HID 'N RAM II
(C) 1984 By Mark Rothstein
And Federal Hill Software
Distributed By Federal Hill Software
825 William St.
Baltimore, Md. 21230

HID 'N RAM II is a programming utility for the 64K Color Computer which enables you, for the first time, to access the entire 64K of RAM from a single BASIC program. Unlike some utilities which "flip" from one page of RAM to another, HID 'N RAM II acually opens a window between the pages, allowing you to store your data in the "hidden" 32K page of RAM and access it directly from a BASIC program. This means you can write a 27K data handling program and still have more than 30K left for the names, addresses or numbers you're crunching. Better yet, HID 'N RAM is the only 64K progamming utility with a built-in machine language sorting routine.

WHY DO YOU NEED HID 'N RAM?

As you probably know, the Microsoft BASIC in the Color Computer was designed to access only 32K of RAM. This is a hangover from the computer's early days, when memory was very expensive. To produce a 32K machiine, Radio Shack originally piggybacked 16K chips. Later it bought "half-good" 64K chips from Motorola and used only one 32K page of RAM. Many of those chips were, in fact, perfectly good, and hackers soon realized that with a few hardware modifications, they could access the entire 64K with a machine language program. Others upgraded from 16K or replaced their half-good 64K chips with good ones.<

By 1982, Radio Shack itself was using full quality chips and in the summer of that year, it began advertising the Color Computer as a "64K" machine. This was something less than full disclosure, however, because the Shack never changed the BASIC interpreter to allow it to access the newly-available memory.

We developed HID 'N RAM because we felt gypped by having a 64K machine that was no better than the old 32K model as far as BASIC programming was concerned. After a lot of experimentation, we came up with a 647-byte machine language driver that we embedded in a BASIC demonstration program. The machine language sits at the top of BASIC. When you edit, CSAVE or SAVE the program, the machine language driver stays with it. When you see how it works, you can delete our BASIC code and write your own, using the techniques you've learned.

The machine language consists of six different subroutines, each defined in a USR call. A USR call is a way of accessing a machine language subroutine from BASIC. It consists of a

definition, which establishes the address of the subroutine, and a statement that actually tells the computer to execute the machine language it finds there. As an example, the statement DEFUSR1=25944 tells BASIC that machine language subroutine No. 1 is found at memory locationn 25944. This type of statement is usually found at the beginning of a program. To exeute it, you use a statement that looks something like C$=USR1(B$). This not only tells the computer to execute the machine language; it also passes a variable from BASIC, in this case B$, for the machine language to work on.

The machine language subroutines in HID 'N RAM work with strings of data that you have defined in your BASIC program. The only limitation is that your data must be in strings of equal length, and the length must be a power of 2. That means they must be exactly 4, 8, 16, 32, 64 or 128 bytes long. We'll show you how to do this.

Subroutine No. 1 stores the data in HID 'N RAM, which we'll call HDR for short. Subroutine No. 2 retrieves the data. Subroutine No. 3 will copy any portion of HDR to any location in normal RAM (be careful with this one!). Subroutine No. 4 sorts the data stored in HDR in ascending order by any substring, while Subroutine No. 5 sorts in descending order. Subroutine No. 6 serves two purposes. It defines the sort substring and he length of the strings you're storing.

The HID 'N RAM PROGRAM

The HID 'N RAM program really consists of two programs. The first is an elementary demonstration of how HDR stores, retrieves and sorts data. The second is a mailing list which will store 450 names in HDR, print one-across labels or a formatted full-page listing, search by name, state or zipcode and sort by name, state or zipcode. The program listing at the back of these instructions is fully commented, and we'll explain it section by section as we go along. When you run HDR, you'll get a menu asking which program you want.

LOADING AND SAVING HID 'N RAM

To load HID 'N RAM from tape, put your cassette in the recorder, press the PLAY button, and ENTER a CLOAD "HDR" command. When the program has loaded and you get the OK prompt, enter a RUN command. There are two copies on your cassette.

To load HDR from disk, just enter a RUN "HDR" command. There is a backup copy on your disk with the filename "HDRBKUP/BAS".

We suggest that you immediately make a backup copy of your HID 'N RAM tape or disk and use it for your day-to-day programming. Put the orriginal away for safekeeping. You can save HID 'N RAM with normal CSAVE and SAVE commands. Be sure you

make a normal binary save and not an ASCII save. Saving the program in ASCII will wipe out the machine language. Likewise, you can't use the MERGE command on disk with an HDR program for the same reason--the machine language will be overwritten. You can, however, save the tape version on disk and he disk version on tape. They are identical.

A Few (Hundred) Words About Strings

If you are familiar with Extended Basic's powerful string handling routines and how they can be used to store both text and numbers, you can skip this section and go on to the program itself. Otherwise, please take the time to read this short course in string programming.

As you probably know, the Color Computer handles two types of variables, numeric and string. In data handling programs, these are generally in subscripted arrays defined in DIM statements at the beginning of a program, such as A(10) or B$(15). A numeric variable can only store numbers, while a string variable can store numbers, text or a combination of the two.

Because BASIC requires a minimum of five bytes just to point to the location of every variable, it is much more efficient to pack data into strings. As an example, suppose a record in your data file consists of five numbers--20, 35, 16, 12 and 14. Defining numeric variables for each of these, even within an array, requires a minimum of 25 bytes. By way of contrast, suppose we pack the numbers into a string that looks like this: 2035161214. We'll call that string A$, and as you can see, it's 10 bytes long. Add another five bytes for the pointers to it, and you have a total of 15 bytes--a 40 percent saving over storing separate numeric variables.

Obviously, we need some way to join the numbers in a string and then separate it into its components. Extended Color BASIC provides us with a number of powerful commands to help out. They are LEFT$, RIGHT$, MID$, STR$, STRING$ and INSTR. The Color Computer is the only low-cost machine with a full implementation of these commands. For a complete discussion of them, consult the appropriate chapters in your Basic and Extended Basic manuals. Three of them are of particular importance in using HID 'N RAM.

PACKING NUMBERS INTO STRINGS

To convert a number into a string, we use the STR$ statement. For example, if we say that A=20, then STR$(A) is a string that looks like this: " 20". You may have noticed the space before the numeric characters. The STR$ command puts this space in the string. To pack your data properly, you have to remove the space. Do it with the MID$ statement. If you say that A$=STR$(A), then to eliminate the space you should then say that

A$=MID$(A$,2). This redefines A$ as the portion of he string which begins with the second character. Now here's a little program that uses this technique to pack five numbers into a string:

```
10 DIM A(5)
20 FOR X=1 TO 5
30 INPUT A(X)
40 A$=STR$(A(X)):A$=MID$(A$,2)
50 B$=B$+A$
60 NEXT X
70 PRINT B$
```

At then end of the program, the computer will print a string called B$ which is composed of the five numbers you entered, in the order you entered them. You could save yourself some trouble by inputting the numbers as strings in the first place, but in many cases you'll want to perform some kind of mathematical calculation on the number before finally storing it in the string. For example, if A(X) is the cost of an item but you want to include a $2 shipping charge in your final data, you might add this line:

```
35 A(X)=A(X)+2
```

Obviously, you must use numeric input here or your mathematical calculation will be meaningless.

All of this is fine if your numbers are of equal length. But what if your five numbers are 20, 6, 110 and 2240? Once they're packed into a string of 2061102240 you have no way of knowing which number starts where. In this case, you'll want to define a fixed length substring for each number. There are a couple of ways of doing this, but this one is the best all around because it will allow most ASCII sorting routines to work properly.

First, figure out how large your largest number will be. For our example, we will assume that we are dealing with whole numbers with a maximum value of 9999 (four digits). So we want each number to occupy four spaces. We'll do it by adding enough zeros to the beginning of each number to pad out the string. The following line added to the program above will suffice:

```
45 IF LEN(A$)<4 THEN A$=STRING$(4-LEN(A$),"0")+A$
```

We can then separate the string into its component numbers using the MID$ statement. The following lines will pick apart the string and print out the numbers it has stored:

```
100 A$(1)=MID$(A$,1,4):A(1)=VAL(A$(1))
110 A$(2)=MID$(A$,5,4):A(2)=VAL(A$(2))
120 A$(3)=MID$(A$,9,4):A(3)=VAL(A$(3))
130 A$(4)=MID$(A$,13,4):A(4)=VAL(A$(4))
```

```
140 A$(5)=MID$(A$(17,4):A(5)=VAL(A$(5))
150 FOR X=1 TO 5
160 PRINT (A(X)
170 NEXT X
```

You can undoubtedly figure out ways to cut down on the size of this code. We do it the long way for the purpose of illustration. In any case, this is one good technique for storing numbers in strings, and it will be invaluable in HID 'N RAM.

PACKING STRING DATA

Text is somewhat easier to work with because it does not always require fixed length strings. However, if you're putting data that requires sorting into your string, you will have to have that part of the data that requires sorting in fixed places within the string.

As an example, suppose you're putting together a mailing list. You want to be able to sort by state and/or zipcode. You know that the abbreviation for a state is always two characters long, while a zip code has five characters. So it will make sorting easier by putting your state and zipcode at the beginning of the string, followed by the name and address. Your string might look like "PA19151NAMEADDRESSCITYSTATEZIP." Your search and sort routines will always look at the first two characters for the state and the third through seventh characters for the ZIP.

But what about the other data, like the Name, address and city? How can we put that data in the string without having to pad it it out into fixed fields that make the string longer than necessary? One way is by using delimiter characters. These are characters that we wouldn't normally find in our string, such as the "$" sign, "@" and "*".

Let's say we want to use these characters to set off the LAST NAME and FIRST NAME in our entry. We know that the first seven characters make up the state and zip code. So the LAST NAME will begin at the eighth character. Between the end of the last name and the beginning of the first name we will insert an asterisk (*). And we'll separate the first name from the street address with a "per" sign (@). So our substring so far, which we'll call A$, will look like this:

```
A$=PA19151JONES*JOHN@825 WILLIAM ST
```

To pick it apart, we will use a little known but very powerful statement known as INSTR. This finds an occurrence of a substring within a larger string and returns the location of the start of that substring. The syntax is A=INSTR(A$,B$) where A$ is the larger string and B$ is the substring you're searching for. Here's how to use it:

```
100 LN=INSTR(A$,"*"):F=INSTR(A$,"@")
110 LASTNAME$=MID$(A$,8,LN-8)
120 FIRSTNAME$=MID$(A$,LN+1, F-(LN+1))
```

This may seem a little complicated, but you'll get the hang of it. And you'll be amazed at how it lets you pack your data into the smallest possible space. We use even more sophisticated techniques in the HDR Mail List. You'll be able to use them in your own programs as well.

Finally, we get down to the question of making all our strings the same length. This is a requirement of HID 'N RAM. For the sake of illustration, we'll say that our string length is 64 characters. But our sample entry is only 53 characters long. We have to figure out some way to pad out the string with 11 spaces. To do this, we use the STRING$ and LEFT$ statements. Here's how it works:

```
A$=LEFT$(A$+STRING$(64,32),64)
```

What this does is add a string of 64 spaces to A$. This makes the new A$ 117 characters long. We then use LEFT$ to separate the first 64 characters of the string. The routine is repeated several times in the demonstration program.

THE BASIC HID 'N RAM PROGRAM

The simple demonstration program begins at LINE 200 and ends at LINE 1030. If you have an earlier version of HDR without the machine language sort, please read this description and look over the listing carefully, as there are a number of critical changes that were necessary to implement the sort routine.

LINES 200-280 are important because they initialize the variables used by the machine language driver. You will need these lines in your own programs. LINE 200 defines the length of the string to be stored, which is 32 bytes long. This is the variable CV and must always be CV in your program. The machine language driver always expects this variable to be set at 32 bytes. If you want to use a different string length you must change CV AND use the DEFINE FIELD routine, which we will discuss later.

IF YOU HAVE RUN THE MAILING LIST, WHICH RESETS THE STRING LENGTH TO 64 CHARACTERS, YOU WILL HAVE TO REDEFINE THE STRING LENGTH THROUGH THE DEFINE FIELD ROUTINE TO GET THIS DEMONSTRATION PROGRAM TO WORK PROPERLY.

LINE 210 identifies the memory location of the start of the machine language driver. Remember that the Color Computer thinks that the machine language is part of BASIC. The end of BASIC is always found by multiplying the value of PEEK(27) by 256 and adding to that the value of PEEK(28). The machine language

starts 647 bytes below that value, which changes with the length of your program. Don't worry, the machine language will always be there.

LINES 220 to 270 define the USR calls that access different machine language subroutines. LINE 280 defines two string variables, UP$ and DOWN$, whichh are used in the sequential access of data you have stored in HDR. It also initializes A and Z, which are used to keep track of how many entries you have stored. Likewise, it initializes two string pointers, AD$ and ZE$. Finally, it sends the program to a string address subroutine at LINE 1030 which will be discussed later. All of these lines must be included in your program for HDR to function properly.

LINES 320 to 350 set up the MAIN MENU for the program. The choices are as follows:

1. STORE DATA IN HDR
2. FETCH DATA FROM HDR
3. COPY FROM HDR TO RAM
4. ASCENDING SORT OF DATA
5. DESCENDING SORT OF DATA
6. DEFINE THE SORT FIELLD
7. RUN THE MAILING LIST

We will discuss the routines in the program in the order in which they occur. Please read the description carefully, as the techniques in these sections are critical.

1. STORE DATA IN HDR

The input and storage of data in HDR appear in LINES 390-490. The routine consists of a FOR-NEXT loop that repeats 100 times. We chose 100 arbitrarily. You could repeat it 1000 times if you wanted and still have some room left over. Remember that we have initialized Z, the current entry number, at zero. This corresponds to the fact that the first string of data stored in HDR will begin at memory location zero. This could be a little confusing on the screen. So LINE 420 identifies the entry as A+1, andd the screen display for the first entry will read, "ENTRY NO. 1." The second entry will be displayed as ENTRY NO. 2 and so on.

Line 410 is crucial to the storage function. It sends you too subroutine 1030, which converts a numeric address (zero to begin with) to a string value. This subroutine must be in all your programs.

Line 420 asks you to input your data and assigns that data to the variable A$. For the purposes of this program, type in and enter anything you want, up to 32 characters. In practice, you may want to use a number of different inputs here and concatenate them into a single string (as we do in the mailing

list). When you are through entering your data (four or five entries will suffice), hit the ENTER key at the entry prompt with no other input.

Line 430 sends you back to the MAIN MENU when you strike the ENTER key with no other input. It also resets the counter variable Z to its correct number. This means that when you return to data input from some other function, you will always be in the right place. Thus, if you quit inputting data after entry No. 4, you will pick up again with entry No. 5.

LINES 440 and 450 actually store the data in HDR. LINE 440 adds the pointer T$ to the string (the first pointer is null, after that the pointer will be "+ ", which tells the machine language to look for the next block of HDR memory. The line also pads out the entry with the appropriate number of spaces, using the technique discussed in the section on string packing. Line 440 shows your padded out string with the pointer attached at the beginning.

LINE 460 is the USR call which stores your string in HDR, while LINE 470 resets the pointer string T$ to "+ " for the next input.

REMEMBER, the critical lines in this function are subroutine 1030, which tells the computer which block of HDR will get this particular string, and LINES 440 to 470, which store the data and set the pointers correctly. The only real programming you have to do in your own program is to define your input properly. The rest can be copied verbatim.

2. FETCH DATA FROM HDR

This routine, LINES 530-630, retrieves data from HDR. Line 530 initializes the initial pointer byte as a null string. Line 540 sets up a FOR/NEXT loop to retrieve the data in sequential blocks. The string address subroutine is called again in LINE 550, while a dummy string is set up in LINE 560. The string consists of 32 spaces preceded by the pointer bytes. That string, called B$, is passed to the machine language via the USR call in line 570. The machine language returns the data from that block of HDR in the form of the string C$. The pointer bytes are stripped away from C$ in LINE 580. This is your data, and in your own program, you will probably want some kind of disassembly routine here to separate it into its components. Line 610 resets the pointer string T$ to "+ ", while LINE 620 completes the FOR/NEXT loop. Once again, you will want to copy this general structure in your own program.

3. COPY DATA FROM HDR TO RAM

This routine, from LINES 680 to 750, will copy any portion of HID 'N RAM to normal RAM. This is for the very adventurous only, as copying data to the wrong spot in normal RAM could

obliterate your program. Remember that you can determine the top of BASIC with this formula: TB=256*PEEK(27) + PEEK(28), where TB is the top of BASIC. You should not copy your HDR data any lower in memory than TB, unless you decide to copy it to the screen area, which consists of memory locations 1024 to 1535. Copying your data to the screen that way will allow you to view it as it is stored in memory and not through the eyes of BASIC's print routine. It will look quite different.

Lines 710 through 740 actually copy the data, using the third USR call. You will be asked for the start and end addresses of the are of HDR that you want to copy. Addresses over 32768 are in HID 'N RAM. You will also be asked for the starting address in normal RAM where you want the data to be copied. Your input should be in decimal form.

4 and 5--SORTING ROUTINES

This routine, from LINES 790 to 870, will sort the strings in HDR. You will be asked for the number of the first and last records you want sorted. If you want all the data sorted, enter zero when asked for the first record. Enter the number of your last record at the second prompt. If you have not defined a sort field and record length, this routine will assume a record length of 32 and a sort field consisting off the first four characters of the string. IF YOU HAVE RUN THE MAILING LIST, YOU MUST REDEFINE YOUR RECORD LENGTH AND SORT STRING OR THIS FUNCTION WILL NOT WORK PROPERLY AND YOU WILL GET AN ERROR MESSAGE.

To make this routine work in your own program, you must define your starting record number as the variable A and your ending record number as B. Then use LINES 830 and 840 exactly as they appear here. Remember that USR4 will sort the data in ascending order, while USR5 sorts in descending order.

The machine language sort routine built into the HDR program is a variant of the bubble sort. It will work faster in your own program if you stop occasionally to sort your data as you go along. Just how long it will take depends on the amount of data you have and how mixed up it is. Our tests indicate that on large files, it is roughly 200 times as fast as a BASIC sort routine. Unfortunately there's no real way to compare it because no other program allows you to store as much data as HDR. We believe the inclusion of this sort makes HID 'N RAM the most sophisticated and useful 64K programming utility on the market.

6. DEFINE THE SORT FIELD

This routine, from lines 920 to 990, is crucial for the proper functioning of HID 'N RAM. It passes to the machine language two important parameters: the length of your record (stored in variable CV) and the location of the substring on

which you want to sort. IF YOU WANT A RECORD LENGTH OF ANYTHING OTHER THAN 32 BYTES, YOU MUST USE THIS ROUTINE AT THE BEGINNING OF YOUR PROGRAM.

LINE 930 prompts you for the he substring you want sorted. You should enter a string of spaces and X's. For example, if you want to sort the first eight characters, enter "XXXXXXXX ". If you want to sort the third through 8th characters, enter " XXXXXX ". You must end the string with a space so that HDR knows where to stop counting. Don't enter the quotation marks from the keyboard, by the way, just the spaces and X's. In your own program, you will probably want to define the sort string, which is called FL$, directly in the code with a statement like FL$=" XXXXXX ".

If you are using the routine to initialize your record length at the beginning of the program, you still must specify a sort field. It doesn't matter what the field looks like at that point--you just need a value to send to the HDR machine language. You can redefine it later when you actually do your sorting. Of course, if you will only be sorting on one field, it's a good opportunity to get the whole process finished at that point.

LINE 940 prompts you for the length of your record. Remember, it must be a power of 2. In your own program, you will probably want to define this directly in the code. We use keyboard input for demonstration purposes only. Once again, both a sort field and a record length must be defined in this routine at the start of your program if the record length is to be anything other than 32 characters.

LINES 950 through 980 take your record length and sort field and pass them to the HDR machine language through the USR call in LINE 980. These should be copied verbatim in your program (leaving out the remarks, if you wish).

This concludes the demonstration HID 'N RAM program. You will probably have to read over it several times before you get the hang of it. Just remember that there are only three crucial routines: storing data in HDR, retrieving it and sorting it. The rest of your program can be carried out with normal BASIC commands, as the mailing list will demonstrate.

THE HID 'N RAM MAILING LIST

The HDR mailing list is designed to store 450 names and addresses in HDR and store them to tape or disk. It will search, sort, display and print out those records by name, state or zip code. There are two different print routines. The first will print one-up mailing labels (we tested ours on the labels Radio Shack sells), while the other will print out a listing with one name on a line across a full page of paper. It will automatically page and put three header line at the top of each

sheet for a pleasing printout of your list.

Each record (that is, a name, street address, city state and zipcode) is in a string of 64 bytes. Theoretically, you can store 32768 bytes of data in HDR, which is the equivalent of 500 such records. But we've found in practice that the last 2000 bytes of HID 'N RAM are somewhat erratic. We don't know why; it's one of the many little mysteries of the Color Computer. So we've played it conservatively and suggest you do the same in your programming. Remember that the number of records you can store is equal to 32768 divided by your string length, minus enough records to stay clear of that last 2000 bytes.

The mail list program begins at LINE 1110. The initialization routine is almost identical to the routine in the demonstration program. The exception is line 1140, which initializes the record length. This is necessary because our records are 64 characters long and not the 32 character default built into the program.

You will see the following choices. Just type the number of the function you want:

1. ENTER DATA FROM KEYBOARD
2. LOAD DATA FROM TAPE OR DISK
3. SAVE FILE TO TAPE OR DISK
4. DISPLAY FILE
5. PRINT FILE
6. SEARCHES
7. SORTS
8. CORRECT AN ERROR
9. BASIC HDR PROGRAM

Look over the listing carefully and you will see that there are a number of clearly defined routines. Most of them are variants of the HDR routines in the demonstration program. Here are the highlights:

1. ENTER DATA FROM THE KEYBOARD

This routine, which begins at LINE 1250, looks very much like the data entry routine in the demonstration program. Instead of entering a single string from the keyboard, however, it calls a data entry subroutine at line 3550. This illustrates the use of BASIC string commands and is worth looking at closely.

The routine at LINE 3550 prompts you for seven entries: last name, first name, address No. 1, address No. 2, city, state and zipcode. We put in two address lines because addresses often include apartment numbers or suites. If there is no second address line, just hit ENTER at the second address prompt prompt. Hitting ENTER with no other input at the LAST NAME prompt will return you to the MAIN MENU just as it did in the demonstration program.

Because we want to sort by state, zipcode and name, we need some way of locating these fields. The mail list program does it by putting the two-letter state abbreviation first in our string, followed by the five-character zip code. This takes care of the first seven bytes of the string.

Alphabetizing the names posed a more difficult problem. We could have put in a fixed field for the last and first names, but that would have wasted space. Instead, we decided to put the last name and first name back to back (last name first) and figure out some way to separate them. Line 3700 does this by finding the length of the last name and converting that number into a single hexadecimal byte. We used hex because it can handle a name of up to 16 characters in a single byte. We store this hex byte right after the zip code. So when we disassemble the string, we take the value of this byte (always the eighth character of the string) and know that the last name starts at the ninth character and extends for the number of characters represented by the hex figure.

The first name is separated from the first street address by the "@" delimiter character in line 3710. Line 3720 checks to see if there is a second address character (A2$). If there is, it is attached to the string and separated from the first address by the delimiter character "%". Line 3730 completes the string by adding the city, set off by the delimiter characters "*" and the up arrow. Thus a typical entry would look like this:

MD21230SJONESJOHN Q@825WILLIAM ST%APT #2*BALTIMORE

This is the string that is padded out to 64 characters and passed on to HID 'N RAM. As you can see, it's a pretty efficient way to pack data.

2 & 3 -- TAPE AND DISK I/O

The Tape/Disk input and output routines begin at lines 1370 and 1600, respectively. Once again, these are variants of the elementary routine that retrieves data from HDR. Only instead of printing the data to the screen, they print the data to tape or disk. Note that in lines 1460 and 1580 we first print to or input from tape or disk the number of records we are dealing wiith. This gives us the upper limit in our FOR/NEXT loop. If you are using a disk drive, you will have the option of getting a directory in these routines. Also, the files created by this

program are stored to disk with an extension of "/HDR" so you can pick them out easily. You can change this in your own I/O routines if you wish.

4. DISPLAY FILE

This routine, beginning at line 1790, offers two types of screen displays. The first will display each record as it would appear on a mailing label. After each entry is displayed, you have four choices -- "C M L P". If you type C you will continue with the next entry. "M" will return you to the MAIN MENU. "L" will print out a mailing label, while "P" will print out a single line listing of that entry.

The other option will display each entry on a single line of the screen. Because of the limited 32-character display of the Color Computer, you're the entry is limited to the name, state and zipcode. But it's a good way to go through your list quickly. After the screen fills up, you have the choice of continuing with the next screenfull of names or returning to the MAIN MENU.

All of he displays and searches make use of the disassembly subroutine at LINE 3800. This takes the string fetched from HDR and separates it into its components. They are given the same variable names they had in the data input subroutine, so that you can follow both the assembly and disassembly of the data string.

The only really tricky thing here is the method used to determine whether there is a second address line. Notice that line 3800 defines the variable A2 as INSTR(C$,"%"). You'll remember that we only use the "%" delimiter if there is a second address string. If there is no "%" in the string, A2 has a value of zero. If there is a second address line, the "%" will show up somewhere in the string and A2 will have a positive value.

5. PRINT THE FILE

The printout routine starts at LINE 2060. It offers two options. One prints out the entire list as labels. The other prints the list, one name and address per line, accross a formattted page. If you choose the full page printout, you will be prompted for three header lines. If you don't want that many lines, just hit the ENTER key with no other input at any or all of the header line prompts. The header will automatically be printed at the top of each page.

Once again, these routines use the normal HDR retrieval technique, followed by the disassembly routine, followed by the appropriate print routine. The mailing label printout subroutine is at line 4010, while the single line printout is at line 4090.

6. SEARCHES

The search routines begin at LINE 2430. You have the option of searching by state, zipcode or name. Each option will ask you for a starting point and ending point for the search. For example, if you want to search for all names from "Carson, John" to "Peters, George", enter Carson and John at the prompts for the last and first names of the beginning of the search and Peters and George for the last and first names of the search limit. If you want to search for just one particular name, enter it when prompted for the start of the search and enter blank lines when prompted for the last name of the search limit.

The same proceedures apply for the searches by state and zipcode. In the state search you must enter the state's two letter abbreviation. In the zipcode search you must enter a five-digit zip code. All three searches give you the choice of a screen display or mailing label printout, so that you can, for example, print out mailing labels for everyone in Pennsylvania, or everyone living in zipcodes 00001 through 30000.

7. SORTS

This routine, beginning at LINE 2980, will sort by state, zipcode or name in ascending or descending order. The sort by state will automatically sort by zipcode within each state. The operative lines here are 3060-3080 which define the sort fields in the variable FL$. This is similar to the sort routine in he demonstration program, except that we define the fields in the code instead of entering them from the keyboard.

Notice that the sort field for state in line 3060 is defined as the first seven characters instead of the first two characters. Because we're doing an ASCII sort, this will automatically cause the zipcode field to be sorted after the state field. If you don't want to sort the zipcodes too, you can redefine FL$ as "XX ". The zipcode sort in line 3070 defines the third through seventh characters, while the alphabetic sort uses the ninth through 26th characters. Remember that the last name starts on the ninth character of the string. We're making the assumption that 18 characters will give us a pretty good alphabetic sort.

The passing of the sort field variables to the HDR machine language takes place in lines 3110 and 3120, while the sorting itself occurs in lines 3150 through 3220.

8. CORRECT AN ERROR

This routine, beginning at line 4180, combines HDR storage and retrieval techniques. You will be asked for the last name and first name of the entry you want to correct. The program searches through HDR until it finds and entry with a name that matches your input. It will display the entire entry and ask whether this is the one you want. If it is, type "Y" and you

will be asked to enter all the information in the entry again. If you type "N" the computer will continue searching for another match. This will allow you to find the proper entry if you should, for example, have more than one John Jones in your file.

Notice that the error correction routine also makes use of the data entry routine at LINE 3550. This saves program space and makes for more efficient code. Also note that error correction makes use of both the HDR retrieval routine (for locating the incorrect entry) and the storage routine for storing the corrected data.

GENERAL NOTES

When you've familiarized yourself with HDR you will undoubtedly want to begin writing your own program. You may want to save portions of our code and incorporate it in your program or delete the entire HDR demonstration and begin from scratch. If you you choose the latter option, DO NOT DELETE THE ENTIRE PROGRAM. LEAVE THE LAST COUPLE OF LINES IN THE PROGRAM INTACT using the command DEL 1-4900. If you delete the entire program, BASIC will have no pointers to the HDR machine language. You can enter BASIC lines with numbers large than ours and then delete our code, but in no case should you use the DEL NNN- command, as this will wipe everything out. Always delete to a specific line number that is lower than the last line number in the program.

You are authorized to make backup copies of this HID 'N RAM program for your own use only. Any other reproduction or resale is prohibited by U.S. and international copyright laws. If you should want to use HDR to create your own commercial programs, you are licensed to do so under the condition that you include this sentence in your program code and documentation: "THIS PROGRAM USES HID 'N RAM, A PROGRAMMING UTILITY AVAILABLE FROM FEEDERAL HILL SOFTWARE, 825 WILLIAM ST., BALTIMORE, MD. 21230. If you should produce a program with commercial potential and don't know how to market it, please contact us as we are always looking for marketable software.

This program is guaranteed to load and run as described in this documentation. No other warranty is expressed or implied and Federal Hill Software bears no responsibility for the consequences of its use. Should your tape or disk fail to load within 60 days of shipping, we will replace it free of charge. After 60 days there is a $5 replacement fee for tape and a $7 fee for disk.

If you're interested an a useful HDR application, Federal Hill Software now offers its popular CoCo-Accountant II program in a 64K HDR version. This home and small business accounting package will handle virtually all your financial record-keeping and make income tax time a breeze. Spend a few minutes each month with your canceled checks, credit card receipts and payroll stubs. Coco Accountant II will list and total expenses

by month, account or payee/income source. It will offset expenses against income for net cash flow statements, sort your entries by date and print out a spreadsheet showing your entire year at a glance.

Special features flag tax deductible expenses and payments subject to state sales tax. Coco Accountant II will even compute the sales tax you paid! The program sorts entries by date and lists most most functions to screen or printer. A separate feature will balance your checkbook and print a monthly reconciliation statement. The program normally sells for $27.95, but HDR owners may purchase it for $21.95 on tape or disk, plus $1.50 for postage and handling.

```
10 CLEAR 2000' HID 'N RAM II
20 '  (C) 1984 BY MARK ROTHESTEIN AND FEDERAL HILL SOFTWARE
30 '
40 'CHOOSE YOUR PROGGRAM
50 '
60 CLS:PRINT"       HID 'N RAM II":PRINT:PRINT"(C) 1984 BY MARK ROTHST
EIN AND":PRINT"       FEDERAL HILL SOFTWARE":PRINT:PRINT"1. RUN BASIC HD
R PROGRAM":PRINT"2. RUN HDR MAILING LIST"
70 HQ$=INKEY$:IF HQ$="" THEN 70
80 IF HQ$="1" THEN 110
90 IF HQ$="2" THEN 120
100 GOTO 70
110 RUN 200
120 RUN 1110
130 '
140 '
150 'START OF BASIC HDR PROGRAM
160 '
170 '
180 ' THIS IS WHERE WE DEFINE OUR INITIAL VARIABLES
190 '
200 CV=32
210 A=256*PEEK(27)+PEEK(28)-651 'START OF HDR MACHINE LANGUAGE
220 DEFUSR1=A 'THIS STORES DATA IN HDR
230 DEFUSR2=A+3 'THIS FETCHES DATA FROM HID 'N RAM
240 DEFUSR3=A+6 'COPIES DATA FROM HDR TO NORMAL RAM
250 DEFUSR4=A+9 'ASCENDING SORT ROUTINE
260 DEFUSR5=A+12 'DESCENDING SORT ROUTINE
270 DEFUSR6=A+15 'DEFINE THE FIELD TO BE SORTED
280 UP$="+ ":DOWN$="- ":A=0:GOSUB 1030:ZE$=AD$:T$=ZE$:Z=0
290 '
300 'MAIN MENU
310 '
320 CLS:PRINT"       HID 'N RAM MENU":PRINT:PRINT"1. STORE DATA IN HDR",
"2. FETCH DATA FROM HDR","3. COPY FROM HDR TO RAM","4. ASCENDING SORT O
F DATA","5. DESCENDING SORT OF DATA","6. DEFINE THE SORT FIELD","7. RUN
 THE MAILING LIST"
330 HW$=INKEY$:IF HW$="" THEN 330
340 ON VAL(HW$) GOTO 390,530,680,790,790,920,120
350 GOTO 330
360 '
370 ' THIS ROUTINE STORES DATA IN HDR
380 '
390 CLS
400 FOR A=Z TO 100
410 GOSUB 1030 'GO TO STRING ADDRESS SUBROUTINE
420 PRINT"ENTRY NO. "A+1:INPUT A$
430 IF A$="" THEN Z=A:GOTO 320
440 B$=T$+LEFT$(A$+STRING$(CV,32),CV) 'ADD THE POINTER STRING AND PAD O
UT WITH SPACES
450 PRINT B$
460 C$=USR1(B$) 'STORE THE STRING IN HDR
```

```
470 T$=UP$:'DEFINE THE POINTER STRING FOR ASCENDING STORAGE
480 NEXT A
490 PRINT "ALL DATA ENTERED":INPUT "HIT enter FOR MENU";PE:GOTO 320
500 '
510 ' RETRIEVE DATA FROM HDR
520 '
530 CLS:T$=ZE$ 'INITIALIZE POINTER STRING T$ AS A NULL STRING
540 FOR A=0 TO Z-1 'SET UP LOOP
550 GOSUB 1030 'STRING ADDRESS SUBROUTINE
560 B$=T$+STRING$(CV,32) 'SET UP A DUMMY STRING
570 C$=USR2(B$) 'RETRIEVE THE DATA FROM HDR
580 C$=MID$(C$,3) 'STRIP AWAY THE POINTER BYTES
590 PRINT "ENTRY NO. "A+1
600 PRINT C$;
610 T$=UP$ 'SET UPWARD POINTER STRING
620 NEXT A
630 INPUT "HIT enter FOR MENU";PE:GOTO 320
640 '
650 'COPY FROM HDR TO NORMAL RAM (BE CAREFUL!)
660 'ADDRESSES OVER 32768 ARE IN HDR
670 '
680 CLS:PRINT"       COPY FROM HDR TO RAM":PRINT:PRINT"ARE YOUR SURE ABO
UT THIS? (Y/N)"
690 HC$=INKEY$:IF HC$=""THEN 690 ELSE IF HC$<>"Y" THEN 320
700 PRINT"ENTER NUMBERS IN DECIMAL":PRINT"NUMBERS ABOVE 32768 ARE IN HD
R":PRINT
710 INPUT "COPY FROM :";A:GOSUB 1030:A$=AD$
720 INPUT "COPY THRU :";A:GOSUB 1030:A$=A$+AD$
730 INPUT "COPY TO   :";A:GOSUB 1030:A$=A$+AD$+STRING$(CV-4,32)
740 A$=USR3(A$) 'COPY THE DATA
750 INPUT "HIT enter TO CONTINUE";PE:GOTO 320
760 '
770 'SORTING ROUTINES
780 '
790 CLS:IF CH$="4" THEN PRINT"      ASCENDING SORT":PRINT
800 IF CH$="5" THEN PRINT  "      DESCENDING SORT":PRINT
810 INPUT "STARTING RECORD NUMBER ";A
820 INPUT "ENDING RECORD NUMBER   ";B
830 GOSUB 1030:A$=AD$
840 A=B:GOSUB 1030:A$=A$+AD$+STRING$(CV-2,0)
850 IF HW$="4" THEN A$=USR4(A$) 'ASCENDING SORT
860 IF HW$="5" THEN A$=USR5(A$) 'DESCENDING SORT
870 PRINT"SORT COMPLETED":INPUT"HIT enter TO CONTINUE";PE:GOTO 320
880 '
890 'DEFINE THE SORT FIELD AND THE STRING LENGTH
900 'YOU MUST USE THIS ONCE YOUHAVE RUN THE MAILING LIST
910 '
920 CLS:PRINT"DEFINE SORT FIELD/STRING LENGTH":PRINT
930 LINE INPUT "SORT STRING: ";FL$
940 INPUT "RECORD LENGTH";CV
950 V=INT(LOG(CV)/LOG(2)+.5):IF V<2 OR V>7 THEN PRINT "STRING LENGTH "C
V" OUT OF RANGE":GOTO 940
960 'THESE TWO LINES DEFINE THE SORT STRING
970 A$=LEFT$(CHR$(0)+CHR$(V)+FL$+STRING$(CV," "),CV+2)
980 A$=USR6(A$):IF ASC(A$)<>0 THEN PRINT "ERROR":PRINTA$
990 PRINT "DEFINITION FINISHED":INPUT "HIT enter FOR MENU";PE:GOTO 320
```

```
1000 '
1010 ' THIS SUBROUTINE CONVERTS A NUMERICAL ADDRESS TO A STRING VALUE
1020 '
1030 AD=INT(A/256):AD$=CHR$(AD)+CHR$(A-256*AD):RETURN
1040 '
1050 '
1060 'BEGIN MAIL LIST PROGRAM
1070 '
1080 '
1090 'DEFINE INITIAL VARIABLES
1100 '
1110 CLEAR 2000:POKE 155,80:CV=64::V=6:A=256*PEEK(27)+PEEK(28)-651:
1120 DEFUSR1=A:DEFUSR2=A+3:DEFUSR3=A+6:DEFUSR4=A+9:DEFUSR5=A+12:DEFUSR6
=A+15
1130 UP$="+ ":DOWN$="- ":A=0:Z=0:GOSUB 3910:ZE$=AD$:T$=ZE$
1140 GF=0:FL$="XX ":GOSUB 3100 ' SET STRING LENGTH TO 64
1150 '
1160 'MAIL LIST MENU
1170 CLS:PRINT"      HID 'N RAM MAIL LIST":PRINT:PRINT"1. ENTER DATA FRO
M KEYBOARD","2. LOAD FILE FROM TAPE OR DISK","3. SAVE FILE TO TAPE OR D
ISK","4. DISPLAY FILE":PRINT"5. PRINT OPTIONS"
1180 PRINT "6. SEARCHES":PRINT"7. SORTS":PRINT"8. CORRECT AN ERROR ","9
. BASIC HDR PROGRAM":IF NF$<>"" THEN PRINT:PRINT"FILE: "+NF$
1190 CH$=INKEY$:IF CH$="" THEN 1190
1200 ON VAL(CH$) GOTO 1250, 1370, 1600, 1790, 2060, 2430, 2980, 4180,10
1210 GOTO 1190
1220 '
1230 'INPUT FROM KEYBOARD
1240 '
1250 FOR A=Z TO 450
1260 GOSUB 3550 'GOTO DATA ENTRY SUBROUTINE
1270 IF LN$="" THEN A=A-1:Z=A+1:GOTO 1170
1280 GOSUB 3910 'GOTO STRING ADDRESS SUBROUTINE
1290 B$=T$+LEFT$(A$+STRING$(CV,32),CV)  'SET UP THE STRING
1300 C$=USR1(B$) 'STORE THE STRING IN HID N RAM
1310 T$=UP$
1320 NEXT A
1330 PRINT"ALL ENTRIES ASSIGNED":INPUT "HIT enter FOR MENU";PE:GOTO 117
0
1340 '
1350 'LOAD DATA FROM TAPE OR DISK
1360 '
1370 CLS:PRINT"LOAD FROM TAPE OR DISK":PRINT
1380 GOSUB 3280
1390 GOSUB 3400
1400 INPUT "FILENAME";NF$:IF LEN(NF$)>8 THEN NF$=LEFT$(NF$,8):PRINT"ABB
R. FILENAME :";NF$
1410 IF MD=1 THEN NF$=NF$+"/HDR"
1420 PRINT"PREPARE "KD$:INPUT "AND HIT enter";PE
1430 T$=ZE$
1440 PRINT "SEARCHING FOR "+NF$
1450 OPEN "I",#MD,NF$
1460 INPUT #MD,Z
1470 FOR A=0 TO Z-1
1480 INPUT #MD, A$
1490 GOSUB 3910
```

```
1500 B$=AD$+LEFT$(A$+STRING$(CV,32),CV)
1510 C$=USR1(B$)
1520 PRINT@480,"ENTRY NO. "A+1;
1530 NEXT A
1540 CLOSE #MD
1550 Z=A
1560 GOTO 1170
1570 '
1580 'SAVE FILE TO TAPE OR DISK
1590 '
1600 CLS:PRINT"SAVE FILE TO TAPE OR DISK":PRINT
1610 GOSUB 3280 ' TAPE OR DISK SUBROUTINE
1620 GOSUB 3400 ' ABORT CHOICE SUBROUTINE
1630 INPUT "FILENAME";NF$:IF LEN(NF$)>8 THEN NF$=LEFT$(NF$,8):PRINT"ABB
R. FILENAME: "+NF$
1640 IF MD=1 THEN NF$=NF$+"/HDR"
1650 PRINT"PREPARE "KD$:INPUT"AND HIT enter";PE
1660 PRINT"SAVING"Z" ENTRIES"
1670 OPEN "O",#MD,NF$
1680 PRINT#MD,Z 'NO. OF ENTRIES IN FILE
1690 T$=ZE$ 'SET POINTER BYTE TO ZERO
1700 FOR A=0 TO Z-1
1710 GOSUB 3910 'GOTO ADDRESS CONVERSION SUBR
1720 B$=T$+STRING$(CV,32) 'CREATE DUMMY STRING
1730 C$=USR2(B$) 'RETRIEVE ENTERY FROM HDR
1740 C$=MID$(C$,3) 'STRIP POINTER FROM STRING
1750 T$=UP$ 'INCREMENT THE POINTER
1760 PRINT#MD, C$ 'PRINT THE ENTRY TO DISK
1770 NEXT A
1780 CLOSE #MD:GOTO 1170
1790 DC=0:CLS:PRINT"        SCREEN OPTIONS":PRINT:PRINT"1. FULL LISTING"
:PRINT"2. SINGLE LINE LISTING","3. MAIN MENU"
1800 DO$=INKEY$:IF DO$="" THEN 1800
1810 IF DO$="3" THEN 1170 ELSE IF DO$="2" THEN 1820 ELSE IF DO$="1" THE
N 1940 ELSE 1800
1820 DC=0:T$=ZE$:CLS
1830 FOR A=0 TO Z-1
1840 IF DC>12 THEN PRINT:GOSUB 3400:DC=0:CLS
1850 GOSUB 3910:GOSUB 3930:GOSUB 3800
1860 NB$=LEFT$(LN$+" "+F$+STRING$(20,32),20)+" "+ST$+" "+ZP$
1870 PRINTNB$
1880 DC=DC+1
1890 NEXT A
1900 GOTO 2020 'THAT'S ALL FOLKS
1910 '
1920 'SCREEN DISPLAY ROUTINE
1930 '
1940 CLS:T$=ZE$
1950 FOR A=0 TO Z-1
1960 GOSUB 3910
1970 GOSUB 3930
1980 GOSUB 3800
1990 GOSUB 3960
2000 GOSUB 4120
2010 NEXT A
2020 PRINT"THAT'S ALL, FOLKS!":INPUT "HIT enter FOR MENU";PE:GOTO 1170
```

```
2030 '
2040 ' PRINT ENTIRE LIST AS MAILING LABELS
2050 '
2060 CLS:PRINT"    PRINTOUT OPTIONS":PRINT:PRINT"1. LABELS":PRINT"2. FU
LL PAGE PRINTOUT","3. MAN MENU"
2070 RP$=INKEY$:IF RP$="" THEN 2070
2080 IF RP$<>"1" AND RP$<>"2" AND RP$<>"3" THEN 2070
2090 IF RP$="3" THEN 1170
2100 IF RP$="2" THEN 2220
2110 PRINT"PREPARE PRINTER":INPUT "AND HIT enter";PE
2120 PRINT"PRINTING LABELS ..."
2130 T$=ZE$
2140 FOR A=0 TO Z-1
2150 GOSUB 3910 'ADDRESS SUBR
2160 GOSUB 3930 'FETCH FROM HDR
2170 GOSUB 3800 'DISASSEMBLE THE STRING
2180 GOSUB 4010 'MAIL LABEL PRINTOUT
2190 PRINT@480, "ENTRY NO. "A+1;
2200 NEXT A
2210 GOTO 1170
2220 'FULL PAGE PRINTOUT
2230 FOR U=1 TO 3:PRINT "HDR. NO."U": ";:LINE INPUT HD$(U):IF HD$(U)=""
 THEN HD$(U)=" "
2240 NEXT U
2250 PRINT"POSITION TOP OF PAPER","AT THE PRINTER HEAD":INPUT "AND HIT
enter";PE
2260 GOSUB 2350 'HEADER ROUTINE
2270 T$=ZE$
2280 FOR A=0 TO Z-1
2290 IF R>50 THEN GOSUB 2350:R=0
2300 GOSUB 3910:GOSUB 3930:GOSUB 3800
2310 GOSUB 4090:R=R+1
2320 NEXT A
2330 GOTO 1170
2340 'PUT HEADER AT TOP OF PAGE
2350 FOR Q=1 TO 10:PRINT#-2:NEXT Q:FOR U=1 TO 3:LH(U)=LEN(HD$(U)):LH(U)
=(80-LH(U)):LH(U)=INT(LH(U)/2 +.5)
2360 PRINT#-2,TAB(LH(U))HD$(U)
2370 NEXT U
2380 FOR Q=1 TO 3:PRINT#-2:NEXT Q
2390 RETURN
2400 '
2410 'SEARCH ROUTINES
2420 '
2430 CLS:PRINT"          SEARCHES":PRINT
2440 PRINT"1. BY NAME":PRINT"2. BY STATE":PRINT"3. BY ZIP":PRINT"4. MAI
N MENU"
2450 SK$=INKEY$:IF SK$="" THEN 2450
2460 IF VAL(SK$)<1 OR VAL(SK$)>4 THEN 2450
2470 IF SK$="4" THEN 1170
2480 CLS:PRINT"              OUTPUT"
2490 PRINT:PRINT "1. SCREEN":PRINT"2. PRINTER"
2500 OP$=INKEY$:IF OP$="" THEN 2500
2510 IF OP$<>"1" AND OP$<>"2" THEN 2500
2520 ON VAL(SK$) GOTO 2540,2720,2850
2530 'SEARCH BY NAME
```

```
2540 CLS:PRINT"SEARCH BY NAME":PRINT
2550 PRINT"START WITH...."
2560 INPUT"LNAME";LN$(1):INPUT "FNAME";LN$(2)
2570 PRINT"END WITH..."
2580 INPUT "LNAME";EN$(1):IF EN$(1)="" THEN 2600
2590 INPUT "FNAME";EN$(2)
2600 TN$(1)=LN$(1)+LN$(2)
2610 IF EN$(1)="" THEN TN$(2)=TN$(1) ELSE TN$(2)=EN$(1)+EN$(2)
2620 L(1)=LEN(TN$(1)):L2=LEN(TN$(2))
2630 T$=ZE$
2640 FOR A=0 TO Z-1
2650 GOSUB 3910 'STRING ADDRESS SUBR
2660 GOSUB 3930 'FETCH FROM HDR
2670 GOSUB 3800 'DISSASEMBLE
2680 IF LEFT$(QN$, LEN(TN$(1))) => TN$(1) AND LEFT$(QN$, LEN(TN$(2))) <
=TN$(2) THEN GOSUB 3480
2690 NEXT A
2700 GOTO 2020 'THAT'S ALL FOLKS
2710 ' SEARCH BY STATE
2720 CLS:PRINT"   SEARCH BY STATE":PRINT"
2730 INPUT "START WITH ";ST$(1):IF LEN(ST$(1))<>2 THEN SOUND 200,1:GOTO
 2730
2740 INPUT "END WITH   ";ST$(2):IF ST$(2)="" THEN ST$(2)=ST$(1):GOTO 27
60
2750 IF LEN(ST$(2))<>2 OR ST$(2)<ST$(1) THEN SOUND 200,1:GOTO 2740
2760 T$=ZE$ 'INITIALIZE POINTER
2770 FOR A=0 TO Z-1
2780 GOSUB 3910
2790 GOSUB 3930
2800 GOSUB 3800
2810 IF LEFT$(C$,2)=>ST$(1) AND LEFT$(C$,2)<=ST$(2) THEN GOSUB 3480
2820 NEXT A
2830 GOTO 2020 'THAT'S ALL FOLK
2840 'SEARCH BY ZIP
2850 CLS:PRINT"     SEARCH BY ZIP":PRINT
2860 INPUT "START WITH ";ZP$(1):IF LEN(ZP$(1))<>5 THEN SOUND 200,1:GOTO
 2860
2870 INPUT "END WITH   ";ZP$(2):IF ZP$(2)="" THEN ZP$(2)=ZP$(1):GOTO 28
90
2880 IF LEN(ZP$(2))<>5 OR VAL(ZP$(2))<VAL(ZP$(1)) THEN SOUND 200,1:GOTO
 2870
2890 T$=ZE$
2900 FOR A=0 TO Z-1
2910 GOSUB 3910:GOSUB3930:GOSUB 3800
2920 IF ZP$=>ZP$(1) AND ZP$ <=ZP$(2) THEN GOSUB 3480
2930 NEXT A
2940 GOTO 2020 'THAT'S ALL FOLKS
2950 '
2960 'SORT ROUTINE
2970 '
2980 CLS:PRINT"   SORT YOUR FILE":PRINT
2990 PRINT"1. BY STATE":PRINT"2. BY ZIP CODE":PRINT"3. BY LAST NAME":PR
INT"4. MAIN MENU"
3000 SO$=INKEY$:IF SO$="" THEN 3000
3010 IF VAL(SO$)<1 OR VAL(SO$)>4 THEN 3000
3020 IF SO$="4" THEN 1170
```

```
3030 CLS:PRINT"     ORDER OF SORT":PRINT:PRINT"1. ASCENDING":PRINT"2. D
ESCENDING"
3040 OT$=INKEY$:IF OT$="" THEN 3040
3050 IF VAL(OT$)<1 OR VAL(OT$)>2 THEN 3040
3060 IF SO$="1" THEN FL$="XXXXXXX " 'FIELD STRING FOR STATE
3070 IF SO$="2" THEN FL$="  XXXXX " 'ZIP FIELD
3080 IF SO$="3" THEN FL$="         XXXXXXXXXXXXXXXXX " 'NAME FIELD
3090 'THIS DEFINES THE FIELD FOR THE SORT ROUTINE
3100 A$=LEFT$(CHR$(0)+CHR$(V)+FL$+STRING$(CV," "),CV+2)
3110 A$=USR6(A$):IF ASC(A$)<>0 THEN PRINT "ERROR"ASC(A$):STOP
3120 IF GF=0 THEN GF=1:RETURN
3130 A=0:B=Z 'DEFINE RECORDS TO BE SORTED
3140 'THIS IS THE ACTUAL SORT
3150 GOSUB 3910 'STRING ADDRESS SUBROUTINE
3160 A$=AD$
3170 A=B
3180 GOSUB 3910
3190 A$=A$+AD$+STRING$(CV-2,0)
3200 '
3210 'LET'S DO THE ACTUAL SORTING
3220 '
3230 IF OT$="1" THEN A$=USR4(A$) ELSE IF OT$="2" THEN A$=USR5(A$)
3240 PRINT "SORT COMPLETED":INPUT "HIT enter FOR MENU";PE:GOTO 1170
3250 '
3260 'TAPE OR DISK SELECTION
3270 '
3280 PRINT "(T)APE OR (D)ISK":PRINT
3290 TD$=INKEY$:IF TD$="" THEN 3290
3300 IF TD$="T" THEN MD=-1 ELSE IF TD$="D" THEN MD=1 ELSE 3290
3310 IF TD$="T" THEN KD$="RECORDER" ELSE IF TD$="D" THEN KD$="DISK DRIV
E":INPUT "DRIVE NO. (0-3):";H:IF H>3 THEN 3310 ELSE DRIVEH
3320 IF TD$="T" THEN RETURN
3330 PRINT"WANT A DIRECTORY? (Y/N)"
3340 YN$=INKEY$:IF YN$="" THEN 3340
3350 IF YN$="Y" THEN DIR:INPUT "HIT enter TO CONTINUE";PE
3360 RETURN
3370 '
3380 '
3390 ' OPTION TO BACK OUT OF I/O
3400 PRINT"(C)ONTINUE (M)ENU"
3410 BK$=INKEY$:IF BK$="" THEN 3410
3420 IF BK$="C" THEN RETURN ELSE 1170
3430 B$=AD$+LEFT$(A$+STRING$(64,32),64) 'SET UP THE STRING
3440 C$=USR1(B$) 'STORE STRING IN HID 'N RAM
3450 '
3460 'SEARCH OPTION TO SCREEN OR PRINTER
3470 '
3480 IF OP$="1" THEN GOSUB 3960:GOSUB 4120:RETURN
3490 IF OP$="2" THEN GOSUB 4010:RETURN
3500 CLS:PRINT"      PRINTOUT OPTIONS":PRINT:PRINT"   (L)ABELS  (M)AIL
LIST"
3510 PP$=INKEY$:IF PP$="" THEN 3510
3520 IF PP$="L" THEN 2060 ELSE IF PP$="M" THEN 2220 ELSE 3510
3530 '
3540 '
3550 CLS
```

```
3560 PRINT "ENTRY NO. "A+1:PRINT
3570 LINE INPUT "LNAME :";LN$
3580 IF LN$="" THEN RETURN
3590 IF LEN(LN$)>15 THEN SOUND 200,1:GOTO 3570
3600 LINE INPUT "FNAME :";F$
3610 LINE INPUT "ADDR1 :";A1$
3620 LINE INPUT "ADDR2 :";A2$
3630 LINE INPUT "CITY  :";CT$
3640 LINE INPUT "STATE :";ST$
3650 IF LEN(ST$)<>2 THEN SOUND 200,10:GOTO 3640
3660 LINE INPUT "ZIP   :";ZP$
3670 IF LEN(ZP$)<>5 OR VAL(ZP$)<1 THEN SOUND 200,10:GOTO 3660
3680 PRINT"IS THIS CORRECT? (Y/N)
3690 CR$=INKEY$:IF CR$="" THEN 3690 ELSE IF CR$<>"Y" THEN 3550
3700 L=LEN(LN$):L$=HEX$(L)
3710 A$=ST$+ZP$+L$+LN$+F$++"@"+A1$
3720 IF A2$<>"" THEN A$=A$+"%"+A2$
3730 A$=A$+"*" +CT$+"^"
3740 IF LEN(A$)>64 THEN SOUND 200,1:PRINT"ENTRY TOO LONG":GOTO 3550
3750 ' PRINT B$
3760 RETURN
3770 '
3780 'DISSASEMBLE THE NAME AND ADDRESS STRING
3790 '
3800 A1=INSTR(C$,"@"):A2=INSTR(C$,"%"):C=INSTR(C$,"*"):L$="&H"+MID$(C$,
8,1):L=VAL(L$):EE=INSTR(C$,"^")
3810 ST$=LEFT$(C$,2):ZP$=MID$(C$,3,5):LN$=MID$(C$,9,L):F$=MID$(C$,9+L,A
1-(9+L))
3820 IF A2>0 THEN A1$=MID$(C$,A1+1,A2-(A1+1)) ELSE A1$=MID$(C$,A1+1,C-(
A1+1))
3830 IF A2>0 THEN A2$=MID$(C$,A2+1,C-(A2+1)) ELSE A2$=""
3840 CT$=MID$(C$,C+1,EE-(C+1))
3850 QN$=LN$+F$:RETURN
3860 PRINT F$+" "+LN$:PRINTA1$:PRINTA2$:PRINTCT$+", "+ST$+" "+ZP$
3870 RETURN
3880 '
3890 '
3900 'CONVERT AN ADDRESS TO A STRING
3910 AD=INT(A/256):AD$=CHR$(AD)+CHR$(A-256*AD):RETURN
3920 'FETCH THE STRING FROM HID N RAM
3930 B$=T$+STRING$(CV,32):C$=USR2(B$):IF ASC(C$)<>0 THEN PRINT "ERROR":
INPUT "HIT enter FOR MENU";PE:GOTO 1170
3940 C$=MID$(C$,3):T$=UP$:RETURN
3950 'SCREEN LISTING ROUTINE
3960 PRINT F$+" "+LN$:PRINTA1$:IF A2$<>"" THEN PRINT A2$
3970 PRINTCT$+", "+ST$+" "+ZP$
3980 PRINT
3990 RETURN
4000 ' MAILING LABEL PRINT ROUTINE
4010 PRINT#-2,TAB(3)F$+" "+LN$
4020 PRINT#-2,TAB(3)A1$
4030 IF A2$<>"" THEN PRINT#-2,TAB(3)A2$
4040 PRINT#-2,TAB(3)CT$+", "+ST$+" "+ZP$
4050 IF A2$="" THEN PRINT#-2
4060 PRINT#-2:PRINT#-2
4070 RETURN
```

```
4080 ' SINGLE LINE PRINTOUT ROUTINE
4090 PRINT#-2,TAB(5)LEFT$(LN$+" "+F$+"  "+A1$+" "+A2$+" "+CT$,65) TAB(7
1)ST$+" "+ZP$
4100 RETURN
4110 'ROUTINE BETWEEN SCREEN LISTINGS
4120 PRINT "YOUR CHOICE: N M L P":PRINT
4130 YC$=INKEY$:IF YC$="" THEN 4130
4140 IF YC$="N" THEN RETURN ELSE IF YC$="M" THEN 1170
4150 IF YC$="L" THEN GOSUB 4010
4160 IF YC$="P" THEN GOSUB 4090
4170 PRINT"DONE":GOTO 4120
4180 'CORRECT AN ERROR HERE
4190 CLS:PRINT"        CORRECT AN ERROR":PRINT:INPUT "LAST NAME";KV$:INP
UT"FIRST NAME";FV$:VT$=KV$+FV$:VT=LEN(VT$)
4200 PRINT"SEARCHING ...":T$=ZE$
4210 FOR A=0 TO Z-1
4220 GOSUB 3910:GOSUB 3930:GOSUB 3800
4230 IF LEFT$(QN$,VT)=VT$ THEN GOSUB 3960 ELSE 4270
4240 PRINT"IS THIS THE ENTRY? (Y/N)"
4260 OU$=INKEY$:IF OU$="" THEN 4260 ELSE IF OU$="Y" THEN 4290
4270 T$=UP$:NEXT A
4280 PRINT"ENTRY NOT FOUND":INPUT "HIT enter FOR MENU";PE:GOTO 1170
4290 PRINT"CORRECTED ENTRY":PRINT
4300 GOSUB 3570
4310 GOSUB 3900
4320 B$=AD$+LEFT$(A$+STRING$(CV,32),CV)
4330 C$=USR1(B$)
4340 PRINT"ENTRY CORRECTED":INPUT "HIT enter FOR MENU";PE:GOTO 1170
10000 SAVE"HDR9":SAVE"HDR9B":SAVE"HDR9/BAS:1":SAVE"HDR9B/BAS:1"
```