SCREEN CRUNCHER

HI-RES GRAPHICS

scrunch /skur-runch/ (*the sound of a hi-res bug being squished*) To reduce a hi-res picture to as few bytes as possible so that it can be unscrunched.

unscrunch /un-skur-runch/ (*the sound of that same bug being unsquished*) To return a scrunched picture to its original pattern. See Scrunch.

REQUIREMENTS:
48K Apple II (or Franklin Ace)
One Disk Drive

Saving a hi-res picture usually requires 34 sectors of disk space. That allows only about 14 pictures to be saved to the disk. Scruncher 1.Ø, a machine-language utility, can usually more than double the number of pictures (28 to 4Ø, in fact!) per disk. It will also quickly "un-scrunch" the picture so that it can be displayed normally.

How To Scrunch

There are two distinctly different techniques used to reduce the amount of space required to store a picture.

One method saves only the commands used to draw the picture. An example would be: draw a circle at 9Ø,9Ø with a radius of 2Ø and color it in with green. The picture (a green circle on the hi-res screen) is not saved as a finished product, but as a series of commands telling another program how to draw a picture. This method, used in many hi-res adventure games such as Wizard and the Princess, can reduce a picture by up to 9Ø%.

Unfortunately, it requires:

1. a special editing program
2. drawing the picture in the fewest number of commands.

Another way is to condense the completed picture. It involves scanning the entire picture to look for "repeaters". This allows you to shrink any drawing you have made previously, and to use any one of the excellent drawing programs now available to draw the picture. It is possible to achieve savings of 3Ø% to 9Ø%, with an average of 46%.

I find this method more suitable as it allows me to use free-form when drawing a picture, whereas the first method restricts me from freely editing and changing my finished picture without an excessive amount of effort.

To actually reduce the amount of storage space (on disk or in memory) required by a hi-res picture, the picture must be encoded. Since the hi-res screen is nothing more than an array of 4Ø bytes across and 192 bytes down, the best way is to encode all the values that are repeated . . .

For example, in illustration 1 the values $ØØ and $3F are both consecutively repeating bytes:

$ØØ occurs twice in a row.
$3F occurs 4 times in a row.

Only $3F is a true repeater.

To encode a picture, 3 decisions must be made:

1. Which way to examine the picture data.
2. How to code in the least number of bytes.
3. What value to use as a marker byte.

1. Scanning the Picture

The number of repeaters that can be found is affected by how the picture is examined. There are at least three ways to look for repetitious values:

A. Sequentially through memory.
B. Horizontally through the picture (as it appears on the screen).
C. Vertically through the picture.

There is a difference between how the screen appears and how it's formatted in

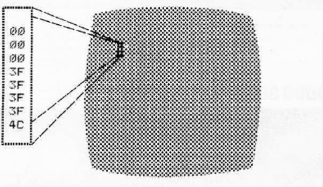


Illustration 1

Take a small part of the hi-res screen and look for values that repeat themselves consecutively. "Repeaters" are values (bytes) that are consecutively repeated 4 or more times. Other repeating bytes are not called repeaters.

memory.

To scan consecutively through memory would be inefficient because the hi-res screen is not oriented as consecutive bytes in memory. And since a hi-res picture is usually not a set of random values, no pattern on the screen would be easily coded unless it was examined in the order it appeared on the screen, not as it appears in memory.

The hi-res screen is only 40 bytes wide, so any value can be repeated horizontally only 40 times.

However, it's 192 bytes tall. Therefore, a value can be repeated up to 192 times. The vertical scanning method obviously provides a greater chance of finding a larger number of repeaters.

2. Coding the Repeaters

Now that a search method has been selected, it's time to create a coding method that uses the least number of bytes.

Scruncher uses three bytes to encode repeaters: a Code Marker, a Counter, and the Repeater.

The Code Marker informs the Unscruncher that encoded information is coming, much like the address marks used by DOS let it know data is coming. The second byte is a Counter that tells how many times to repeat the third byte, which is the actual repeat value that is encoded.

Illustration 2

FE 34 22

FE — the Marker Byte.
34 — Number of times to repeat.
22 — Byte to repeat.

In illustration 2, the three bytes are shown in their relationship to other unencoded bytes.

Because encoding takes three bytes, a repeater must be repeated at least 4 times consecutively. No space is saved when the repetition is less than four, and for every repetition greater than three, another byte of space is saved. If a value is repeated down the height of the picture, 189 bytes are saved (192 - 3 = 189).

3. Selecting the Marker

If any byte's value can be a part of the hi-res picture, what value can be used to indicate a coded sequence?

There is only one criterion used to select the best possible marker value: the number of times it is found in the hi-res picture.

The fewer times the byte value is found, the better that value will work. The reason for this is that every time a value is found which is the same as the repeat marker, it must be encoded, even

DEMO

```
10  NORMAL : TEXT : HOME
20 D$ =  CHR$ (4)
30  REM
```

RELOCATE?

```
40  IF  PEEK (103) = 1 AND  PEEK
     (104) = 96 THEN 60
50  POKE 103,1: POKE 104,96: POKE
     24576,0: PRINT D$"RUN DEMO"
60  NORMAL : TEXT : HOME : POKE 2
     30,32: POKE  - 16304,0: POKE
      - 16300,0: POKE  - 16297,0:
      POKE  - 16301,0
70  IF  PEEK (8 * 256 + 3) = 169 AND
      PEEK (8 * 256 + 4) = 255 THEN
     110
80  VTAB 22: PRINT "PLEASE WAIT W
     HILE I LOAD THE FILES"
90  PRINT D$"BLOAD PACK,A$803": PRINT
     D$"BLOAD UN-PACK,A$300"
100  REM
WHERE END OF COMPRESSED
     PICTURE IS
110 LO = 8 * 256 + 15 * 16 + 12
120  HOME : POKE  - 16304,0: VTAB
     22: PRINT "COMPRESS/DECOMPRE
     SS (C/D)?";: GET A$: PRINT
130  IF A$ <  > "C" AND A$ <  > "
     D" THEN  PRINT "ILLEGAL ENTR
     Y": GOTO 120
140  IF A$ = "D" THEN 360
150  REM
```

COMPRESS OPTION

```
160  VTAB 24: PRINT "ENTER NAME O
     F HI-RES PICTURE TO COMPRESS
     "
170  HTAB 5: INPUT "> ";NA$
180  IF  LEFT$ (NA$,1) = D$ THEN
      TEXT : HOME : PRINT NA$: GET
     A$: HOME : POKE  - 16304,0: GOTO
     160
190  IF NA$ = "" THEN 220
200  PRINT D$"BLOAD"NA$",A$2000"
210  REM
COMPRESS PICTURE
220  CALL 8 * 256 + 3
230 LE =  PEEK (LO) +  PEEK (LO +
     1) * 256 - 16384
240  PRINT "LENGTH OF COMPRESSED
     PICTURE:"LE
250  PRINT "NUMBER OF BYTES SAVED
     :"8192 - LE
260  PRINT "PERCENTAGE DIFFERENCE
     "100 -  INT (LE / 8192 * 100
     )"%"
270  PRINT "BSAVE THIS COMPRESSIO
     N (Y/N)? ";: GET A$: PRINT
280  IF A$ <  > "Y" THEN  HOME : GOTO
     120
290  PRINT "UNDER WHAT NAME ('.C'
      IS APPENDED)"
300  HTAB 5: INPUT "> ";NA$
310  IF  LEFT$ (NA$,1) = D$ THEN
      TEXT : HOME : PRINT NA$: GET
     A$: POKE  - 16304,0: HOME : VTAB
     22: GOTO 290
320  IF NA$ = "" THEN  HOME : GOTO
     120
330  PRINT D$"BSAVE"NA$".C,A$4000
     ,L"LE
340  GOTO 120
350  REM
```

DECOMPRESS OPTION

```
360  PRINT "COMPRESSED PICTURE ('
     .C' IS APPENDED)"
370  HTAB 5: INPUT "> ";NA$
380  IF  LEFT$ (NA$,1) = D$ THEN
      TEXT : HOME : PRINT NA$: GET
     A$: POKE  - 16304,0: HOME : VTAB
     22: GOTO 360
390  IF NA$ = "" THEN 410
400  PRINT D$"BLOAD"NA$".C,A$4000
     "
410  CALL 3 * 256: REM  UNPACK PI
     CTURE
420  GOTO 120
```

if it is found only once (increasing the code instead of decreasing it!).

If this value was not encoded, there would be no way to tell the difference between a marker and a byte with the value of the marker, since they both are the same.

Illustration 3

FE Ø1 FE

FE — the Marker Byte.
Ø1 — Number of times to repeat the marker
FE — the Marker Byte.

Illustration 3 shows this. Imagine the illustration as a section of the encoded picture, and $FE as the selected marker byte. The decoding program would find the $FE, and assuming it to be a marker, would repeat the value $22, 54 ($34) times. The program has no way of knowing that this byte was not a marker for a repeater. So to encode the information properly it becomes necessary to encode all values found that are the same as the marker byte.

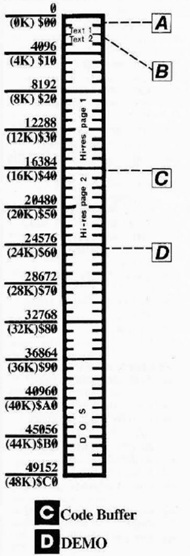
To solve the problem of selecting the best marker value, Scruncher searches the entire hi-res picture looking for the value that is found the least number of times. This value is then used as the marker byte and is stored as the first value in the code buffer. Unscruncher then looks at this first value and uses it as the code marker, allowing each picture to have the best possible value as its code marker.

How Scruncher Works

Scruncher examines the hi-res picture vertically just as it appears on the screen. It checks each byte to see if it is the value of the marker or if it has been encountered four times in a row. If either condition is true, then coding takes place and the code is moved to the code buffer. If both conditions are false, then it stores that unaltered value in the code buffer. This process continues until all 40 columns are transferred.

If no value is repeated more than three times, there are still 512 bytes (two sectors or one-half of a kilobyte) saved because the program ignores the presence of "hidden" bytes on the hi-res screen.

But nearly all pictures have values that occur more than three times. Generally, a saving of 20% or better occurs.



C Code Buffer

D DEMO

Hex Dump for UN-PACK

A

```
Ø3ØØ- A2 ØØ AØ 4Ø 8C 3Ø Ø3 AØ
Ø3Ø8- ØØ 8C 2F Ø3 2Ø 2E Ø3 85
Ø31Ø- FE 2Ø 2E Ø3 C5 FE FØ Ø5
Ø318- 2Ø 3A Ø3 9Ø F4 2Ø 2E Ø3
Ø32Ø- 85 ØØ 2Ø 2E Ø3 2Ø 3A Ø3
Ø328- C6 ØØ DØ F9 FØ E3 AD FF
Ø33Ø- FF EE 2F Ø3 DØ Ø3 EE 3Ø
Ø338- Ø3 6Ø 48 98 48 29 CØ 8D
Ø34Ø- 68 Ø3 4A 4A ØD 68 Ø3 8D
Ø348- 68 Ø3 68 8D 69 Ø3 ØA ØA
Ø35Ø- ØA 2E 69 Ø3 ØA 2E 69 Ø3
Ø358- ØA 6E 68 Ø3 AD 69 Ø3 29
Ø36Ø- 1F Ø9 2Ø 8D 69 Ø3 68 9D
Ø368- FF FF C8 CØ CØ 9Ø Ø7 AØ
Ø37Ø- ØØ E8 EØ 28 BØ Ø1 6Ø 68
Ø378- 68 6Ø
```

Hex Dump for PACK

B

```
Ø8Ø3- A9 FF 8D Ø9 Ø9
Ø8Ø8- A9 ØØ 8D Ø7 Ø9 85 FE A9
Ø81Ø- ØØ 8D Ø8 Ø9 8D 1D Ø8 A9
Ø818- 2Ø 8D 1E Ø8 AD FF FF CD
Ø82Ø- Ø7 Ø9 DØ Ø5 EE Ø8 Ø9 FØ
Ø828- 29 EE 1D Ø8 DØ EE EE 1E
Ø83Ø- Ø8 AD 1E Ø8 C9 4Ø DØ E4
Ø838- AD Ø8 Ø9 DØ Ø8 AD Ø7 Ø9
Ø84Ø- 85 FE 4C 59 Ø8 CD Ø9 Ø9
Ø848- BØ Ø8 8D Ø9 Ø9 AD Ø7 Ø9
Ø85Ø- 85 FE CE Ø7 Ø9 DØ B8 A5
Ø858- FE BD ØØ 4Ø A2 ØØ 86 ØØ
Ø86Ø- 86 Ø1 AØ 4Ø 8C FD Ø8 AØ
Ø868- ØØ 8C A2 Ø8 A9 Ø1 8D FC
Ø87Ø- Ø8 A9 2Ø 8D A3 Ø8 98 48
Ø878- 29 CØ 8D A2 Ø8 4A 4A ØD
Ø88Ø- A2 Ø8 8D A2 Ø8 68 8D A3
```

```
Ø888- Ø8 ØA ØA ØA 2E A3 Ø8 ØA
Ø89Ø- 2E A3 Ø8 ØA 6E A2 Ø8 AD
Ø898- A3 Ø8 29 1F Ø9 2Ø 8D A3
Ø8AØ- Ø8 BD FF FF 24 Ø1 3Ø ØB
Ø8A8- 85 Ø3 A9 8Ø 85 Ø1 3Ø Ø7
Ø8BØ- C5 Ø3 FØ Ø3 2Ø CC Ø8 E6
Ø8B8- ØØ C8 CØ CØ 9Ø B8 2Ø CC
Ø8CØ- Ø8 E8 EØ 28 FØ 26 AØ ØØ
Ø8C8- 84 Ø1 FØ AA 48 84 Ø4 A4
Ø8DØ- ØØ CØ Ø4 9Ø 18 A5 FE 2Ø
Ø8D8- F8 Ø8 98 2Ø FB Ø8 A5 Ø3
Ø8EØ- 2Ø FB Ø8 A9 ØØ 85 ØØ 68
Ø8E8- A4 Ø4 85 Ø3 6Ø A5 Ø3 C5
Ø8FØ- FE FØ E2 2Ø FB Ø8 88 DØ
Ø8F8- FA FØ E8 8D FF FF EE FC
Ø9ØØ- Ø8 DØ Ø3 EE FD Ø8 6Ø
```

Entering the Scruncher

There are three sections in the Scruncher program. Two are machine language programs (PACK and UN-PACK), and the last one is in BASIC. The demo allows you to load a picture into memory and scrunch (using PACK) or un-scrunch it (using UN-PACK).

Directions for Entering PACK

1) Enter the monitor.

CALL-151

2) Type in the hex-dump for PACK.

3) Save PACK to the disk. Do not return to BASIC. BSAVE can be done from the monitor. Just . . .

BSAVE PACK,A$803,L$106

Directions for Entering UN-PACK

1) Type in the hex-dump for UN-PACK.

2) Save UN-PACK to the disk. (You can do this from the monitor or from BASIC.)

BSAVE UN-PACK,A$300,L$

3) Return to BASIC. (If you haven't done so already.)

3D0G

Directions for Entering DEMO

1) Reset Applesoft pointers.

FP

2) Type the DEMO Applesoft listing.

3) Save the program.

SAVE DEMO

To use the program, simply RUN DEMO. It will first relocate itself (more on that later), and then load the programs PACK and UN-PACK. You will notice that the hi-res screen is now displayed.

At this time you will be asked either to "Compress" or "Decompress" a picture. The "Compress" option will PACK (encode) the picture so that it takes up less room. "Decompress" will UN-PACK (decode) a compressed picture.

Type either C or D.

If you decide to compress a picture, you will be prompted to enter the picture's name.

If you simply press RETURN, the current picture (as shown on the hi-res page) will be compressed.

PACK

```
1000 *--------------------------------
1010 * HI-RES PICTURE PACKER PROGRAM
1020 *
1030 *              BY
1040 *
1050 *          ROBB CANFIELD
1060 *
1070 *          NOV. 15 1982
1080 *
1090 *--------------------------------
1100
1110
1120
1130 COUNTER  .EQ $00      NUMBER OF TIMES TO REPEAT
1140 FIRST.TIME.RAN .EQ $01 ($00 MEANS FIRST RUN)
1150 TABLE    .EQ $3       LOCATION OF BYTE TO REPEAT
1160 YSAVE    .EQ $4       Y-REG SAVE AREA
1170
1180 REP.CHAR .EQ $FE      MARKER CHARACTER
1190
1200
1210          .OR $803
1220          .TF PACK
1230
1250
1260
1270 *--------------------------------
1280 * FIND BEST REPEAT BYTE, BY
1290 * SEARCHING THRU THE HIRES SCREEN
1300 *--------------------------------
1310
1320
1330 SEARCH
1340          LDA #$FF     RESET LAST.COUNT
1350          STA LAST.COUNT
1360          LDA #$00     RESET SEARCH POINTERS
1370          STA CURRENT
1380          STA REP.CHAR
1390 .0
1400          LDA #$00
1410          STA CURRENT.COUNT
1420          STA .1+1
1430          LDA #$20
1440          STA .1+2
1450
1460 .1       LDA $FFFF    GET A BYTE FROM HIRES SCREEN
1470          CMP CURRENT  SAME AS REPEAT BYTE?
1480          BNE .2       NO SO CONTINUE
1490          INC CURRENT.COUNT
1500          BEQ .3
1510 .2       INC .1+1     INCREMENT ADDRESS
1520          BNE .1
1530          INC .1+2
1540          LDA .1+2     IS IT 40?
1550          CMP #$40
1560          BNE .1
1570          LDA CURRENT.COUNT GET BEST REPEATE VALUE
1580          BNE .5
1590          LDA CURRENT  GET BYTE
1600          STA REP.CHAR
1610          JMP .4
1620 .5       CMP LAST.COUNT
1630          BGE .3
1640          STA LAST.COUNT
1650          LDA CURRENT  SAVE REPEAT BYTE (NEW ONE)
1660          STA REP.CHAR
1670
1680 .3       DEC CURRENT  GET NEXT BYTE TO CHECK
1690          BNE .0
1700          LDA REP.CHAR SAVE REPEAT BYTE IN BUFFER
1710 .4       STA $4000
1720
1730
1740
1750 *--------------------------------
1760 * START TO COMPRESS PICTURE.
1770 *--------------------------------
1780
1790
1800
1810          LDX #$00     RESET HORIZONTAL OFFSET
1820          STX COUNTER  RESET REP COUNT
```

Screen Cruncher

continued from page 25

Ctrl-D allows you to enter a DOS command such as CATALOG (just type: ctrl-D CATALOG).

Otherwise, the text you enter will be used as the name of a file you wish to load. Once the file is loaded, it will be displayed and compressed. You will see the results of the compression at the bottom of the screen:

1) How many bytes long the compressed picture is.
2) How many bytes were saved.
3) The percentage of savings.

Then you will be asked if you wish to save the compressed version. If you do, press Y for yes and enter its name (again, you may use ctrl-D to enter a DOS command). The suffix .C will be appended automatically to the name of the compressed version of the picture to distinguish it from the original file.

If return is pressed, you will exit this routine without saving the file.

Any other key means "no".

When the option DECOMPRESS (D) is used, you will be asked to enter the name of the compressed file (.C is automatically appended, so do not enter it as part of the file name).

Again, ctrl-D allows you to enter a DOS command.

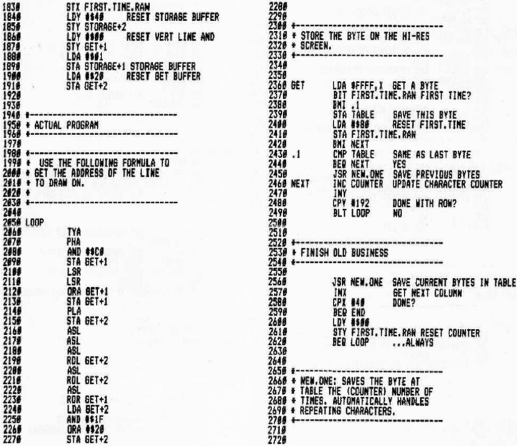
Pressing RETURN will decompress the currently compressed picture in memory (if there is none, garbage will appear on the screen).

When a picture is decompressed it will appear on the hi-res screen as it is decoded, producing a rather nice scrolling effect from left to right.

To use the UN-PACKer in your own programs, first BLOAD UN-PACK. Then BLOAD the compressed picture, which is normally located on page two of hi-res ($4000). To decompress the picture, enter a CALL 768. By examining the source code you can determine how to load the coded information anywhere, and can control where the decoded picture will be drawn (normally on page one of hi-res, $2000).

DEMO consists of three major sections:

PACK



```
1830          STX FIRST.TIME.RAM
1840          LDY #$40     RESET STORAGE BUFFER
1850          STY STORAGE+2
1860          LDY #$00     RESET VERT LINE AND
1870          STY GET+1
1880          LDA #$01
1890          STA STORAGE+1 STORAGE BUFFER
1900          LDA #$20     RESET BET BUFFER
1910          STA GET+2
1920
1930
1940 *-------------------------------
1950 * ACTUAL PROGRAM
1960 *-------------------------------
1970
1980 *-------------------------------
1990 *  USE THE FOLLOWING FORMULA TO
2000 * GET THE ADDRESS OF THE LINE
2010 * TO DRAW ON.
2020 *
2030 *-------------------------------
2040
2050 LOOP
2060          TYA
2070          PHA
2080          AND #$C0
2090          STA GET+1
2100          LSR
2110          LSR
2120          ORA BET+1
2130          STA GET+1
2140          PLA
2150          STA GET+2
2160          ASL
2170          ASL
2180          ASL
2190          ROL GET+2
2200          ASL
2210          ROL GET+2
2220          ASL
2230          ROR GET+1
2240          LDA BET+2
2250          AND #$1F
2260          ORA #$20
2270          STA GET+2
```

```
2280
2290
2300 *-------------------------------
2310 * STORE THE BYTE ON THE HI-RES
2320 * SCREEN.
2330 *-------------------------------
2340
2350
2360 GET      LDA $FFFF,X  GET A BYTE
2370          BIT FIRST.TIME.RAN FIRST TIME?
2380          BMI .1
2390          STA TABLE    SAVE THIS BYTE
2400          LDA #$80     RESET FIRST.TIME
2410          STA FIRST.TIME.RAN
2420          BMI NEXT
2430 .1       CMP TABLE    SAME AS LAST BYTE
2440          BEQ NEXT     YES
2450          JSR NEW.ONE  SAVE PREVIOUS BYTES
2460 NEXT     INC COUNTER  UPDATE CHARACTER COUNTER
2470          INY
2480          CPY #192     DONE WITH ROW?
2490          BLT LOOP     NO
2500
2510
2520 *-------------------------------
2530 * FINISH OLD BUSINESS
2540 *-------------------------------
2550
2560          JSR NEW.ONE  SAVE CURRENT BYTES IN TABLE
2570          INX          GET NEXT COLUMN
2580          CPX #40      DONE?
2590          BEQ END
2600          LDY #$00
2610          STY FIRST.TIME.RAN RESET COUNTER
2620          BEQ LOOP     ...ALWAYS
2630
2640
2650 *-------------------------------
2660 * NEW.ONE: SAVES THE BYTE AT
2670 * TABLE THE (COUNTER) NUMBER OF
2680 * TIMES. AUTOMATICALLY HANDLES
2690 * REPEATING CHARACTERS.
2700 *-------------------------------
2710
2720
```

I. Relocate itself and load the files.
II. Compress a picture.
III. Decompress a picture.

The first part (I) checks to see if the Applesoft pointers are pointed at $4000 (normally they point at $801). If not, they are modified and the program is re-RUN. Next, the program checks to see if the required files (PACK and UN-PACK) are loaded. If not, they are loaded for you automatically.

The second part (II) will compress a picture by getting the picture's name and calling the Compress Routine ($803, CALL 2651).

The third section (III) will decompress a picture ($300, CALL 768).

```
2730 NEW.ONE  PHA            SAVE CURRENT.BYTE
2740          STY YSAVE      SAVE Y-REG
2750          LDY COUNTER
2760          CPY #$4        USE REPEAT CHARACTER
2770          BLT NO.REPEAT
2780 REPEAT   LDA REP.CHAR GET REPEAT CHARACTER
2790          JSR STORAGE    AND SAVE IT
2800          TYA            NO. OF TIMES TO REPEATE IN ACCU.
2810          JSR STORAGE    SAVE IT
2820          LDA TABLE      GET CHARACTER TO REPEAT
2830          JSR STORAGE    SAVE IT
2840 EXIT     LDA #$00       RESET COUNTER
2850          STA COUNTER
2860          PLA            RETRIEVE ACCUM.
2870          LDY YSAVE      AND Y-REG
2880          STA TABLE      SAVE CURRENT BYTE IN THE TABLE
2890 END      RTS
2900
2910
2920 NO.REPEAT LDA TABLE     GET BYTE TO REPEAT
2930          CMP REP.CHAR IS IT THE REPEAT CHARACTER
2940          BEQ REPEAT     YES, HANDLE REPEATING CHAR.
2950 .1       JSR STORAGE    AND REPEAT IT Y TIMES
2960          DEY            DONE?
2970          BNE .1         NO
2980          BEQ EXIT       YES SO EXIT
2990
3000
3010
3020 *---------------------------------
3030 * STORAGE: SAVES THE ACCUM. AT
3040 * STORAGE AREA AND INCREMENTS
3050 * THIS VALUE
3060 *---------------------------------
3070
3080 STORAGE
3090          STA $FFFF      ADDRESS TO STORE INFORMATION
3100          INC STORAGE+1 INCREMENT THIS ADDRESS
3110          BNE .1         CARRY?
3120          INC STORAGE+2 YES, SO INCREMENT HIGH BYTE
3130 .1       RTS            RETURN TO CALLER
3140
3150
3160
3170
3180 CURRENT  .BS 1
3190 CURRENT.COUNT .BS 1
3200 LAST.COUNT .BS 1
```

UN-PACK

```
1000 *---------------------------------
1010 * HIRES PICTURE UN-PACKER PROGRAM
1020 *
1030 *               BY
1040 *
1050 *          ROBB CANFIELD
1060 *
1070 *         COPYRIGHTED 1983
1080 *       BY SOFTKEY PUBLISHING
1090 *
1100 *---------------------------------
1110
1120
1130
1140 YSAVE    .EQ $1         Y-REG SAVE AREA
1150 COUNTER  .EQ $00        COUNTER FOR REPEATING
1160
1170 REP.CHAR .EQ $FE        MARKER BYTE
1180
1190
1200          .OR $300
1210          .TF UN-PACK
1220
1230
1240
1250          LDX #$00       RESET COLUMN COUNT TO 0
1260          LDY #$40       RESET GET TO $4000
1270          STY GET+2
1280          LDY #$00
1290          STY GET+1
1300          JSR GET        GET REPEAT BYTE
1310          STA REP.CHAR
1320
1330
1340
1350
1360 UN.SCRUNCHER
1370          JSR GET        GET A BYTE TO DECODE
1380          CMP REP.CHAR IS IT THE REPEAT CHARACTER
1390          BEQ DO.REPEAT YES SO RUN THRU REPEAT CYCLE
1400          JSR STORE      NOT REPEAT SO JUST SAVE IT
1410          BCC UN.SCRUNCHER ...ALWAYS
1420
1430 DO.REPEAT
1440          JSR GET        GET NUMBER OF TIMES TO REPEAT
1450          STA COUNTER
1460          JSR GET        GET CHARACTER TO REPEAT
1470 .1       JSR STORE      SAVE IT
1480          DEC COUNTER    KEEP TRACK OF COUNTER
1490          BNE .1         DONE?
1500          BEQ UN.SCRUNCHER YES, SO CONTINUE DECODING
1510
1520
1530 *---------------------------------
1540 * THE GET ROUTINE: GETS A BYTE
1550 * AND INCREMENTS POINTER
1560 *
1570 *---------------------------------
1580
1590
1600 GET      LDA $FFFF      GET A BYTE
1610          INC GET+1      INCREMENT LOW BYTE
1620          BNE .1
1630          INC GET+2      INCREMENT HIGH BYTE
1640 .1       RTS            RETURN TO CALLING PROGRAM
1650
1660
```

```
1670 *-------------------------------
1680 * STORE: STORES ACCUM AT HI-RES
1690 * PAGE AND UPDATES VERT/HORZ
1700 * ADDRESS, ROUTINE FORCES AN
1710 * EXIT TO CALLING ROUTINE IF
1720 * YOU ARE DONE
1730 *-------------------------------
1740
1750
1760 STORE
1770          PHA           SAVE BYTE TO PLACE ON SCREEN
1780          TYA           GET VERTICAL LINE NUMBER
1790          PHA           CONVERT TO AN ACTUAL ADDRESS
1800
1810
1820 *-------------------------------
1830 *  USE THE FOLLOWING FORMULA TO
1840 * GET THE ADDRESS OF THE LINE WE
1850 * WISH TO DRAW ON.
1860 *
1870 *-------------------------------
1880
1890          AND #$C0
1900          STA STORE2+1
1910          LSR
1920          LSR
1930          ORA STORE2+1
1940          STA STORE2+1
1950          PLA
1960          STA STORE2+2
1970          ASL
1980          ASL
1990          ASL
2000          ROL STORE2+2
2010          ASL
2020          ROL STORE2+2
2030          ASL
2040          ROR STORE2+1
2050          LDA STORE2+2
2060          AND #$1F
2070          ORA #$20
2080          STA STORE2+2
2090          PLA
2100
2110
2120 *-------------------------------
2130 * STORE THE BYTE ON THE HI-RES
2140 * SCREEN.
2150 *-------------------------------
2160
2170
2180 STORE2   STA $FFFF,X  SAVE THE BYTE
2190          INY          GET NEXT VERT LINE
2200          CPY #192     DONE WITH COLUMN?
2210          BLT .1       NO SO CONTINUE
2220          LDY #$00     RESET VERT LINE TO 0
2230          INX          GET NEXT COLUMN NUMBER
2240          CPX #40      DONE WITH COLUMNS
2250          BGE .2       YES SO RETURN TO CALLER
2260
2270 .1       RTS          REMAIN WITHIN THIS PROGRAM
2280
2290 .2       PLA          PULL OFF LAST CALLER
2300          PLA
2310          RTS          AND RETURN TO CALLER
2320
```

Get.Obj

```
1960 NEXT.LINE STA YP
1970           SEC
1980           LDA XP
1990           SBC #28
2000           STA XP
2010           BNE GET.NEXT.BYTE
2020 *-------------------------------
2030 *
2040 * SET UP FOR SHAPES
2050 *
2060 *-------------------------------
2070
2080 SET.POSN JSR SETHCOL
2090          LDA YP
2100          LDY #00
2110          LDX XP
2120          JMP HPOSN
2130
2140
2150 SAVEALL  STX XSAVE
2160          STY YSAVE
2170          RTS
2180
```

```
2190
2200 RESTORE  LDX XSAVE
2210          LDY YSAVE
2220          RTS
2230
2240
2250
```