MICROPOLIS USERS GROUP

MUG Newsletter # 4 - November 1980

Published monthly by the MUG, Buzz Rudow - Editor
604 Springwood Cir., Huntsville AL (205) 883-2621

Subscription (August through July) rates:
U.S., Canada, Mexico; $12/year: Other, $18/year
Mid-year subscribers receive current year's back issues.

POSTCARD FOR COMMENTS

Feedback is an indispensable element of any function meant to provide a service - such as the MUG. I've had the pleasure of speaking with, or receiving notes from, several of the members. For the majority of you, however, I have no idea whether you consider the newsletter good, bad, or ho-hum. That's why you'll find the preaddressed postcard in this mailing. Please take the time to scribble a few succinct comments related to the sum of the first four issues. Just stuff like "more A", "less B", "too technical", "more detail", or "how about an article on (topic)". I'd appreciate you signing it but I don't need any addresses - I have that information.

.....

NEVER ENOUGH TIME

I haven't, as usual, completed the investigation needed to write about all the topics I proposed for this issue. I tend to get caught up in the first thing I tackle, in this case the "TX" program which follows. The "curse" of all programmers, according to project managers who are looking for an on-time delivery, is a tendency for eternal improvements. There's always something one can do to make it shorter, make it faster, make it more universal, make it more idiot-proof. So I get to playing and then, darn - it's the end of the month already.

.....

BASIC REPLACEMENT FOR FILECOPY

Each of us, at some time, needs to rearrange the structure of a file. The FILECOPY utility doesn't allow moving a selected portion of a file so I implemented a BASIC routine. In it's most elemental form, the routine looks like this:

```
05 DIM W$(250)
10 STRING "^"
15 OPEN 1 "OLD"
20 OPEN 2 "N:NEW"
25 FOR I=1 TO RECPUT(1)-1
30 GET 1 RECORD I W$
35 PUT 2 W$
40 NEXT I
45 STRING ","
50 CLOSE 1
55 CLOSE 2
60 END
```

This code does an equivalent task to FILECOPY - will reproduce a file. Each time you run it you have to edit lines 15 and 20 so that "OLD" and "N:NEW" have the correct names and correct drive-reference.

Use line 10 to change the string delimiter to any character not found in your data. Reading a record into a text string with the string delimiter designated as some character not contained in the record, causes all of the record to be placed in your designated variable - in this case W$. W$ is dimensioned to 250 so that a total record can fit. It doesn't matter if your data is small strings separated by commas (or whatever), or numbers separated by spaces, or a mix of the two. It will all go into W$ without changing a bit (pun) of the data. When written back out to a disk, the data is still an exact reproduction of the original.

You don't have to always write to a new file. Line 20 can say OPEN 2 "NEW". With that change, the program would read for OLD on disk-0 and write it on the end (append) of NEW on disk-0. You can write disk-to-disk by prefixing "0:" or "1:" to the file name.

You don't have to move the entire OLD file. If you only want to move records 20 through 25, edit line 25 to read FOR I=20 TO 25.

As an example, suppose you want to rearrange a 1000-record file and put the middle 500 on the front, followed by the first 250, then the last 250. Run the program three times using successive changes of lines 15, 20, and 25 as follows.

```
15 OPEN 1 "OLD"          15 OPEN 1 "OLD"          15 OPEN 1 "OLD"
20 OPEN 2 "N:NEW"        20 OPEN 2 "NEW"          20 OPEN 2 "NEW"
25 FOR I=251 TO 750      25 FOR I=1 TO 250        25 FOR I=751 TO 1000
```

REWRITE FOR "IDIOTS"

The above program is fine for the programmer/operator who wrote the program. But get someone else on the keyboard and the routine is suddenly useless. Only the author knows what it is meant to do, and what to change to make it do various things. So I rewrote the program for "publication", named it "TX", and the listing is on pages 3 and 4.

I tried to make it as "idiot-proof" as possible, with the exception of answering file-name prompts. It is still possible to blow the program by responding with an existing file-name for the name of a proposed new file, and by responding with a non-existent file name for the name of an existing file.

I don't normally write code this well idiot-proofed, but it is the preferred method if either untrained operators are running it, or if a crash will cause a sizable loss of time or data.

TEXT STRINGS, NOT NUMBERS

All responses to prompts are input as text strings. In this way, the response can be checked for numeric value before it is converted to a number. A non-numeric response to a numeric input request causes an error, as you know. Numeric responses are then validated against the file size. The first file transferred must be greater then 1 and less than or equal to the file size. The last file

```
 10      !"TX" - Transfers records, file-to-file
 20      ! Set clear-screen char
 30      !
 40      S$=CHAR$(11):IF PEEK(16R500)=79 S$=CHAR$(4)
 50      DIM W$(250),V$(49,250)
 60      STRING "^"
 70      !
 80      ! Select input file
 90      !
100      B$=" INPUT "
110      GOSUB 860
120      GOSUB 700
130      OPEN 1 F$
140      !
150      ! Select output file
160      !
170      B$=" OUTPUT "
180      GOSUB 860
190    > INPUT "Does the output file presently exist? (Y or N):";R$
200      IF R$<>"Y" AND R$<>"N" PRINT:PRINT "ANSWER WITH A 'Y' or 'N'!"
         :PRINT:GOTO 190
210      IF R$="N" GOTO 300
220      !
230      ! Open output file if it's old
240      !
250      GOSUB 700
260      GOTO 350
270      !
280      ! Open output file if it's new
290      !
300    > INPUT "Enter Name of OUTPUT File:";F$
310      F$="N:"+A$+":"+F$
320      !
330      ! Specify records for transfer
340      !
350    > OPEN 2 F$
360    > INPUT "Enter # of 1st record to be transferred:";R$
370      GOSUB 1010
380      IF A>0 PRINT:PRINT "INPUT MUST BE ALL NUMERIC!":PRINT:GOTO 360
390      Y=VAL(R$)
400      IF Y<1 OR Y> RECPUT(1)-1 PRINT:PRINT "1ST RECORD MUST BE >0 an
         d <";RECPUT(1):PRINT:GOTO 360
410    > PRINT "Enter # of last record to be transferred,"
420      INPUT "or '9999' if remainder of file:";R$
430      GOSUB 1010
440      IF A>0 PRINT:PRINT "INPUT MUST BE ALL NUMERIC:":PRINT:GOTO 410
450      Z=VAL(R$)
460      IF Z>RECPUT(1)-1 Z=RECPUT(1)-1
470      IF Z<Y PRINT:PRINT "LAST RECORD MUST BE >";Y-1:PRINT:GOTO 410
480      !
490      ! Perform the transfer
500      !
510      Q=0
520      FOR I=Y TO Z
530          GET 1 RECORD I W$
540          V$(Q)=W$
550          Q=Q+1
560          IF Q=50 GOSUB 930
```

```
 570       NEXT I
 580       IF Q=0 GOTO 600
 590       GOSUB 930
 600     > CLOSE 1
 610       CLOSE 2
 620       PRINT S$
 630       PRINT "Transferred record";Y;"thru";Z;"for a total of";Z-Y+1;"
           records."
 640       STRING ","
 650       END
 660       !
 670       ! Subroutine for displaying contents of disk;
 680       !                    selecting file for operation
 690       !
 700 >*>   PRINT S$
 710  *    PRINT TAB(10);"THE FOLLOWING FILES ARE AVAILABLE:"
 720  *    PRINT
 730  *    F$=A$+":DIR"
 740  *    DISPLAY F$
 750  *    PRINT
 760  *    PRINT "If desired file is not listed, insert"
 770  *    PRINT "another disk, type 'X', press RETURN."
 780  *    PRINT
 790  *    PRINT "Enter Name of";B$;:INPUT "File:";F$
 800  *    IF F$="X" GOTO 700
 810  *    F$=A$+":"+F$
 820 <*    RETURN
 830       !
 840       ! Subroutine for selecting drive
 850       !
 860 >*    PRINT S$
 870  *>   PRINT "Enter Drive # for";B$;:INPUT "File; (0-1):";A$
 880  *    IF A$<"0" OR A$>"1" PRINT:PRINT "DRIVE SELECTION MUST BE '0' o
           r '1'!":PRINT:GOTO 870
 890 <*    RETURN
 900       !
 910       ! Subroutine for writing V$-array
 920       !
 930 >*    FOR R=0 TO Q-1
 940  *        PUT 2 V$(R)
 950  *    NEXT R
 960  *    Q=0
 970 <*    RETURN
 980       !
 990       ! Subroutine to verify all numeric input
1000       !
1010 >*    A=0
1020  *    FOR I=1 TO LEN(R$)
1030  *        IF MID$(R$,I,1)<"0" OR MID$(R$,I,1)>"9" A=A+1
1040  *    NEXT I
1050 <*    RETURN
1060       !
1070       ! VARIABLES
1080       !
1090       ! A : FLAG FOR NON-NUMERIC
1100       ! I : LOOP VARIABLE
1110       ! Q : INDEX FOR V$-ARRAY
1120       ! R : LOOP VARIABLE
```

```
 1130      ! Y : FIRST RECORD
 1140      ! Z : LAST RECORD
 1150      ! A$: DRIVE #
 1160      ! B$: 'INPUT' OR 'OUTPUT' PROMPT
 1170      ! F$: DRIVE & FILE NAME
 1180      ! R$: GENERAL TEXT INPUT BUFFER
 1190      ! S$: CLEAR-SCREEN CHAR
 1200      ! V$: SET OF 50 OUTGOING RECORDS (FILE 2)
 1210      ! W$: INCOMING RECORD (FILE 1)
```

```
Title:  TX


Variable sizes:    Real - 5     Integer - 3    String - 40

Logical memory end:  BFFF H     0000 H bytes are reserved in high memory

FA*

FN*

A  I  Q  R  Y  Z
A$(40) B$(40) F$(40) R$(40) S$(40) W$(250)


V$(49,250)


Memory Allocation
====== ==========

Interpreter:     22272  =  5700 H
Program:          2514  =  09D2 H

Real Var:          130  =  0082 H
Integers:            0  =  0000 H
Strings:           462  =  01CE H
Arrays:          12604  =  313C H
Total Var:       13196  =  338C H

Total Alloc:     37982  =  945E H
Available:       49152  =  C000 H

Dynamic allocation & buffers not included
```

---

transferred must be equal to or greater than the first file transferred, and less than or equal to the file size.

The concept of storing the input in an array and then writing when full (line 540 and the subroutine at lines 900-970) is used in lieu of a one-for-one GET 1/PUT 2 to speed things up and to reduce head actuation cycles if you're going disk to disk.

The dimensioning of the V$-array is arbitrary - you can use 10, or 100. The impact is memory used to store the array, 250 bytes per dimension increment. If used as a stand-alone program, you can go to about 90 in a 48K system.

UNDERSTAND?

I do not know whether each of you considers this sort of program trivial or beyond comprehension. It is my intention to build, as the months go on, a set of these programs, which can then be integrated into a system. But unless you can understand what's being done, and why, you won't be able to modify them for your particular file attributes or system needs. So I am quite willing to discuss the routines I write, including modifications for your needs, by letter or phone. If calling, I am normally home at 7PM (Central Time) weekdays, or any time weekends.

.....

PROGRAMMING AIDS

The initial group of MUG members were reached because Systemation included a flyer in their promotional mailing. I assumed most members were aware of Systemation's set of programming aid products. Now we're expanding and that's not the fact anymore. So, the listing of the program was done by way of BEM, the following compacted version of the TX program was done by CRUNCH, and the cross-reference listing of TX was produced by XREF. BEM, CRUNCH and XREF are all products of Systemation. A considerable amount can be said about the options and uses of BEM and XREF. It will have to wait for a subsequent issue, however.

.....

CP/M FOR MICROPOLIS

One of the things I didn't get around to tackling this month was a CP/M article. I haven't changed my opinion of wanting to stay with Micropolis exclusively. There are instances in my business where an understanding of CP/M would be of benefit, however. I, as well as some of the members, would like to be able to convert programs back and forth between MDOS/BASIC and CP/M. One of these months I'll accomplish some work in this area.

For now, all I can do is steer you to some literature I've glanced at and found worthy of purchase - though I haven't studied it well enough to put it to use. The "CP/M Handbook with MP/M" by Rodnay Zaks, published by Sybex ($13.95, 2344 Sixth Street, Berkley CA 94710) is easy to read and seemingly very complete. I finally discovered that control-P enabled me to list the disk directory to the printer - something I'd half-heartedly been trying to do for a year and a half. "S-100 Microsystems" magazine ($9.50/yr for six issues, PO Box 789-M, Morrestown NJ 07960) has run a good series of articles on 'An Introduction to CP/M'. You'll have to go back to issue one, Jan/Feb 1980. It's a good magazine in general for those of us who don't own TRS, Apple, Pet, etc - which is all one gets in most computer magazines these days. Finally, there's "Lifelines", a monthly magazine published by Lifeboat ($18/year, 1651 Third Ave, New York NY 10028). Lifeboat is the source of CP/M for Micropolis. Lifelines has a columnist, going by the name of Zoso, who's almost worth the price of the subscription himself. Typical of his offerings is the following - to the tune of Yankee Doodle Dandy.

I'm a Dinky Little Tandy,
Assembled largely in Hong Kong;
I'll process data near the speed of light,
And only rarely do it wrong.

I'm likely to expire when you need me most,
When some Haitian-made component fails;
My keyboard bounces like a tennis ball,
But that's what owning me entails.

I can be quite temperamental,
If the climate's hot, cold, wet or dry,
Or if there is the smallest variance,
In your hundred-ten volt supply.

I'm the new star of those magazines,
Which long ago were somewhat good;
Advertise, convince poor fools to buy,
Things their own designers never would.

I'm the Dinky Little Tandy,
The wave of the future, wait and see;
Make no improvements to my circuit boards,
Or you'll void my curious warranty.

Yes, I'm the Dinky Little Tandy,
Pride of so many thousand eyes;
Trust me to supervise your business,
And I'll warranty the odd surprise.

CP/M VERSION 2.2

I spoke to Jerry Sawyer of Lifeboat Associates and was informed that Version 2.2 of the Micropolis Mod II CP/M system is released. I won't go into the differences between Versions 1.X and 2.2 but the 2.2 version, selling for $200, is worth the extra $55. One major change is that file size can now be up to 512 K-bytes instead of the 30K or so (I'm not sure of the number) allowed by Version 1.X. Jerry also tells me that they'll have a Micropolis hard disk CP/M system in less than six months.

.....

SOFTWARE DIRECTORY

As stated previously, one of the purposes of this newsletter is to document the software available for Micropolis systems. I received an ad from Micro-Serve, Inc. (PO Box 482, Nyack NY 10960) for a

"Software Vendor Directory". At $46 plus $25/6-months for updates, it may be a bit expensive for the average computerist. After talking with Joan Mc Daniel, Micro-Serve's president, I decided that there's probably enough info in the directory to benefit us so I've ordered it. Ms. Mc Daniel has no objections to reproduction and distribution of the applicable sections in the MUG newsletter.

.....

LETTERS

Dear MUG:

Thanks for your hard work. I'm enjoying your output immensely. Although you probably won't hear from me often, you can be sure that I review your material, and I appreciate it.

Regarding the MICROPOLIS NEWS: As you commented, it was adequate. But, why the 11x15 paper size? Don't the editors think their product will be worth saving in a binder?

Regarding M. Rothstein's question about keeping information across program LOADS: He might enjoy making use of the 42-byte BASIC "Sign-On" message space (my machine address 16R2F7A-16R2FA3). That area is displayed only once, after loading BASIC (never, if the application program is autoloaded), and it is not usually disturbed as long as BASIC remains in memory.

If 42-byte A$ is to be kept across a program LOAD:

```
900     FOR N%=0 TO 41
901     POKE(N%+16R2F7A)=ASC(MID$(A$,N%+1,1))
902     NEXT N%
```

If you are closing down the system, or loading an overlay program:

```
990     SAVE "DATA" 16R2F7A, 16R2FA3
```

In the first application program after reloading BASIC:

```
050     LOAD "DATA"
```

Now, and after LOADing another application program:

```
100     DIM A$(42)
101     A$="": FOR N%=16R2F7A TO 16R2FA3
102     A$=A$+CHAR$(PEEK(N%))
103     NEXT N%
```

A$ is now the constant data you stored.

Jerry Factor
709 No. Palm Drive, Beverly Hills, CA 90210

Jerry-
That's a neat idea. After giving thought to it, some other applications of your principle come to mind. A lot of us have a PROM/RAM board

Title: TX.CRUNCH

```
  10       S$=CHAR$(11):IFPEEK(16R500)=79S$=CHAR$(4)
  50       DIMW$(250),V$(49,250):STRING"^":B$=" INPUT ":GOSUB860:GOSUB700
           :OPEN1F$:B$=" OUTPUT ":GOSUB860
 190     > INPUT"Does the output file presently exist? (Y or N):";R$:IFR$
           <>"Y"ANDR$<>"N"PRINT:PRINT"ANSWER WITH A 'Y' or 'N'!":PRINT:GO
           TO190
 210       IFR$="N"GOTO300
 220       GOSUB700:GOTO350
 270       REM
 300     > INPUT"Enter Name of OUTPUT File:";F$:F$="N:"+A$+":"+F$
 350     > OPEN2F$
 360     > INPUT"Enter # of 1st record to be transferred:";R$:GOSUB1010:I
           FA>0PRINT:PRINT"INPUT MUST BE ALL NUMERIC!":PRINT:GOTO360
 390       Y=VAL(R$):IFY<1ORY>RECPUT(1)-1PRINT:PRINT"1ST RECORD MUST BE >
           0 and <";RECPUT(1):PRINT:GOTO360
 410     > PRINT"Enter # of last record to be transferred,":INPUT"or '999
           9' if remainder of file:";R$:GOSUB1010:IFA>0PRINT:PRINT"INPUT
           MUST BE ALL NUMERIC:":PRINT:GOTO410
 450       Z=VAL(R$):IFZ>RECPUT(1)-1Z=RECPUT(1)-1
 470       IFZ<YPRINT:PRINT"LAST RECORD MUST BE >";Y-1:PRINT:GOTO410
 480       Q=0:FORI=YTOZ:GET1RECORDIW$:V$(Q)=W$:Q=Q+1:IFQ=50GOSUB930
 570       NEXTI:IFQ=0GOTO600
 590       GOSUB930
 600     > CLOSE1:CLOSE2:PRINTS$:PRINT"Transferred record";Y;"thru";Z;"fo
           r a total of";Z-Y+1;"records.":STRING",":END
 700   >*> PRINTS$:PRINTTAB(10);"THE FOLLOWING FILES ARE AVAILABLE:":PRIN
           T:F$=A$+":DIR":DISPLAYF$:PRINT:PRINT"If desired file is not li
           sted, insert":PRINT"another disk, type 'X', press RETURN.":PRI
           NT:PRINT"Enter Name of";B$;:INPUT"File:";F$:IFF$="X"GOTO700
 810   <*  F$=A$+":"+F$:RETURN
 860   >*  PRINTS$
 870    *> PRINT"Enter Drive # for";B$;:INPUT"File; (0-1):";A$:IFA$<"0"OR
           A$>"1"PRINT:PRINT"DRIVE SELECTION MUST BE '0' or '1'!":PRINT:G
           OTO870
 890   <*  RETURN
 930   ><  FORR=0TOQ-1:PUT2V$(R):NEXTR:Q=0:RETURN
1010   >*  A=0:FORI=1TOLEN(R$):IFMID$(R$,I,1)<"0"ORMID$(R$,I,1)>"9"A=A+1
1040   <*  NEXTI:RETURN
```

CRUNCH compacted the program from 2514 bytes to 1239 bytes, a savings of 1275. In addition to saving memory space, execution speed will increase.

```
Cross-Reference of TX

XREF element            Referenced at:
============            ==============


A                          380    440   1010   1030

I                          520    530    570   1020   1030   1040

Q                          510    540    550    560    580    930    960

R                          930    940    950

Y                          390    400    470    520    630

Z                          450    460    470    520    630

A$                         310    730    810    870    880

B$                         100    170    790    870

F$                         130    300    310    350    730    740    790
                           800    810

R$                         190    200    210    360    390    420    450
                          1020   1030

S$                          40    620    700    860

W$                          50    530    540

V$(49,250)                  50    540    940

File #1                    130    400    460    530    600

File #2                    350    610    940

  190  GOTO                200

  300  GOTO                210

  350  GOTO                260

  360  GOTO                380    400

  410  GOTO                440    470

  600  GOTO                580

  700  GOTO                800

  700  GOSUB               120    250

  860  GOSUB               110    180

  870  GOTO                880

  930  GOSUB               560    590
```

```
Cross-Reference of TX

XREF element            Referenced at:
============            ==============


 1010  GOSUB              370    430

AND                       200

CHAR$                      40

CLOSE                     600    610

DIM                        50

DISPLAY                   740

END                       650

FOR                       520    930   1020

GET                       530

GOSUB                     110    120    180    250    370    430    560
                          590

GOTO                      200    210    260    380    400    440    470
                          580    800    880

IF                         40    200    210    380    400    440    460
                          470    560    580    800    880   1030

INPUT                     190    300    360    420    790    870

LEN                      1020

MID$                     1030

NEXT                      570    950   1040

OPEN                      130    350

OR                        400    880   1030

PEEK                       40

PRINT                     200    380    400    410    440    470    620
                          630    700    710    720    750    760    770
                          780    790    860    870    880

PUT                       940

RECORD                    530

RECPUT                    400    460

RETURN                    820    890    970   1050

STRING                     60    640
```

Cross-Reference of TX

XREF element	Referenced at:						
TAB	710						
TO	520	930	1020				
VAL	390	450					
+	310	550	630	730	810	1030	
-	400	460	470	630	930		
=	40	100	170	210	310	390	450
	460	510	520	540	550	560	580
	730	800	810	930	960	1010	1020
	1030						
<>	200						
>	380	400	440	460	880	1030	
<	400	470	880	1030			

as part of the base operating system. The nominal 1K RAM isn't used (except for maybe the last dozen bytes in some systems) after MDOS/BASIC gains control.

To apply this memory to your program, one could substitute it's starting address for your 16R2F7A and starting-address-plus-42 bytes for your 16R2FA3. By changing the end-address to be start + 250, dimensioning A$ to 250, and changing the loop bounds to 0-249, one can save a few more characters. You can also use the same principle with the end of your contiguous memory if deallocated from BASIC with a MEMEND.

The next couple of tricks to figure out are how to save and restore numeric and text arrays, or maybe even operate on the deallocated memory directly (Ah-ha, I think I see the kernal of a word processor). How about it, MUGgies? Send me your solutions.

.....

POSTSCRIPT

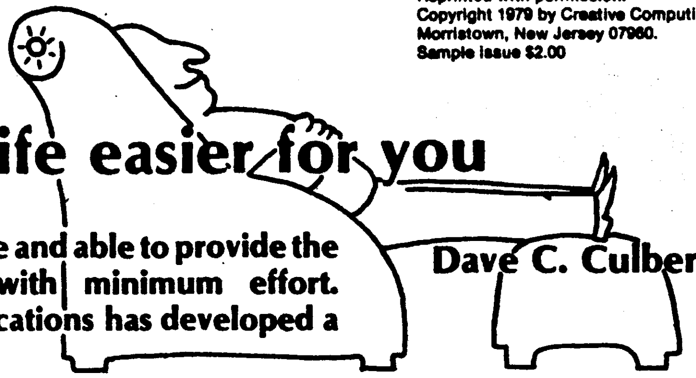
I assume you read the magazines and have seen the Micropolis sale that Priority One Electronics (16723 Roscoe Blvd., Sepulveda CA 91343, 1-800-423-5633, or in CA, AK, HI - 213-894-8171) is having. MOD II 1053's at $995 (list $1895). Other units also on sale, but not at quite the same savings.

Also, if any of you have done any work on terminals for the blind, I'd appreciate hearing from you.

11/1/80

A Data Base System With A Purpose

Making life easier for you



Data base systems should be easy to use and able to provide the user with valuable information...with minimum effort. Apparently Creative Computer Applications has developed a system which does just that.

Dave C. Culbertson

Relatively few people can accurately define a data management system. This type of program could be very important to an individual as well as any business. We all have to list or catalogue items but the problem is that the subject of these lists will vary. Many programmers make a valiant effort in trying to write individual programs to cover most of the routine needs they have. Unfortunately, there seems to be no end to the number of different types of items you wish to keep track of.

Creative Computer Applications has supplied the solution to this big headache. Their Data Management System is a dream come true. The program is written from the programmers point of view. It allows you to customize the number of data inputs as well as the names of the entry prompts, without rewriting program lines. These data prompts will also be used later in the program to output report titles or to allow you to sort the data into categories using your data as the key. The documentation is supplied in the form of a binder and is written in a well organized manner.

System Features

Presently this Data Management System is supplied on a single floppy diskette in the Micropolis "Mod II" format (a CP/M version will soon be available).

Now, to the specifics of this system. The program is supplied as a number of individual program segments. However the only program which the user must load and RUN is the first one, FMAINT. This is a menu program which automatically executes any of the required segments as they are needed. There are six choices; Termination of Processing, File Maintenance, Report Generation, File Compacting, File Defination and Sorting. With the combination of all of these functions put into one neat package and by comparing this to the same capabilities available in the CP/M format, there is a substantial cost savings in this system. You may use the program with either Micropolis' Basic Version 3.0 or 4.0. Due to the use of the chain function, I found that I could not list after running the Report Generator. It turned out to be due to a small bug in The Micropolis Basic. I had not sent in my Software Registration card and so Micropolis had not sent me the software update. A good lesson for all of us!

A Practical Application

In order to use this program and provide a better explanation of its functions, let us imagine that you have a small retail appliance store and wish to catalogue an up-to-date status on your customers as they bring in repairs. Your first task will be to use function 5 to Define the File. You may create a new file name, delete a name which already exists, list the file names which the system is aware of, or list the particular information for that file. To start with we will call the Create function, this puts the name of our new file into the system. We now must tell the system the particulars of this file. Since this is primarily a "Report Orientated" program, think of a piece of paper with a series of columns at the top of the paper, you must suggest a field I.D. which is an abbreviation of the name (1-5 characters). In our example, Date is the first column and we input DA for the I.D. code, Date for the field name, 7 as the tab position for the start of the field and 8 as the length of our field. This will allow the date to occupy positions 7 to 15 on our paper headings. We continue in a like manner for each heading we need. The system will then output to the terminal or your line printer a neat listing of all of this information. The system permits up to 24 column assignments. You are now ready to begin data entry. The menu will come back on the screen and allow you to select the file maintenance section. This section actually puts the name of the file on the disk and then allows you to add data, delete data, update any one section of a file, inspect a record, or scan the file for specific data. If you select the add function, a new record will be written at the end of the file. The program will begin to request each of the prompts you input in the previous step. If any prompt does not apply, you need only depress the return key. No data will be input for that prompt and the next one will appear on the screen. If you wish to change only one field you may use the update function. This works in a different manner. It requests the I.D. code and the data. This is sligthly inconvenient if you cannot remember the I.D. code but then you were the person who input this code in the first section. The delete function does not actually wipe out the entire record, it only flags this record for deletion. If you call this record, a message will be output to the screen saying "This is a deleted record." You may recover this condition by updating any one field. The scan function is a form of instant information retrieval. You could request the name of a customer, as in our example, and the program will search for the occurrence of this name through each record in the file. The number or numbers will be output to the screen. If your file is extensive this process may take a minute or so, as the program must look at every record. The inspection process allows you to call for the contents of that record, such as viewing a record number which the scan section has output.

After you have finished the input or posting of your data for that day, you have several choices for manipulation of the data. You may wish to print a report minus the deleted records. You may print a report with all the records, even the ones flagged for deletion. You may wish to sort the data in some manner or you may compact the entire file. We shall elect to compact our file. This process removes the deleted records and tightens up our file. Example: Suppose our file has 10 records and #8 has been flagged for deletion. The file will be the same for the first 7 records, #8 will be gone, #9 becomes #8 and #10 becomes #9. We have several choices in this section, we can compact the file in place, with or without a back-up file, or create a new file under a different name. This process can take several minutes to perform if the file is large.

Now you may wish to sort the file into useable data. The counter personnel at your store may wish to have the list of customer names in alphabetical order. No problem for this system. You can sort on any field in ascending or descending order. The program can perform a full alphanumeric merging sort on any field or fields which you select. This means that if the boss walks in and says that he would like a report on the repairs which are ready and he would like a report on the item brands, with the most expensive repairs for each listed first, you have the capability to perform this function. All you have to do is to select the field for brand in ascending order and the field for total cost in descending order. Then print out your boss's request. This is one of the fastest, most versatile sorting routines I have ever seen, written in a Basic language.

We have now progressed to the object of all of our data posting, compacting and sorting; the printing out of our report. This is where many programs fail badly to provide the needed capabilities. This programming system is the exception to this rule. You input your title, width of your printer, depth of your paper, force (which designates whether or not to print out deleted records) and type (report form or printing of mailing labels). These are designated as the run options. You then have a choice as to which columns you want printed and in what order they will appear on your report. There is also an option to shorten or truncate the length of the individual fields. The report printed will be paginated with the title input for each page and a header line which contains the column heading. A useful option is the subtotal and total section. This allows a subtotal to appear for each company grouping or on any field you designate. You also have the option to specify the number of decimal places (up to 5 places). If this option is used, the subtotals will be printed on the following line in the report with a lined border the width of the page. The grand total will appear at the end of the report in a like manner.

The mailing labels section of this report generator performs in a similar manner, permitting the same versatility of printing as we found in the report form printout. The only critique I have for this section would be the inability to use multiwidth labels.

The Documentation

The documentation supplied with this system comes in two forms. First, you are supplied with a binder which contains complete information on every phase of the system. In using the system the programmer will note that each message output to the terminal will have a number assigned to the beginning of the line. In the book is a listing by number of every message, the reason for the message, your action required and the program results. This makes use of the programming system very easy. Also included is a sample execution log at the end of the binder. The system is also supplied with an inventory program. The format is written in a step-by-step self-teaching mode. This permits you to start using the system immediately although I would suggest you read the documentation to be able to make use of the variety of functions available.

Finally, a page in the documentation is provided to permit users to share suggestions for improving this Data Management System. If you acquire a copy of the system, I urge you to share those suggestions. This is one of the best Data Management Systems I have seen to date. With input from programmers across the country, it could become the best data management system ever written. ■

--------------- *End Of Article* ---------------

```
Available from:  Custom Electronics Inc.
                 238 Exchange Street
                 Chicopee Mass. 01013
                 Phone 413-592-4761
```

CCA DATA MANAGEMENT SYSTEM

For Micropolis - Mod II

You've **got** your computer - **now** put it to use!

The CCA DMS lets you define, maintain, sort and produce reports and labels for any data files you need. Applications for the CCA DMS are limited only by your imagination.

SAMPLE APPLICATIONS

BUSINESSMEN	ATTORNEYS	DOCTORS
Inventory	Time spent on cases	Patient history
Customer lists	Library of documents	Billing
A/R - A/P	Billing	Supply vendors
Payroll	Name & Address	Appointments

ONLY $150.00 — EXTENSIVE DOCUMENTATION