Matrix software

315 Marion Ave, Big Rapids, MI 49307 (616) 796-2483 or 796-0381

SORT from MATRIX Software

SORT is a powerful, flexible and easy to use in-memory sorting program written in 6502 machine language. Most sorts are accomplished in less than a second , and very large sorts take only a few seconds. The algorithm is a diminishing increment insertion sort, with optimally chosen increments. This algorithm has the advantage of being significantly faster (but not much longer) than simpler ones, and significantly smaller (but not much slower) than more complicated ones. Moreover, unlike some of the more complicated algorithms, there are no conditions under which the performance of SORT degenerates or fails.

SORT derives as much information as possible about the data it is to sort from BASIC's internal variable descriptors. SORT figures out for itself whether it is sorting integers, floating point numbers or strings, and adjusts itself accordingly. In addition, SORT determines whether the array it is sorting has one dimension or two, and takes into account the size of each dimension. The calling program has only to supply information that SORT can't possibly figure out for itself.

SORT is supplied in eprom. SORT is available on the Commodore PET, APPLE II and ATARI 800. The MATRIX eprom contains versions of SORT for large keyboard PETS only (BASIC 3.0 & BASIC 4.0). In addition, a machine language text screen dump is included in the Eprom as a bonus for users. To call SORT, simply use the SYS command in BASIC. The proper numbers to use with the SYS command are given below.

Before calling SORT, all appropriate parameters must be set using poke commands. The exact locations to POKE along with a description of how to use these parameters follows.

INSTALLING THE EPROM

SORT resides in the enclosed 2716 2K EPROM. This eprom must be properly installed before SORT can be used in conjunction with a BASIC program. Please refer to the enclosed instruction sheet on EPROM INSTALLATION for details. The SORT EPROM on the APPLE II is available via a high quality board designed to work in any slot in the APPLE except slot #0.

SORT SOMETHING!

Altho SORT has been designed with some sophisticated uses in mind, most sorts are very easy to perform and require almost no set-up. Here is an example where we are going to sort an array (´A´) containing six floating point numbers. You may want to key in this short program to gain confidence in the use of SORT. You will note SORT makes use of the Zero-eth subscript ´A(0)´ as the first element in the array. In this sort, all that is required in the way of set-up is to tell sort WHICH array we want sorted and to issue a call to SORT. The array is sorted and available to us for use as part of our BASIC program.

```
10 DIM A(5),B(100):REM DIMENSION ARRAYS
15 REM
20 A(0)=257.50:A(1)=125.30:A(2)=90.24:REM ASSIGN VALUES
25 A(3)=356.48:A(4)=21.24:A(5)=524.17:REM TO ARRAYS
30 REM
50 POKE905,1 :REM TELL SORT YOU WANT FIRST ARRAY DIMENSIONED
55 POKE907,0 :REM TELL SORT YOU WANT ZERO DIMENSION
60 SYS 36864 :REM SORT ARRAY (3.0 BASIC AT $9000 SOCKET)
70 REM
80 FORI=0TO5  :REM PRINT OUT
90 PRINTA(I)  :REM SORTED
95 NEXTI      :REM ARRAY
97 END
```

The ´POKE905,1´ is the only set-up required to achieve this sort. It tells SORT that we want to sort the first array dimensioned in the program which was ´A´. If we had wanted to sort the ´B(100)´ array, we would have done a ´POKE905,2´ and so forth. The ´SYS´ command is a call to the sorting algorithm itself. The number following the ´SYS´ command (SYS 36864) will vary depending upon WHICH BASIC you have (3.0 or 4.0) and on which of the three empty ROM sockets in the PET you have installed your SORT EPROM. The above example assumes that the MATRIX EPROM is installed in the $9000 socket of a PET with 3.0 BASIC "### COMMODORE BASIC ###". You will find a TABLE of these ´SYS´ commands at the end of these instructions for BASIC

versions 3.0 and 4.0 (and for the three possible sockets). This 'SYS' command is the only command in SORT that differs between sockets/BASICS. All of the set-up commands (i.e. POKE905,5 etc.) are identical for both BASICS, and all sockets.

You just performed your first sort. You could have as easily sorted an integer array or any number of other floating point arrays by POKEing their position in the DIMension statement and call the SORT via the 'SYS' command. String arrays are just as easy and offer many additional opportunities in the way of multi-key sorts. Using the advanced multi-key string capabilities of SORT, it is possible to extract very specific information from a string filled with data. The set-up procedure is detailed below. Be sure to read over this material with care.

SORT SET-UP Parameters

There are three parameters that must be set before SORT is called. These parameters are stored in the second cassette buffer of the PET using POKE commands. These locations have been selected so as not to interfere with use of the second cassette buffer by the new BASIC 4.0.

(1) WHICH ARRAY TO SORT? (POKE 905,X) This array identifier tells SORT which array you wish to sort. As a program is executed, arrays are declared explicitly by either appearing in DIM statements, or implicitly by appearing in experessions (BASIC statements). As each array is declared in this way, each may be counted in sequence, starting with 1 (the first). The number POKED into location 905 is simply the counting sequence number of the array you wish to sort. Once this value has been set, it stays set, and need not be POKED again until you desire to sort a different array. In other words, you tell SORT which array to sort by indicating the order (1,2,3, etc.) in which it was dimensioned. Take for example the following line of BASIC code:

```
10 DIM A$(25),B$(100),CA$(250)
```

If I wish to have the 'B$(100)' sorted, I would indicate this to SORT by a POKE 905,2 statement since the 'B$' array is the second array dimensioned. A POKE 905,1 would select the 'A$(25)' or first array and so forth.

The only exception to this straight-forward proceedure is if the programmer fails to dimension arrays via the DIM statement and, instead, just uses them in the program code. This can only happen if the array has eleven or less elements.

In this case, the programmer will have to keep in mind that SORT counts its arrays in the order in which they are dimensioned via a DIM statement and/or in the order in which they are actually used in a program.

(2) WHICH DIMENSION? (POKE 907,X) This parameter is asking you which dimension or index of the array you are sorting on. This parameter only pertains to multiple dimensioned arrays- arrays that have more than one dimension [i.e. A(1,5), B$(2,20), I%(2,1000)]. Otherwise, always set this parameter with a POKE 907,0 which indicates that you are working on the first dimension (zero) of the selected array. A example in BASIC:

```
10 DIM A$(2,1000), B$(1,20)
```

There are three dimensions to the ´A$´ array. They are the ´0th´, the ´1st´ and the ´2nd´ dimensions. Each of these dimensions consists of 1000 elements. Dont forget the zero-th element. If I wanted to sort on the ´3rd´ dimension of the A$ array, I would perform a POKE 907,2 and so forth. Since most sorting is done with arrays of but a single dimension, a POKE 907,0 will be what is usually needed. NOTE: When sorting multi-dimensional string arrays with more than one key you will have to poke the dimension to be sorted for each key. More will be detailed on this below on the section on MULTI-KEY sorts.

(3) HOW MANY KEYS? (POKE 906,X) This parameter is used when sorting STRING arrays to determine how many KEYS in a particular string are to be considered by SORT. This parameter is to be set to ´1´ for INTEGER, FLOATING-POINT and single key STRING arrays (POKE 906,1). SORT is capable of handling up to 20-KEYs in a sort at the same time. Here is an example of how a three KEY sort might be used.

```
A$(1)="JOHN      MICHAEL     ERLEWINE    "
       01234567890123456789212345678 93 = position in string
```

Imagine an array (A$) containing the first, middle and last names of a group of clients. We want to sort these names by (1) LAST name, (2) FIRST name and (3) MIDDLE name. This will require using THREE keys. We would communicate this to SORT by a POKE 906,3. We also need to tell SORT where in the string to begin sorting (on each key) and also how many characters to evaluate (for each key). We do this thru using two more parameters. These extra parameters need only to be set when sorting STRING arrays and can be ignored when sorting REAL (Floating Point) or INTEGER arrays.

MULTI-KEY SORTS

Keep in mind that for each KEY we use in a string array sort, we must specify what character position in the string to begin sorting on and how many characters are to be evaluated and even what dimension of an array is to be sorted if we have a multiple dimensioned array. SORT does not care whether the characters to be sorted in a string are letters of the alphabet or numbers. There are twenty POKE locations beginning at POKE 947,X to indicate where in a string each key is to begin and twenty POKE locations beginning at POKE 927,X to indicate how many characters are to be evaluated for each key and twenty POKE locations beginning at 907,X to indicate (for each key) which dimension of the array is to be sorted (this only necessary for multiple dimensioned arrays.

Evaluate#		Start At	Dimension#
POKE 927,X	KEY-1	POKE 947,X	POKE 907,0
POKE 928,X	KEY-2	POKE 948,X	POKE 908,0
POKE 929,X	KEY-3	POKE 949,X	POKE 909,0
....	KEY-4	....	....
etc. thru			
POKE 946,X	KEY-20	POKE 966,X	POKE 926,X

Note that the number of characters to evaluate for the first KEY (POKE 927,X) matches the character position to begin sorting for the first key (POKE 947,X) matches the dimension to sort on (POKE 907,X) and so forth. Let us use the example given above to illustrate.

```
        First     Middle      Last

A$(1)="JOHN       MICHAEL     ERLEWINE    "
A$(2)="JOHN       MICHELL     ERLEWINE    "
       01234567890123456789212345678 93=  string position
```

We have indicated that we want to use a three KEY sort via the POKE 906,3 parameter mentioned above. Now we want to define the three keys so that our list of names is sorted so that the Last name is sorted, then the first and lastly the middle names. We will place the starting character position of our first key in location POKE 947,X. Note that character positions within the string are counted starting with the numeral '0' for the first character, '1' for the second and so on. Therefore, the first character position for sorting the Last names starts at the 20th position in the string and continues for 11 places. Our first key, therefore will be set as follows:

POKE 947,20 = First KEY Starting
POKE 927,11 = First KEY #Evaluate
POKE 907, 0 = First KEY Dimension

The field for our second key will be the First names and we will start sorting at the first character of the string which is position zero and evaluate 9 characters. Our third and final KEY will be the middle names starting with the 9th position and evaluating 11 characters. The three key set-up will look like this.

Position	Evaluate	Dimension	
POKE 947,20	POKE 927,11	POKE 907,0	FIRST KEY
POKE 948,0	POKE 928,9	POKE 908,0	SECOND KEY
POKE 949,9	POKE 929,11	POKE 909,0	THIRD KEY

SUMMARY

SORT requires several parameters to be defined by the user. These are as follows.

POKE 905,X ARRAY to be sorted in the order DIMensioned by BASIC
POKE 906,X HOW MANY KEYS in the sort

In addition, STRING arrays require three additional parameters to be set.

POKE 907,X DIMENSION TO SORT ON FOR CURRENT KEY
POKE 927,X NUMBER of CHARACTERS to be EVALUATED in string
POKE 947,X WHICH CHARACTER POSITION to BEGIN EVALUATION

We have shown how each of these last three parameters may be further defined up to a total of 20 KEYS.

These parameters may be POKED into memory either in immeditate mode or as part of a BASIC program. There are no default values for the SORT parameters so they must be set. Once these have been set, SORT may be run as many times as needed. Two different sorts may share some of the same parameters and only those that differ need to be changed. That is all there is to the set-up proceedure. All that remains is to initialize SORT.

SORT= SYS ????? [* see TABLE of addresses]

Once the SORT parameters have been set, SORT is

accomplished via a call to SYS 36864 (this number depends upon whether you have BASIC 3.0 or 4.0 and which ROM socket you have installed the SORT EPROM- see TABLE at end of instruc tions). The SORT will be accomplished and the sorted array is then available to the user. I am sure that the user will quickly find many uses for this multi-key hi-speed sort. In particular, STRING arrays can be set up and manipulated via the sort in very many useful ways. Here is an example to give you some ideas.

```
10 DIM A$(200)

      Birth Date*Name                    *Size Pet*Last Sale *

A$(1)="1941 07 18*ERLEWINE    MICHAEL *32K  PET*1980 12 12*
A$(2)="1911 12 24*JONES       BOB     * 8K  PET*1979 01 15*
A$(3)="1951 08 14*ERLEWINE    ANNE    *16K  PET*1980 02 25*
```

The above STRING array contains information stored so that it can be both sorted and printed as it stands- without re-formatting. The set-up for this array would be as follows.

```
POKE 905,1    First Array Dimensioned
POKE 906,1    One Key

POKE 907,0    Zero-th Dimension
POKE 927,X    total # characters to evaluate (see below)
POKE 947,X    position of character to begin SORT (see below)
```

I May sort on Last name by POKE 927,11 & POKE 947,11 & POKE 908,0. I may sort on Birthdate by POKE 927,10 & POKE 947,0 & POKE 908,0. I may sort on the most recent purchase with a POKE 927,10 & POKE 947,40 & POKE 908,0. I can even sort on the size PET owned with a POKE 927,2 & POKE 947,31 & POKE 908,0.

After the SYS 36864 sorts each of the above, I can simply print out the array and obtain a meaningful sorted list that highlights the key I have selected. This 'pictured' storage of information can be a very powerful tool. Of course, I could have used a multi-key sort (for example) all those born in a given year and the size PET owned and the most recent purchase made.

COMMODORE PET Limitations using SORT

Given the current buffer size of the PET, the maximum number of dimensions for arrays is as follows: 7 dimensions for integer arrays, 3 dimensions for floating point and 5 dimensions for string arrays. Thus in the floating point array

A(2,999), there are three dimensions (0,1 and 2) each containing 999 records. Most sorting tasks of several hundred members are instantaneous. Even large sorts take only a few seconds. Companies interested in including SORT in their own software are invited to call us for license details.

MATRIX TEXT SCREEN DUMP

Also included in the MATRIX eprom is a text screen dump for both the 40-column and 80-column CBM computers. This routine will dump all ASCII characters from the screen to a printer. The dump will ignore the Commodore special graphic symbols and replace them with the space character ´ ´. This dump should work with any ASCII printer. Here is the set up.

```
100 GETG$:IFG$=""THEN100
110 IFG$="?"THEN OPEN4,4:CMD4:SYS37681:PRINT#4:CLOSE4:GOTO100
120 RETURN
```

The above BASIC subroutine can be inserted in your programs for use with our screen dump. When the program you have running finishes formatting a screenfull of DATA you can do a GOSUB100 for the screen dump.

The GET command in line 100 will hold the screen steady. Pressing the ´?´ KEY will execute the screen dump after which program control is directed to the GET command. Press any KEY but ´?´ and you return from the subroutine and continue normal program flow.

In line 110, after the ´?´ KEY has been pressed, the following occurs. (1) FILE #4 is opened to Device# 4 (Commodore Printer). A CMD4 is issued directing the computers attention to device #4. A SYS 37681 call is made to the machine language screen dump after which a PRINT#4 returns program control to BASIC. At that point, FILE 4 is closed and the program is directed to the GET routine in line 100. The user may substitute different file numbers and/or device numbers. One Note: The REVERSE FIELD space will print out as an asterisk. This allows the use of the space, when in reverse field, for borders, etc. on the screen that will also reproduce to the printer.

Having a screen dump handy provides instant access to hard copy without re-formatting the copy for the printer. We hope you find this and SORT useful. Call us if you have any questions.

SYSTEM COMMANDS for ROM Sockets $9000, $A000 & $B000

	ROM $9000	ROM $A000	ROM $B000
SORT 3.0	SYS 36864	SYS 40960	SYS 45056
SCREEN DUMP 40c	SYS 37681	SYS 41777	SYS 45873
SCREEN DUMP 80c	SYS 37765	SYS 41861	SYS 45957
SORT 4.0	SYS 37849	SYS 41945	SYS 46041

The MATRIX SORT comes in eprom prepared for ROM location $9000. However, it can also be ordered for the $A000 or the $B000 ROM sockets. Please refer to the correct system addresses for the eprom you have ordered. These instructions assume the $9000 socket is being used. See above table for other location SYS addresses.

REFERENCE TABLE of SORT PARAMETERS

PARAMETER	FUNCTION	THIS EXAMPLE
POKE 905,1	ARRAY DIMENSIONED	First ARRAY
POKE 906,1	Number of KEYS	ONE KEY
POKE 907,0	DIMENSION OF ARRAY	Zero-th Dimension
POKE 927,5	Number CHARACTERS in STRING for EVALUATION	SORT on five CHARACTERS
POKE 947,0	WHAT CHARACTER to BEGIN EVALUATION	START with FIRST Character
SYS 36864	EXECUTE SORT	$9000 SOCKET for SORT EPROM