

THE 80 NOTEBOOK

AUGUST 1980

ISSUE #5

.....

NOTE: The term "TRS-80" is a registered trademark of Radio Shack, a Division of Tandy Corporation. THE 80 NOTEBOOK is not affiliated with Radio Shack or Tandy Corporation in any way.

.....

"WHAT IS DATA BASE MANAGEMENT AND HOW CAN YOU MAKE USE OF IT?"

The term data base management refers to the creation, maintenance and inquiry to information collectively stored in a file. This file could be located on cassette tape or on a floppy disk.

For the purposes of this article, we will be considering that our data base file is located on cassette tape so that a normal 16K Level 2 BASIC system could use the concepts to be explored.

For those of you with little computer experience, let's look at some rudimentary terminology concerning our subject:

Think of a file as a filing cabinet in an office. If we examine the drawers of that filing cabinet, we will find records. Each record in the file would be like a folder in the drawers of the filing cabinet. Each record contains one or more pieces of information called fields. Each field would be like a document in an individual folder.

Now let's look at an example of what we are discussing:

In our example, we have a file of inventory items to be sold by a store. Each record in a file like this would represent one of the items sold by the store.

The fields we might expect in each record would be - an item number (an identification number or catalog number perhaps), item description text, selling price, number of items on hand, and so on.

Any and all information needed about an item must be represented by fields in each record. Even if only certain items in our file required a certain field, every record in the file must contain that field. The arrangement, number and type of fields in each record must be the same - a uniform file format.

This is only one example of what a data base file might contain. The uses of such a file are endless. Any collection of items or people on which statistics may be kept can be maintained in a data base file.

AUGUST 1980

PAGE 1

Now let's look at the kind of program needed to create, maintain and examine a data base file of this type:

Such a program must have several functions which could be selected by a menu routine. This routine would display the names of these functions along with a selection code and then ask the user which function he desires.

Once this selection has been made, the routine would pass control to the appropriate function routine.

These functional routines would perform the following tasks: 1) initialize a file, 2) read a data base from tape, 3) find a record, 4) display a record, 5) change a record, 6) reformat a file, 7) sort a file, 8) write a data base to tape, and 9) report a file on the printer.

Here is the BASIC code which would carry out this selection process:

```
100 PRINT "SELECTION MENU"
110 PRINT "1) INITIALIZE A FILE"
120 PRINT "2) READ A DATA BASE FROM TAPE"
130 PRINT "3) FIND A RECORD"
140 PRINT "4) DISPLAY A RECORD"
150 PRINT "5) CHANGE A RECORD"
160 PRINT "6) REFORMAT A FILE"
170 PRINT "7) SORT A FILE"
180 PRINT "8) WRITE A DATA BASE TO TAPE"
190 PRINT "9) REPORT A FILE ON THE PRINTER"
200 PRINT "10) END": PRINT
210 INPUT "PLEASE ENTER YOUR SELECTION"; I
220 ON I GOTO 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000
230 END
```

Now let's take a look at these functions individually in more detail:

The initialization of a file is required to use or create any file under control of this system. Whenever a file is manipulated by this system, it is done so from a table in memory where the file is stored during the manipulation. This table has an entry for each record in the file (or potential or future record in that file).

Before a file could be read into this table in memory or created for the first time, the system must be told the maximum number of records the file will or does contain. This number must allow for the necessary record room for any future record adds to an existing file.

In BASIC, tape records have a maximum allowable record length. Because of this, our data base system will allow a maximum record length of 250 characters.

Once the system knows how many records are to be allocated to the current file, the routine will set up a table of N number of 250 character blank records.

Each blank record in the table denotes an unused record position in the file which can be filled by either a tape record read or a record change function.

When a file is eventually written to tape by the appropriate function, any unused record positions will be treated as deleted records and will not be written to tape.

Here is the BASIC code which would handle this function:

```
1000 INPUT "HOW MANY RECORDS ARE IN THIS FILE"; K
1010 J=(K*270)+1000
1020 CLEAR J
1030 J=FRE(B$)
1040 N=(J-1000)/270
1050 DIM A$(N)
1060 FOR I=0 TO N
1070 A$(I)=STRING$(250," ")
1080 NEXT I
1090 GOTO 100
```

Note the use of the FRE function in this routine to retrieve the total number of available string space characters. Once the string space is computed from the number of records in the file and the string space is "CLEAR"-ed, the values in K and J are lost.

Because of this, the FRE function is used to get back the J value, and from that, the number of records in the file can easily be computed so that the DIM statement can be executed to set up the file table requested.

Once the file table has been initialized, if a previously created data base file is to be manipulated, it must be read into the file table. This can be handled by the second function in our data base program.

Should the number of records on the tape exceed the file table size specified in the initialization function, the system must abort the tape read operation. If this does happen, you must re-execute the initialization function before continuing.

Here is the BASIC code which would read a data base file from a tape into the memory table:

```
2000 K=0
2010 INPUT #-1,B$
2020 IF B$="/*" THEN 100
2030 K=K+1
2040 IF K.GT.N THEN 2080
2050 IF LEN(B$).NE.250 THEN 2110
2060 A$(K)=B$
2070 GOTO 2010
2080 PRINT "YOU HAVE EXCEEDED YOUR FILE TABLE SPACE"
2090 PRINT "YOU MUST RE-INITIALIZE THE FILE TABLE SPACE"
2100 GOTO 1000
2110 PRINT "RECORD LENGTH IS NOT 250"
2120 PRINT "THIS CAN NOT BE A DATA BASE TAPE FILE"
2130 IF K.GT.1 THEN 2090
2140 GOTO 100
```

Note the use of a "/" record as the last or end of file record on our data base file tape format. When this record is sensed, the routine returns to the function menu routine.

Also, the routine checks the record length to insure that the tape being read is a data base file tape. If a record length other than 250 occurs, the routine will request that the file table initialization function be executed if the condition occurs on other than the first record of the tape file read operation.

After records have been established in the file table, you may wish to locate a specific record in the file using a predefined search argument. Since your quest for a record may require the examination of several fields, this function must set up a table of field positions and search values.

Since the system should set some limit on the number of fields to be examined in the search, this routine will allow up to 20 fields to be specified in the search.

The following is the BASIC code needed to set up the field definition tables to be used in the record search function and in other functions to be discussed later:

```
1090 DIM X(20),Y(20),V$(20),Z(20)
1100 FOR I=0 TO 20
1110 X(I)=0
1120 Y(I)=0
1130 V$(I)=" "
1140 Z(I)=0
1150 NEXT I
```

With this in mind, let's take a look at how this field definition table system works. The X table holds the starting position of the field in the record. Therefore, this value would be in the range 1 to 250. The Y table holds the size of the field being defined. This value must be from 1 to 250.

The result of the starting position in X + the field size in Y minus 1 must never exceed 250 because this would indicate a field size going outside of the record.

The Z table indicates if a field is numeric by a value of 1. Numeric fields are stored in a record in their string form.

Now let's look at the BASIC code needed to initialize the field definition table with the desired search arguments:

```
3000 K=0
3005 S=0: T=0: R=0: B$=""
3010 INPUT "FIELD STARTING POSITION FROM 1 TO 250"; S
3020 IF S=0 THEN 3300
3030 IF S.LT.0 OR S.GT.250 THEN 3010
3040 INPUT "FIELD SIZE"; T
3050 IF T.LT.1 OR T.GT.250 OR S+T-1.GT.250 THEN 3040
```

```

3060 INPUT "ENTER 0 FOR TEXT FIELD, 1 FOR NUMERIC FIELD"; R
3070 IF R.LT.0 OR R.GT.1 THEN 3060
3080 K=K+1
3090 X(K)=S
3100 Y(K)=T
3110 Z(K)=R
3115 IF R=1 THEN 3140
3120 INPUT "VALUE TO BE SEARCHED FOR"; B$
3130 GOTO 3160
3140 INPUT "VALUE TO BE SEARCHED FOR"; H
3150 B$=STR$(H)
3160 G=LEN(B$)
3170 IF G.GT.T OR G.LT.1 THEN 3115
3175 IF G.LT.T THEN B$=B$+STRING$(T-G," ")
3180 V$(K)=B$
3190 IF K.LT.20 THEN 3005
3200 GOTO 3300

```

Note that in the above routine, the field search value must be within the field size constraint. Smaller text field values are padded with blanks.

Now that we have our search arguments, we can take a look at how the search is carried out. To make the search as flexible as possible, the routine will ask for a starting record number from 1 to N, where N is the file table size.

This will allow the user to search for more than one record with the desired search arguments by repeating a search starting at the next highest record number from the last record matched in such a search.

Here is the BASIC code needed to carry out this search function:

```

3300 INPUT "STARTING RECORD NUMBER"; J
3310 IF J=0 THEN 100
3320 IF J.LT.0 OR J.GT.N THEN 3300
3330 FOR I=J TO N
3335 F$=A$(I)
3340 FOR M=1 TO K
3350 S=X(M)
3360 T=Y(M)
3370 R=Z(M)
3380 B$=V$(M)
3390 IF R=1 THEN H=VAL(B$): GOTO 3410
3400 IF B$.NE.MID$(F$,S,T) THEN 3460 ELSE 3430
3410 P$=MID$(F$,S,T)
3420 IF H.NE.VAL(P$) THEN 3460
3430 NEXT M
3440 PRINT "MATCH AT RECORD NUMBER"; I
3450 GOTO 3300
3460 NEXT I
3470 PRINT "NO MATCH"
3480 GOTO 100

```

Notice that a response of zero to the starting record number prompt will end the search mode and return the system to the menu routine.

Once a matching record has been found, that record's record number or position in the data base file table is printed. Note the use of the VAL function to retrieve the numbers from numeric string fields and numeric search arguments to be used in making comparisons.

Once a record has been located and its record number is known, the user will want to display that record on the screen. The display function will ask for the desired record's location number and will display the record along with a scale to allow the user to easily find individual fields on a position by position basis.

Here is the BASIC code which will display a record:

```

4000 INPUT "DESIRED RECORD NUMBER"; J
4005 IF J.LT.0 OR J.GT.N THEN 4000
4010 IF J=0 THEN 1000
4020 PRINT STRING$(9,"0"); STRING$(10,"1"); STRING$(10,"2"); STRING$(10,"3")
; STRING$(10,"4"); STRING$(10,"5"); STRING$(4,"6")
4025 FOR I=1 TO 6
4030 PRINT "1234567890";
4035 NEXT I
4040 PRINT "123"
4050 F$=A$(J)
4060 PRINT MID$(F$,1,63)
4070 PRINT STRING$(36,"0"); STRING$(27,"1")
4080 PRINT STRING$(6,"6"); STRING$(10,"7"); STRING$(10,"8"); STRING$(10,"9")
; STRING$(10,"0"); STRING$(10,"1"); STRING$(7,"2")
4090 FOR I=1 TO 6
4100 PRINT "4567890123";
4110 NEXT I
4120 PRINT "456"
4130 PRINT MID$(F$,64,63)
4140 PRINT STRING$(63,"1")
4150 PRINT STRING$(3,"2"); STRING$(10,"3"); STRING$(10,"4"); STRING$(10,"5")
; STRING$(10,"6"); STRING$(10,"7"); STRING$(10,"8")
4160 FOR I=1 TO 6
4170 PRINT "7890123456";
4180 NEXT I
4190 PRINT "789"
4200 PRINT MID$(F$,127,63)
4210 PRINT STRING$(10,"1"); STRING$(51,"2")
4220 PRINT STRING$(10,"9"); STRING$(10,"0"); STRING$(10,"1"); STRING$(10,"2")
); STRING$(10,"3"); STRING$(10,"4"); "5"
4230 FOR I=1 TO 6
4240 PRINT "0123456789"
4250 NEXT I
4260 PRINT "0"
4270 PRINT MID$(F$,190,61)
4280 PRINT "HIT THE X KEY TO CONTINUE";
4290 B$=" "
4300 B$=INKEY$
4310 IF B$.NE."X" THEN 4290
4320 PRINT " "
4330 GOTO 4000

```

Note that the desired record number prompt will allow the user to enter zero to return to the selection menu routine.

When the display of the currently selected record is completed, the user must hit the X key to allow the system to continue. The X key will return the user to the record number prompt to allow the user to display another record.

Once we have examined a record through the display function, some of the information in that record may require some changes. To do this, the change function allows the user to modify fields in a record by specifying the field starting position, field size and the new value for that field. This activity can be continued for as many field changes as desired in a record and for as many records requiring changes before returning to the menu routine.

Here is the BASIC code needed to perform these field changes:

```
5000 INPUT "DESIRED RECORD NUMBER"; J
5010 IF J=0 THEN 1000
5020 IF J.LT.0 OR J.GT.N THEN 5000
5030 F=A$(J)
5040 INPUT "FIELD STARTING POSITION"; S
5050 IFS=0 THEN A$(J)=F$: GOTO 5000
5060 IF S.LT.0 OR S.GT.250 THEN 5040
5070 INPUT "FIELD SIZE"; T
5080 IF T.LT.1 OR T.GT.250 OR T+S-1.GT.250 THEN 5070
5090 PRINT "FIELD WAS"; MID$(F$,S,T)
5100 INPUT "NEW FIELD VALUE"; B$
5110 H=LEN(B$)
5115 IF H.LT.1 OR H.GT.T THEN 5100
5120 IF H.LT.T THEN B$=B$+STRING$(T-H," ")
5130 P$=""
5140 IF S.GT.1 THEN P$=MID$(F$,1,S-1)
5150 P$=P$+B$
5160 IF S+T-1.LT.250 THEN P$=P$+MID$(F$,S+T,250-(S+T-1))
5170 F=P$
5180 GOTO 5040
```

When entering the desired record number, if the user enters a zero, the change routine will return to the menu routine.

If the user enters a zero to the field starting position prompt, the change routine will allow the user to return to the desired record number prompt to select another record to be changed.

Since the user is specifying the starting field positions and field sizes, it is important that you plan the field layout of your records carefully so that there are no overlapping fields in your record layout. Remember - if you are going to search, sort, reformat or report your file, every record must have the same record layout of fields!

When dealing with numeric fields, you must plan the field sizes to be large enough to hold the string form of a number along with its leading blank, sign character, decimal point, maximum number of digits possible in each field, and possibly the exponent notations characteristic of a floating point number if you are using such numeric information.

When you wish to insert a new record in the file, use the find function to locate a blank record - an unused record position to be used for your new record. Be sure that as you fill the fields of a new record with information that at least one of those fields will always contain some non-blank text regardless of any of the possible data changes that may occur in the future of that record so that the record will never be completely filled with blanks and considered an unused record position.

This can be done by including a record identification field when planning your record layout. This field would be a one character field containing a non-blank character which is never altered unless the record is deleted.

To delete a record from the file, simply fill the entire record with blanks. The easiest way to do this is to enter a field change treating the record as one 250 character field and entering a blank for the new field value.

When entering new field values, it is not necessary to input a value with exactly the same number of characters as the field it is for. The value length must not exceed the field size but if it is less, the system will pad the field to the right with blanks.

When you wish to blank out a field, you need only enter a single blank as the new value because of this.

In the course of maintaining a data base file, a time may come when your original record field layout becomes inadequate for one reason or another. This may happen because of the need to sandwich a new field between two existing fields in a file layout.

To do this, the reformat function allows the user to rearrange the layout of fields in each record allowing blank areas where desired.

The first operation this function will carry out is to determine how to move the existing fields around into the new record format. For this, the user must input each field's starting position in the current record format, the field size and the starting position in the new record layout.

Once this is known, the actual reformatting can be carried out on the file table records. Since this operation uses the X, Y and Z tables discussed earlier, the maximum number of field redefinitions is limited to 20.

Here is the BASIC code needed for this function:

```
6000 K=0
6010 INPUT "FIELD STARTING POSITION"; S
6020 IF S=0 THEN 6300
6030 IF S.LT.0 OR S.GT.250 THEN 6010
6040 INPUT "FIELD SIZE"; T
6050 IF T.LT.1 OR T.GT.250 OR T+S-1.GT.250 THEN 6040
6060 INPUT "NEW STARTING POSITION"; R
6070 IF R.LT.1 OR R.GT.250 OR R+T-1.GT.250 THEN 6060
6080 K=K+1
6090 X(K)=S
```



```

6100 Y(K)=T
6110 Z(K)=R
6120 IF K.LT.20 THEN 6010
6300 FOR I=1 TO N
6310 P$=STRING$(250," ")
6320 F$=A$(I)
6330 FOR J=1 TO K
6340 S=X(J)
6350 T=Y(J)
6360 R=Z(J)
6370 B$=""
6380 IF R.GT.1 THEN B$=MID$(P$,1,R-1)
6390 B$=B$+MID$(F$,S,T)
6400 IF R+T-1.LT.250 THEN B$=B$+MID$(P$,R+T,250-(R+T-1))
6410 P$=B$
6420 NEXT J
6430 A$(I)=P$
6440 NEXT I
6450 GOTO 100

```

Again, when the user enters zero to the field starting position prompt, this signals the user has finished entering the field redefinitions for his new file record format with less than 20 field redefinitions in total.

This function should be preceded by the file initialization function followed by data base file read function. After the reformat function is completed, the file should be written to tape by the data base file tape write function.

In the course of displaying your data base file records, before using the report file function and possibly before writing a data base file to tape or after a reformat function, you may wish to sort your file into a particular sequence. This can be done using the sort function.

To do this, using the X, Y and Z tables, the user will specify the fields on which the file is to be sorted by entering the starting position, field size and sort type of each sort field required up to a maximum of 20 fields.

The sort types allowed by the routine are 1) ascending text field, 2) descending text field, 3) ascending numeric field and 4) descending numeric field.

Here is the BASIC code needed to input the sort arguments and perform the actual sort. As before, entering a zero to the field starting position prompt will signal the end of the sort argument input phase.

```

7000 K=0
7010 INPUT "FIELD STARTING POSITION"; S
7020 IF S=0 THEN 7300
7030 IF S.LT.0 OR S.GT.250 THEN 7010
7040 INPUT "FIELD SIZE"; T
7050 IF T.LT.1 OR T.GT.250 OR S+T-1.GT.250 THEN 7040
7060 INPUT "SORT TYPE"; R
7070 IF R.LT.1 OR R.GT.4 THEN 7060
7080 K=K+1

```

```

7090 X(K)=S
7100 Y(K)=T
7110 Z(K)=R
7120 IF K.LT.20 THEN 7010
7300 FOR J=1 TO K
7310 S=X(J)
7320 T=Y(J)
7330 R=Z(J)
7335 L=0
7340 FOR I=1 TO N
7350 B=A$(I)
7360 F=A$(1): M=1
7370 IF I.LT.N THEN B=A$(I+1): M=I+1: F=A$(I)
7380 ON R GOTO 7400, 7500, 7600, 7700
7400 IF MID$(F$,S,T).GT.MID$(B$,S,T) THEN P=A$(I): A$(I)=A$(M): A$(M)=P:
L=1
7410 GOTO 7800
7500 IF MID$(F$,S,T).LT.MID$(B$,S,T) THEN P=A$(I): A$(I)=A$(M): A$(M)=P:
L=1
7510 GOTO 7800
7600 P=MID$(F$,S,T): G=VAL(P)
7610 P=MID$(B$,S,T): H=VAL(P)
7620 IF G.GT.H THEN P=A$(I): A$(I)=A$(M): A$(M)=P: L=1
7630 GOTO 7800
7700 P=MID$(F$,S,T): G=VAL(P)
7710 P=MID$(B$,S,T): H=VAL(P)
7720 IF G.LT.H THEN P=A$(I): A$(I)=A$(M): A$(M)=P: L=1
7800 NEXT I
7810 IF L=1 THEN 7335
7820 NEXT J

```

Remember - blank (deleted or unused) records are also sorted into order!

When you are entering sort field arguments, be sure you enter them in lowest to highest order of sorting priority. For example, if sorting on two fields, the first field defined for the sort would be the minor sort field and the second field defined for the sort would be the major sort field.

After a file has been changed, reformatted or sorted, it should be saved in its updated form on tape. This can be done by using the data base file tape write function. This function will drop all deleted records (unused, blank records) from the file as it is copied to tape.

Here is the BASIC code needed to perform this function:

```

8000 FOR I=1 TO N
8010 IF A$(I)=" " THEN 8030
8020 PRINT #-1, A$(I)
8030 NEXT I
8040 PRINT #-1, "/*"

```

We have one more function available to the user in our system - the file report function. Once a file has been initialized, read and optionally sorted, it can be listed on a printer in a formatted fashion.

This is done by giving the user an 80 character, blank print line to work with and allows him to select what fields from the file records he wishes to print and where in the print line they should be placed. Blank file records are ignored in this printing of the file.

Using the X, Y and Z tables, the first section of this routine will allow the user to enter the starting position within a file record, field size and the starting position of the field within the 80 character print line for each field desired to be listed up to a maximum of 20 fields.

Again, when the user enters a zero to the starting position within a file record prompt, this will signal that the user has finished entering his field selection definitions and the routine can perform the actual listing operation.

Here is the BASIC code needed to perform this function:

```

9000 K=0
9010 INPUT "ENTER FIELD STARTING POSITION WITHIN RECORD"; S
9020 IF S=0 THEN 9300
9030 IF S.LT.0 OR S.GT.250 THEN 9010
9040 INPUT "FIELD SIZE"; T
9050 IF T.LT.1 OR T.GT.250 OR T+S-1.GT.250 THEN 9040
9060 INPUT "FIELD STARTING POSITION WITHIN PRINT LINE"; R
9070 IF R.LT.1 OR R.GT.80 OR R+T-1.GT.80 THEN 9060
9080 K=K+1
9090 X(K)=S
9100 Y(K)=T
9110 Z(K)=R
9120 IF K.LT.20 THEN 9010
9300 FOR I=1 TO N
9310 F=A$(I)
9320 IF F="" THEN 9500
9330 P$=STRING$(80," ")
9340 FOR J=1 TO K
9350 S=X(J)
9360 T=Y(J)
9370 R=Z(J)
9380 B$=""
9390 IF R.GT.1 THEN B$=MID$(P$,1,R-1)
9400 B$=B$+MID$(F$,S,T)
9410 IF R+T-1.LT.80 THEN B$=B$+MID$(P$,R+T,80-(R+T-1))
9420 P$=B$
9430 NEXT J
9440 LPRINT P$
9500 NEXT I
9510 GOTO 100

```

While this is a simplified type of data base management system for the TRS-80, it does provide the most needed capabilities while using the least hardware and software resources.

Some of the suggestions we would have for possible improvements include: headings, totals and group breaks on reports, adding the capability to the search function to select records based on a range of values for a chosen field, and adding the capability to select records conditionally for acceptance during the

tape read function - this would allow the user to selectively print listings of portions of a current data base file, create sub-files from a larger data base file by writing portions of a data base file to a new tape, selectively create a reformatted sub-file from a larger data base file and selectively sort only the records you desire for a sub-file creation or listing.

If you have 32K or more and like a challenge, you might try devising the BASIC code needed to implement these improvements and any others you might come up with.

Whether you try this or not, we hope you will make valuable use of this compact little data base system and use its techniques in your other programming tasks requiring file manipulation.

JOIN THE 1000'S WHO ARE ENJOYING

THE 80 NOTEBOOK

THE MONTHLY JOURNAL FOR ALL TRS-80 USERS

WITH THESE EXCITING FEATURES AND DEPARTMENTS: (AND MORE!!)

- * SCIENTIFIC SOFTWARE
- * LETTERS TO THE EDITOR
- * PRACTICAL APPLICATIONS
- * ENTERTAINMENT PROGRAMS
- * WORD PROCESSING SYSTEMS
- * ARTIFICIAL INTELLIGENCE
- * SYSTEM UTILITY SOFTWARE
- * RADIO SHACK NEWS RELEASES
- * DATA BASE MANAGEMENT SYSTEMS
- * EDUCATIONAL AND CAI PROGRAMMING
- * SIMULATIONS AND COMPUTER MODELING
- * GRAPHICS AND ANIMATION TECHNIQUES
- * ASSEMBLY LANGUAGE PROGRAMMING LESSONS
- * LEVEL I, II AND DISK BASIC PROGRAMMING LESSONS
- * OPERATING SYSTEMS, LANGUAGES AND COMPILER DESIGN
- * PROGRAM LISTINGS (UP TO 24 PAGES IN EVERY ISSUE!!)
- * ARTICLES DEALING WITH UNUSUAL AND INTERESTING USES FOR YOUR TRS-80
- * GAMES
- * PUZZLES
- * CONTESTS
- * GAMBLING
- * NEW PRODUCTS
- * TRS-80 CLUB NEWS
- * PERSONAL FINANCE
- * SOFTWARE EXCHANGE
- * BUSINESS SOFTWARE
- * SORTING TECHNIQUES
- * FREE CLASSIFIED ADS
- * PRODUCT EVALUATIONS
- * ARTICLES ON HARDWARE

NOTE: The term "TRS-80" is a registered trademark of Radio Shack, a Division of Tandy Corporation, who is not affiliated with THE 80 NOTEBOOK in any way.

THE 80 NOTEBOOK
R. D. #3
Nazareth, PA 18064
Phone: 215-759-6873

NAME _____
ADDRESS _____

CHECK ONE: NEW _____ RENEWAL _____
1 YEAR SUBSCRIPTION: \$14 _____
2 YEAR SUBSCRIPTION: \$26 _____
CANADA/MEXICO: ADD \$5/ YEAR

3 YEAR SUBSCRIPTION: \$36 _____
SAMPLE COPY: \$2 _____ AIR MAIL \$4 _____
FOREIGN: ADD \$12/ YEAR

AVOID THE 1980 PRICE INCREASES - SUBSCRIBE TODAY!

THE 80 NOTEBOOK
R.D.#3
Nazareth, PA 18064

BULK RATE
U.S. POSTAGE
PAID
Stockertown, PA 18083
Permit No. 8

4/80

3/81