7B 22 C9 13        0830          LD   (STSORC),HL    & STORE
0E7E 21 D5 13        0840 INITPR   LD   HL,STRT1       INITIALIZE
0E81 22 BB 13        0850          LD   (TBPNTR),HL    TABLE POINTER
0E84 3E 1E           0860          LD   A,NPAIRS       STORE MAX
0E86 32 BE 13        0870          LD   (PAIRS),A      ADDR PAIRS
0E89 21 D7 12        0880 ADRLUP   LD   HL,MSG6        STADDR MSG
0E8C CD 3B 10        0890          CALL ADDRLD         GET A STRTNG ADDR
0E8F E5              0900          PUSH HL
0E90 CD 54 10        0910          CALL TABLOD         & LOAD IN TABLE
0E93 E1              0920          POP  HL
0E94 28 06           0930          JR   Z,LDEND        IF NULL LINE
0E96 21 E6 12        0940          LD   HL,MSG7        NDADDR MSG
0E99 CD 3B 10        0950          CALL ADDRLD         GET NDADDR
0E9C CD 54 10        0960 LDEND    CALL TABLOD         & STORE
0E9F 28 0F           0970          JR   Z,RELOC1       IF NULL LINE
0EA1 3A BE 13        0980          LD   A,(PAIRS)      UPDATE PAIRS
0EA4 3D              0990          DEC  A              COUNTER
0EA5 32 BE 13        1000          LD   (PAIRS),A
0EA8 20 DF           1010          JR   NZ,ADRLUP
0EAA AF              1020          XOR  A              SET ZERO FLAG
0EAB 21 00 00        1030          LD   HL,0
0EAE 18 DF           1040          JR   ADRLUP+6
0EB0 AF              1050 RELOC1   XOR  A
0EB1 ED 5B BF 13     1060          LD   DE,(STRESD)    RESIDENT STRT
0EB5 2A C1 13        1070          LD   HL,(NDRESD)    RESIDENT END
0EB8 ED 52           1080          SBC  HL,DE          GET BYTE CNT
0EBA 23              1090          INC  HL
0EBB 22 C7 13        1100          LD   (BYTCNT),HL    & SAVE
0EBE EB              1110          EX   DE,HL
0EBF 2A CD 13        1120          LD   HL,(STDEST)    CALC
0EC2 19              1130          ADD  HL,DE          DEST END
0EC3 2B              1140          DEC  HL
0EC4 22 CF 13        1150          LD   (NDDEST),HL    & STORE
0EC7 2A C9 13        1160          LD   HL,(STSORC)    CALC
0ECA 19              1170          ADD  HL,DE          SOURCE END
0ECB 2B              1180          DEC  HL             & STORE
0ECC 22 CB 13        1190          LD   (NDSORC),HL
0ECF 3A BD 13        1200          LD   A,(TASK)
0ED2 FE 00           1210          CP   0
0ED4 CA B9 0F        1220          JP   Z,MOVE
0ED7 21 D5 13        1230 FIXREF   LD   HL,STRT1       SET TABLE PNTR
0EDA 22 BB 13        1240          LD   (TBPNTR),HL    AT TOP
0EDD AF              1250          XOR  A              DISP=STDEST-STSORC
0EDE 2A CD 13        1260          LD   HL,(STDEST)    CALC
0EE1 ED 5B C9 13     1270          LD   DE,(STSORC)    DISPLACEMENT
0EE5 ED 52           1280          SBC  HL,DE          AND
0EE7 22 D1 13        1290          LD   (DISP),HL      STORE
0EEA E5              1300          PUSH HL
0EEB AF              1310          XOR  A
0EEC 2A BF 13        1320          LD   HL,(STRESD)
```

```
0EEF ED 52          1330          SBC  HL,DE          OFFSET=STRESD-STSORC
0EF1 22 C3 13       1340          LD   (OFFSET),HL
0EF4 EB             1350          EX   DE,HL
0EF5 E1             1360          POP  HL
0EF6 AF             1370          XOR  A
0EF7 ED 52          1380          SBC  HL,DE          OFFST1=DISP-OFFSET,IE,
0EF9 22 C5 13       1390          LD   (OFFST1),HL    OFFST1=STDEST-STRESD
0EFC 2A BB 13       1400 START    LD   HL,(TBPNTR)
0EFF CD 2B 10       1410          CALL DELDCK         GET STRT ADDR
0F02 D5             1420          PUSH DE
0F03 DD E1          1430          POP  IX
0F05 CD 2B 10       1440          CALL DELDCK         GET END ADDR
0F08 CA 28 11       1450          JP   Z,DSPLAY       &, IF NOT 0,
0F0B D5             1460          PUSH DE
0F0C FD E1          1470          POP  IY
0F0E 22 BB 13       1480          LD   (TBPNTR),HL    SAVE TABLE PNTR
0F11 DD 7E 00       1490 CONTIN   LD   A,(IX+0)       1ST BYTE OF INSTR
0F14 FE DD          1500          CP   0DDH           SPECIAL
0F16 28 48          1510          JR   Z,DDFD         HANDLING
0F18 FE ED          1520          CP   0EDH           FOR
0F1A CA A7 0F       1530          JP   Z,EDFIND       DD,
0F1D FE FD          1540          CP   0FDH           ED,
0F1F 28 3F          1550          JR   Z,DDFD         & FD INSTRS.
0F21 01 1A 00       1560 BYTE3    LD   BC,1AH         CHK OTHER
0F24 21 EF 11       1570          LD   HL,TABLE3      3-BYTE INSTRS
0F27 ED B1          1580          CPIR                IF FOUND, CHK
0F29 28 57          1590          JR   Z,CHKCNG       FOR POSS. CHNG
0F2B 01 19 00       1600 BYTE2    LD   BC,25D         IS IT THE BEGIN.
0F2E ED B1          1610          CPIR                OF 2-BYTE
0F30 20 02          1620          JR   NZ,INC1        INSTR?
0F32 DD 23          1630          INC  IX             IF SO, SKIP
0F34 DD 23          1640 INC1     INC  IX             2 BYTES
0F36 DD E5          1650 CHCKND   PUSH IX             HAVE WE REACHED
0F38 E1             1660          POP  HL             END OF BLOCK?
0F39 ED 5B C3 13    1670          LD   DE,(OFFSET)    SORC=RESD-OFFSET
0F3D AF             1680          XOR  A
0F3E ED 52          1690          SBC  HL,DE
0F40 EB             1700          EX   DE,HL
0F41 2A CB 13       1710          LD   HL,(NDSORC)
0F44 AF             1720          XOR  A
0F45 ED 52          1730          SBC  HL,DE          IF SO,
0F47 30 03          1740          JR   NC,NXTCHK
0F49 C3 28 11       1750 FINISH   JP   DSPLAY         EXIT TO DISPLAY.
0F4C AF             1760 NXTCHK   XOR  A              ELSE, CHK TO SEE
0F4D FD E5          1770          PUSH IY             IF AT END
0F4F E1             1780          POP  HL             OF A SECTION
0F50 D5             1790          PUSH DE
0F51 ED 5B C3 13    1800          LD   DE,(OFFSET)    SORC=RESD-OFFSET
0F55 ED 52          1810          SBC  HL,DE
0F57 D1             1820          POP  DE
0F58 AF             1830          XOR  A
0F59 ED 52          1840          SBC  HL,DE          IF NOT,
0F5B 30 B4          1850          JR   NC,CONTIN      CONTINUE; ELSE,
0F5D C3 FC 0E       1860          JP   START
```

```
0F60 21 22 12      1870 DDFD   LD   HL,STBLDF     IF 1ST BYTE OF
0F63 DD 23         1880        INC  IX            INSTR IS DD OR FD,
0F65 DD 7E 00      1890        LD   A,(IX+0)      SEARCH TABLE FOR
0F68 01 0F 00      1900        LD   BC,15D        2ND BYTE. IF FOUND,
0F6B ED B1         1910        CPIR               MUST CHK WHETHER
0F6D 28 06         1920        JR   Z,CKDDFD      CHNG IS REQD
0F6F               1930 *
0F6F               1940 *IF NO MATCH, THEN 3 BYTES, NO CHNG
0F6F               1950 *
0F6F DD 23         1960 SKIP2  INC  IX
0F71 DD 23         1970 SKIP1  INC  IX
0F73 18 C1         1980        JR   CHCKND
0F75               1990 *
0F75               2000 *ROUTINE CHKS DD OR FD INSTR FOR POSS. CHNG
0F75               2010 *
0F75 79            2020 CKDDFD LD   A,C
0F76               2030 *
0F76               2040 *>=13, 4 BYTES, REPLACE
0F76               2050 *
0F76 FE 0D         2060        CP   13D
0F78 30 08         2070        JR   NC,CHKCNG
0F7A               2080 *
0F7A               2090 *<13 OR >=11, 4 BYTES, NO REPLACE
0F7A               2100 *
0F7A FE 0B         2110        CP   11D
0F7C 38 F3         2120        JR   C,SKIP1       SKIP 1 MORE BYTE
0F7E DD 23         2130        INC  IX
0F80 18 ED         2140        JR   SKIP2         SKP 2 BYTES
0F82               2150 *
0F82               2160 *ROUTINE CHKS LAST 2 BYTES OF 3-
0F82               2170 *OR 4-BYTE INSTR FOR POSS. CHNG.
0F82               2180 *
0F82 DD 23         2190 CHKCNG INC  IX
0F84 DD 6E 00      2200        LD   L,(IX+0)      LD HL WITH LAST
0F87 DD 66 01      2210        LD   H,(IX+1)      2 BYTES OF INSTR
0F8A EB            2220        EX   DE,HL
0F8B 2A C9 13      2230        LD   HL,(STSORC)   IF <SOURCE
0F8E AF            2240        XOR  A             START, NO
0F8F ED 52         2250        SBC  HL,DE         CHNG, SO
0F91 30 DC         2260        JR   NC,SKIP2      SKIP 2 BYTES.
0F93 2A CB 13      2270        LD   HL,(NDSORC)   IF >SOURCE
0F96 23            2280        INC  HL            END, NO
0F97 ED 52         2290        SBC  HL,DE         CHNG, SO
0F99 38 D4         2300        JR   C,SKIP2       SKIP 2 BYTES
0F9B               2310 *
0F9B               2320 *IF >=STSORC & <=NDSORC, CHNG LAST 2 BYTES
0F9B               2330 *
0F9B 2A D1 13      2340        LD   HL,(DISP)     GET DISPLACEMENT
0F9E 19            2350        ADD  HL,DE         ADJUST ADDR
0F9F DD 75 00      2360        LD   (IX+0),L      STORE BACK,
0FA2 DD 74 01      2370        LD   (IX+1),H      & POSITION PNTR
0FA5 18 C8         2380        JR   SKIP2         FOR NXT INSTR
0FA7 21 31 12      2390 EDFIND LD   HL,STBLED     SEARCH ED TABLE
0FAA DD 23         2400        INC  IX            FOR 2ND BYTE.
```

```
0FAC DD 7E 00      2410          LD   A,(IX+0)      IF FOUND, POSS.
0FAF 01 06 00      2420          LD   BC,6          CHNG REQD,
0FB2 ED B1         2430          CPIR               ELSE, NOT REQD.
0FB4 C2 34 0F      2440          JP   NZ,INC1
0FB7 18 C9         2450          JR   CHKCNG
0FB9 3A BD 13      2460 MOVE     LD   A,(TASK)
0FBC FE 01         2470          CP   1
0FBE 28 22         2480          JR   Z,MUVDUN
0FC0 2A BF 13      2490          LD   HL,(STRESD)   STRESD TO HL
0FC3 ED 5B CD 13   2500          LD   DE,(STDEST)   DEST STRT
0FC7 ED 4B C7 13   2510          LD   BC,(BYTCNT)
0FCB 7A            2520          LD   A,D           DOWN MOVE?
0FCC BC            2530          CP   H
0FCD 38 11         2540          JR   C,MUVDWN      DOWN MOVE
0FCF 20 04         2550          JR   NZ,MUVUP      UP MOVE
0FD1 7B            2560          LD   A,E
0FD2 BD            2570          CP   L
0FD3 38 0B         2580          JR   C,MUVDWN
0FD5 2A C1 13      2590 MUVUP    LD   HL,(NDRESD)
0FD8 ED 5B CF 13   2600          LD   DE,(NDDEST)
0FDC ED B8         2610          LDDR               MOVE BLOCK
0FDE 18 02         2620          JR   MUVDUN
0FE0 ED B0         2630 MUVDWN   LDIR               MOVE BLOCK
0FE2 21 4B 13      2640 MUVDUN   LD   HL,TAPMSG     IS NEW TAPE
0FE5 CD 9A 10      2650          CALL TVTED         DESIRED?
0FE8 CD BE 10      2660          CALL TVTI
0FEB CD DA 10      2670          CALL OUTCR
0FEE FE CE         2680          CP   'N'
0FF0 CA 00 05      2690          JP   Z,MENU
0FF3 FE D9         2700          CP   'Y'
0FF5 20 F1         2710          JR   NZ,MUVDUN+6
0FF7 CD 21 10      2720          CALL KYBDST
0FFA 21 72 13      2730          LD   HL,WRTMSG
0FFD CD 9A 10      2740          CALL TVTED
1000 3A BD 13      2750          LD   A,(TASK)      WAS CODE
1003 FE 01         2760          CP   1             MOVED?
1005 20 09         2770          JR   NZ,MOVED
1007 2A BF 13      2780          LD   HL,(STRESD)   IF NOT, USE
100A ED 5B C1 13   2790          LD   DE,(NDRESD)   RESD ADDRS
100E 18 07         2800          JR   LODRIT
1010 2A CD 13      2810 MOVED    LD   HL,(STDEST)   IF SO, USE
1013 ED 5B CF 13   2820          LD   DE,(NDDEST)   DEST ADDRS
1017 22 18 01      2830 LODRIT   LD   (STRITE),HL
101A ED 53 1A 01   2840          LD   (NDRITE),DE
101E C3 34 01      2850          JP   WRITE         WRITE TAPE
1021 21 27 13      2860 KYBDST   LD   HL,STMSG
1024 CD 9A 10      2870          CALL TVTED
1027 CD BE 10      2880          CALL TVTI
102A C9            2890          RET
102B               2900 *
102B               2910 *ROUTINE LDS DE W/1 ADDR OF ADDRESS
102B               2920 *PAIR, & ADJUSTS FOR OFFSET.
102B               2930 *
102B 5E            2940 DELDCK   LD   E,(HL)
```

```
102C 23          2950        INC  HL
102D 56          2960        LD   D,(HL)
102E 23          2970        INC  HL
102F 7A          2980        LD   A,D          TEST FOR ZERO
1030 B3          2990        OR   E
1031 F5          3000        PUSH AF           SAVE FLAGS
1032 E5          3010 ADJUST PUSH HL
1033 2A C3 13    3020        LD   HL,(OFFSET)  RESD=SORC+OFFSET
1036 19          3030        ADD  HL,DE
1037 EB          3040        EX   DE,HL
1038 E1          3050        POP  HL
1039 F1          3060        POP  AF
103A C9          3070        RET
103B             3080 *
103B             3090 *ROUTINE ACQUIRES ASCII STRING FOR AN
103B             3100 *ADDRESS & CONVERTS IT TO A BINARY NUMBER.
103B             3110 *UPON EXIT, BINARY NUMBER HELD IN HL.
103B             3120 *
103B 22 D3 13    3130 ADDRLD LD   (TEMP),HL
103E CD 9A 10    3140        CALL TVTED
1041 CD 69 10    3150        CALL LINEIN
1044 2B          3160        DEC  HL
1045 CD E2 10    3170        CALL NUMBER
1048 D0          3180        RET  NC
1049 21 7B 13    3190        LD   HL,ERMSG
104C CD 9A 10    3200        CALL TVTED
104F 2A D3 13    3210        LD   HL,(TEMP)
1052 18 E7       3220        JR   ADDRLD
1054             3230 *
1054             3240 *STORES HL AT ADDRESS INDICATED BY
1054             3250 *TBPNTR, AND CHKS FOR NULL LINE.
1054             3260 *
1054 ED 5B BB 13 3270 TABLOD LD   DE,(TBPNTR)
1058 7D          3280        LD   A,L
1059 12          3290        LD   (DE),A
105A 13          3300        INC  DE
105B 7C          3310        LD   A,H
105C 12          3320        LD   (DE),A
105D 13          3330        INC  DE
105E ED 53 BB 13 3340        LD   (TBPNTR),DE
1062 21 9C 13    3350        LD   HL,LINBUF+1
1065 7E          3360        LD   A,(HL)
1066 FE 0D       3370        CP   0DH          CR
1068 C9          3380        RET
1069 21 9C 13    3390 LINEIN LD   HL,LINBUF+1  GET LINE OF
106C E5          3400        PUSH HL           INPUT & STORE
106D C5          3410        PUSH BC           IN LINE BUFFER
106E 06 1F       3420        LD   B,31D
1070 CD BE 10    3430 LNLUP  CALL TVTI         GET CHAR
1073 FE 0D       3440        CP   0DH          CR?
1075 20 03       3450        JR   NZ,BKSPCE    IF SO,END LINE
1077 77          3460        LD   (HL),A
1078 18 1D       3470        JR   LINEND
107A FE 88       3480 BKSPCE CP   88H          CTRL H USED FOR
```

```
107C 28 04        3490          JR   Z,RUBOUT      BACKSPACE
107E FE FF        3500          CP   0FFH          RUBOUT USED
1080 20 08        3510          JR   NZ,STORE      FOR BACKSPACE
1082 2B           3520 RUBOUT   DEC  HL
1083 3E 9B        3530          LD   A,9BH
1085 CD CE 10     3540          CALL TVTO
1088 18 E6        3550          JR   LNLUP
108A 77           3560 STORE    LD   (HL),A
108B 23           3570          INC  HL
108C 10 E2        3580          DJNZ LNLUP
108E 21 F5 12     3590          LD   HL,MSG8       LINE TOO LONG MSG
1091 CD 9A 10     3600          CALL TVTED
1094 C1           3610          POP  BC
1095 18 D2        3620          JR   LINEIN
1097 C1           3630 LINEND   POP  BC
1098 E1           3640          POP  HL
1099 C9           3650          RET
109A              3660 *
109A              3670 *TV EDITOR SCROLLING ROUTINE. 377 CAUSES
109A              3680 *ERASE; A CHAR <200 OUTPUTS THAT
109A              3690 *NUMBER OF SPACES. A ZERO ENDS THE STRING.
109A              3700 *
109A 7E           3710 TVTED    LD   A,(HL)
109B CB 7F        3720          BIT  7,A
109D 28 0E        3730          JR   Z,ZROTST
109F FE FF        3740          CP   0FFH
10A1 20 05        3750          JR   NZ,TVOUT
10A3 CD 00 09     3760          CALL ERASE
10A6 18 0D        3770          JR   HLINC
10A8 CD CE 10     3780 TVOUT    CALL TVTO
10AB 18 08        3790          JR   HLINC
10AD A7           3800 ZROTST   AND  A
10AE C8           3810          RET  Z
10AF 47           3820          LD   B,A
10B0 CD B8 10     3830 SPCLUP   CALL SPACE
10B3 10 FB        3840          DJNZ SPCLUP
10B5 23           3850 HLINC    INC  HL
10B6 18 E2        3860          JR   TVTED
10B8 3E A0        3870 SPACE    LD   A,240
10BA CD CE 10     3880          CALL TVTO
10BD C9           3890          RET
10BE CD A5 09     3900 TVTI     CALL KEYIN
10C1 F6 80        3910          OR   200
10C3 FE FF        3920          CP   377
10C5 28 05        3930          JR   Z,TVTO-2
10C7 FE 8D        3940          CP   215
10C9 28 07        3950          JR   Z,TVTO+4
10CB C9           3960          RET
10CC 3E 9B        3970          LD   A,233
10CE FE 8D        3980 TVTO     CP   215
10D0 20 04        3990          JR   NZ,NTCR
10D2 CD 93 09     4000          CALL TVCR
10D5 C9           4010          RET
10D6 CD 28 09     4020 NTCR     CALL TVOUT1
```

```
10D9 C9            4030          RET
10DA F5            4040 OUTCR    PUSH AF
10DB 3E 8D         4050          LD   A,215
10DD CD CE 10      4060          CALL TVTO
10E0 F1            4070          POP  AF
10E1 C9            4080          RET
10E2               4090 *
10E2               4100 *ROUTINE CONVERTS AN ASCII STRING FOR
10E2               4110 *A HEX OR OCTAL NUMBER TO THE CORRESPONDING
10E2               4120 *BINARY NUMBER. THE ROUTINE EXITS UPON
10E2               4130 *ENCOUNTERING AN INVALID CHAR. MAX BINARY
10E2               4140 *VAL 16 BITS. UPON RET, BIN VAL IS IN HL.
10E2               4150 *
10E2 11 00 00      4160 NUMBER LD   DE,0
10E5 EB            4170          EX   DE,HL
10E6 1A            4180          LD   A,(DE)        OCT OR HEX
10E7 47            4190          LD   B,A
10E8 13            4200          INC  DE
10E9 B7            4210          OR   A
10EA F5            4220          PUSH AF
10EB 78            4230 NXTCHR   LD   A,B
10EC FE 03         4240          CP   3
10EE 1A            4250          LD   A,(DE)
10EF 28 0E         4260          JR   Z,OCTAL
10F1 FE C7         4270 HEX      CP   307           'G'
10F3 30 26         4280          JR   NC,OUTGO
10F5 FE C1         4290          CP   301           'A'
10F7 30 0A         4300          JR   NC,SUBSTP
10F9 FE BA         4310          CP   272           ':'
10FB 30 1E         4320          JR   NC,OUTGO
10FD 18 04         4330          JR   SUBSTP
10FF FE B8         4340 OCTAL    CP   270           '8'
1101 30 18         4350 NXTTST   JR   NC,OUTGO
1103 D6 B0         4360 SUBSTP   SUB  260           '0'
1105 38 14         4370          JR   C,OUTGO
1107 FE 10         4380          CP   10H
1109 38 02         4390          JR   C,LDCA
110B D6 07         4400          SUB  7
110D 4F            4410 LDCA     LD   C,A
110E F1            4420          POP  AF
110F C5            4430          PUSH BC
1110 29            4440 ADDLUP   ADD  HL,HL
1111 10 FD         4450          DJNZ ADDLUP
1113 C1            4460          POP  BC
1114 F5            4470          PUSH AF
1115 79            4480          LD   A,C
1116 85            4490          ADD  L
1117 6F            4500          LD   L,A
1118 13            4510          INC  DE
1119 18 D0         4520          JR   NXTCHR
111B F1            4530 OUTGO    POP  AF
111C FE 03         4540          CP   3
111E 20 03         4550          JR   NZ,VALCHR     VALID CHAR?
1120 7C            4560          LD   A,H
```

```
1121 1F            4570         RRA
1122 67            4580         LD   H,A
1123 1A            4590 VALCHR  LD   A,(DE)
1124 FE B0         4600         CP   0B0H
1126 3F            4610         CCF
1127 C9            4620         RET
1128               4630 ;
1128               4640 ;MEMORY DUMP ROUTINE DISPLAYS OCTAL
1128               4650 ;OR HEX DATA FOR ENTIRE BLOCK. CAN
1128               4660 ;BE STOPPED BY CTRL X IF ENTIRE
1128               4670 ;DUMP IS NOT WANTED
1128               4680 ;
1128 AF            4690 DSPLAY  XOR  A
1129 ED 5B BF 13   4700         LD   DE,(STRESD)
112D 2A C1 13      4710         LD   HL,(NDRESD)
1130 ED 52         4720         SBC  HL,DE
1132 23            4730         INC  HL
1133 EB            4740         EX   DE,HL
1134 E5            4750 DMPLP1  PUSH HL
1135 D5            4760         PUSH DE
1136 ED 5B C5 13   4770         LD   DE,(OFFST1)
113A 19            4780         ADD  HL,DE
113B 7C            4790         LD   A,H
113C CD 6F 11      4800         CALL ASCHAR
113F 7D            4810         LD   A,L
1140 CD 6F 11      4820         CALL ASCHAR
1143 D1            4830         POP  DE
1144 E1            4840         POP  HL
1145 CD B8 10      4850         CALL SPACE
1148 06 06         4860         LD   B,6
114A CD B8 10      4870 DMPLP2  CALL SPACE
114D 7E            4880         LD   A,M
114E CD 6F 11      4890         CALL ASCHAR
1151 1B            4900         DEC  DE
1152 7B            4910         LD   A,E
1153 B2            4920         OR   D
1154 28 11         4930         JR   Z,DSPLDN
1156 23            4940 NXDSPL  INC  HL
1157 10 F1         4950         DJNZ DMPLP2
1159 DB 00         4960         IN   0
115B FE 98         4970         CP   98H          CTRL X
115D CA B9 0F      4980         JP   Z,MOVE
1160 3E 8D         4990         LD   A,215
1162 CD CE 10      5000         CALL TVTO
1165 18 CD         5010         JR   DMPLP1
1167 3E 8D         5020 DSPLDN  LD   A,215
1169 CD CE 10      5030         CALL TVTO
116C C3 B9 0F      5040         JP   MOVE
116F               5050 ;
116F               5060 ;CONVERTS BYTE IN ACC TO ITS
116F               5070 ;OCTAL OR HEX CHARS & DISPLAYS
116F               5080 ;ON TV.
116F               5090 ;
116F C5            5100 ASCHAR  PUSH BC
```

```
1170 F5            5110         PUSH AF
1171 3A 9B 13      5120         LD   A,(LINBUF)
1174 FE 03         5130         CP   3
1176 28 13         5140         JR   Z,OCTNUM
1178 CD B8 10      5150         CALL SPACE
117B F1            5160         POP  AF
117C 4F            5170         LD   C,A
117D 0F            5180         RRCA
117E 0F            5190         RRCA
117F 0F            5200         RRCA
1180 0F            5210         RRCA
1181 E6 0F         5220         AND  0FH
1183 CD A9 11      5230         CALL ASCCON
1186 79            5240         LD   A,C
1187 E6 0F         5250         AND  0FH
1189 18 19         5260         JR   OUT1
118B F1            5270 OCTNUM  POP  AF
118C 07            5280         RLCA
118D 07            5290         RLCA
118E 4F            5300         LD   C,A
118F E6 03         5310         AND  3
1191 CD A9 11      5320         CALL ASCCON
1194 79            5330         LD   A,C
1195 07            5340         RLCA
1196 07            5350         RLCA
1197 07            5360         RLCA
1198 4F            5370         LD   C,A
1199 E6 07         5380         AND  7
119B CD A9 11      5390         CALL ASCCON
119E 79            5400         LD   A,C
119F 07            5410         RLCA
11A0 07            5420         RLCA
11A1 07            5430         RLCA
11A2 E6 07         5440         AND  7
11A4 CD A9 11      5450 OUT1    CALL ASCCON
11A7 C1            5460         POP  BC
11A8 C9            5470         RET
11A9 FE 0A         5480 ASCCON  CP   10D
11AB 38 06         5490         JR   C,LT10
11AD D6 09         5500         SUB  011
11AF F6 C0         5510         OR   300
11B1 18 02         5520         JR   OUTPUT
11B3 F6 B0         5530 LT10    OR   260
11B5 CD CE 10      5540 OUTPUT  CALL TVTO
11B8 C9            5550         RET
11B9               5560 *
11B9               5570 *ROUTINE READS TAPE INTO MEMORY
11B9               5580 *BEGIN. AT STRESD. READS UNTIL
11B9               5590 *NO STRT BIT IS RECD WITHIN TDELAY
11B9               5600 *TIME, THEN LDS NDRESD & RETURNS.
11B9               5610 *
11B9 CD 21 10      5620 RDTAPE CALL KYBDST
11BC ED 5B BF 13   5630         LD   DE,(STRESD)
11C0 DB 01         5640 STRTBT IN   1
```

```
11C2 E6 01          5650          AND  1
11C4 20 FA          5660          JR   NZ,STRTBT
11C6 06 00          5670 TDELAY LD   B,0
11C8 05             5680          DEC  B
11C9 28 1E          5690          JR   Z,LODEND
11CB DB 01          5700          IN   1
11CD E6 01          5710          AND  1
11CF 20 F7          5720          JR   NZ,TDELAY+2
11D1 CD 9C 00       5730          CALL BYTRED
11D4 7C             5740          LD   A,H
11D5 12             5750          LD   (DE),A
11D6 1A             5760          LD   A,(DE)
11D7 BC             5770          CP   H
11D8 28 04          5780          JR   Z,MEMOK
11DA 3E AE          5790          LD   A,'.'
11DC 18 05          5800          JR   TVCALL
11DE 7A             5810 MEMOK  LD   A,D
11DF E6 07          5820          AND  7
11E1 C6 B0          5830          ADD  260
11E3 CD FA 00       5840 TVCALL CALL TV
11E6 13             5850          INC  DE
11E7 18 DD          5860          JR   TDELAY
11E9 1B             5870 LODEND DEC  DE
11EA ED 53 C1 13    5880          LD   (MDRESD),DE
11EE C9             5890          RET
11EF                5900 *
11EF                5910 *TABLE OF 1ST BYTE OF 3-BYTE INSTRUCTIONS
11EF                5920 *
11EF                5930 TABLE3 DC   01H,11H,21H,22H,2AH
  01   11   21   22   2A

11F4                5940          DC   31H,32H,3AH,0C2H,0C3H
  31   32   3A   C2   C3

11F9                5950          DC   0C4H,0CAH,0CCH,0CDH,0D2H
  C4   CA   CC   CD   D2

11FE                5960          DC   0D4H,0DAH,0DCH,0E2H,0E4H
  D4   DA   DC   E2   E4

1203                5970          DC   0EAH,0ECH,0F2H,0F4H,0FAH
  EA   EC   F2   F4   FA

1208                5980          DB   0FCH
  FC
1209                5990 *
1209                6000 *TABLE OF 1ST BYTE OF 2-BYTE INSTRUCTIONS
1209                6010 *
1209                6020 TABLE2 DC   06H,0EH,10H,16H,18H
  06   0E   10   16   18

120E                6030          DC   1EH,20H,26H,28H,2EH
  1E   20   26   28   2E
```

```
1213                6040          DC    30H,36H,38H,3EH,0C6H
  30   36   38   3E   C6

1218                6050          DC    0CBH,0CEH,0D3H,0D6H,0DBH
  CB   CE   D3   D6   DB

121D                6060          DC    0DEH,0E6H,0EEH,0F6H,0FEH
  DE   E6   EE   F6   FE

1222                6070 *
1222                6080 *TABLE OF 2ND BYTE OF DD OR FD INSTRUCTION
1222                6090 *
1222                6100 STBLDF DC    21H,22H,2AH,36H,0CBH
  21   22   2A   36   CB

1227                6110          DC    09H,19H,23H,29H,2BH
  09   19   23   29   2B

122C                6120          DC    39H,0E1H,0E8H,0E9H,0F9H
  39   E1   E8   E9   F9

1231                6130 *
1231                6140 *TABLE OF 2ND BYTE OF ED INSTRUCTIONS
1231                6150 *
1231                6160 STBLED DC    43H,4BH,53H,5BH,73H
  43   4B   53   5B   73

1236                6170          DB    7BH
  7B
1237                6180 MSG1     DC    377,6
  FF   06
1239                6190          DB    'Z-80 RELOCATE PROGRAM'
  DA   AD   B8   B0   A0
  D2   C5   CC   CF   C3
  C1   D4   C5   A0   D0
  D2   CF   C7   D2   C1
  CD
124E                6200          DC    8DH,8DH
  8D   8D
1250                6210          DB    'ENTER TASK NUMBER: '
  C5   CE   D4   C5   D2
  A0   D4   C1   D3   CB
  A0   CE   D5   CD   C2
  C5   D2   BA   A0
1263                6220          DC    8DH,12D
  8D   0C
1265                6230          DB    '0=MOVE ONLY'
  B0   BD   CD   CF   D6
  C5   A0   CF   CE   CC
  D9
1270                6240          DC    8DH,12D
  8D   0C
1272                6250          DB    '1=ADJUST REFS ONLY'
  B1   BD   C1   C4   CA
```

```
  D5   D3   D4   A0   D2
  C5   C6   D3   A0   CF
  CE   CC   D9
1284              6260        DC   8DH,12D
  8D   0C
1286              6270        DB   '2=MOVE & ADJUST REFS'
  B2   BD   CD   CF   D6
  C5   A0   A6   A0   C1
  C4   CA   D5   D3   D4
  A0   D2   C5   C6   D3

129A              6280        DC   8DH,0
  8D   00
129C              6290 MSG2   DB   'ENTER ADDRESS MODE (D/H) '
  C5   CE   D4   C5   D2
  A0   C1   C4   C4   D2
  C5   D3   D3   A0   CD
  CF   C4   C5   A0   A8
  CF   AF   C8   A9   A0

12B5              6300        DB   0
  00
12B6              6310 MSG3   DB   'SOURCE START = '
  D3   CF   D5   D2   C3
  C5   A0   D3   D4   C1
  D2   D4   A0   BD   A0

12C5              6320        DB   0
  00
12C6              6330 MSG5   DB   8DH
  8D
12C7              6340        DB   'DEST START   = '
  C4   C5   D3   D4   A0
  D3   D4   C1   D2   D4
  A0   A0   A0   BD   A0

12D6              6350        DB   0
  00
12D7              6360 MSG6   DB   'STADDR      = '
  D3   D4   C1   C4   C4
  D2   A0   A0   A0   A0
  A0   A0   BD   A0
12E5              6370        DB   0
  00
12E6              6380 MSG7   DB   'NDADDR      = '
  CE   C4   C1   C4   C4
  D2   A0   A0   A0   A0
  A0   A0   BD   A0
12F4              6390        DB   0
  00
12F5              6400 MSG8   DB   8DH
  8D
12F6              6410        DB   'LINE TOO LONG--RE-ENTER'
  CC   C9   CE   C5   A0
```

```
  D4   CF   CF   A0   CC
  CF   CE   C7   AD   AD
  D2   C5   AD   C5   CE
  D4   C5   D2
130D                6420         DC   8DH,0
  8D   00
130F                6430 MSGTAP DB   'INPUT FROM TAPE? (Y/N) '
  C9   CE   D0   D5   D4
  A0   C6   D2   CF   CD
  A0   D4   C1   D0   C5
  BF   A0   A8   D9   AF
  CE   A9   A0
1326                6440         DB   0
  00
1327                6450 STMSG  DB   'START CASSETTE, THEN PRESS '
  D3   D4   C1   D2   D4
  A0   C3   C1   D3   D3
  C5   D4   D4   C5   AC
  A0   D4   C8   C5   CE
  A0   D0   D2   C5   D3
  D3   A0
1342                6460         DB   'ANY KEY'
  C1   CE   D9   A0   CB
  C5   D9
1349                6470         DC   8DH,0
  8D   00
134B                6480 TAPMSG DB   8DH
  8D
134C                6490         DB   'DO YOU WISH TO WRITE NEW '
  C4   CF   A0   D9   CF
  D5   A0   D7   C9   D3
  C8   A0   D4   CF   A0
  D7   D2   C9   D4   C5
  A0   CE   C5   D7   A0

1365                6500         DB   'TAPE? (Y/N) '
  D4   C1   D0   C5   BF
  A0   A8   D9   AF   CE
  A9   A0
1371                6510         DB   0
  00
1372                6520 WRTMSG DB   'WRITING'
  D7   D2   C9   D4   C9
  CE   C7
1379                6530         DC   8DH,0
  8D   00
137B                6540 ERMSG  DB   'ADDRESS ERROR--PLEASE RE-ENTER'
  C1   C4   C4   D2   C5
  D3   D3   A0   C5   D2
  D2   CF   D2   AD   AD
  D0   CC   C5   C1   D3
  C5   A0   D2   C5   AD
  C5   CE   D4   C5   D2
```

```
1399                6550         DC    8DH,0
  8D    00
139B                6560 LINBUF  DS    32D
13BB                6570 TBPNTR  DS    2
13BD                6580 TASK    DS    1
13BE                6590 PAIRS   DS    1
13BF                6600 STRESD  DS    2
13C1                6610 NDRESD  DS    2
13C3                6620 OFFSET  DS    2
13C5                6630 OFFST1  DS    2
13C7                6640 BYTCNT  DS    2
13C9                6650 STSORC  DS    2
13CB                6660 NDSORC  DS    2
13CD                6670 STDEST  DS    2
13CF                6680 NDDEST  DS    2
13D1                6690 DISP    DS    2
13D3                6700 TEMP    DS    2
13D5                6710 STRT1   DS    8D

NO ERRORS FOUND

FILE  3000  674C
READY
```