REMark

Issue 23 • December 1981

Official magazine for users of Heath computer equipment.

on the cover . . . .

Merry Christmas from the Heath Users' Group

on the stack

>CAT

"REMark" is a HUG membership magazine published ten times yearly. A subscription cannot be purchased separately without membership. the following rates apply.

	U.S. Domestic	Canada & Mexico	International
Initial	$18	$20 US FUNDS	$28
Renewal	$15	$17 US FUNDS	$22

Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

Back issues are available at $2.50 plus 10% handling and shipping. Requests for magazines mailed to foreign countries should specify mailing method and add the appropriate cost.

Send payment to:

Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

HUG Manager and Editor .... Bob Ellerton
Assistant Editor and
Software Developer .... Patrick Swayne
HUG Secretary .... Nancy Strunk
Software Developer .... Gerry Kabelman
HUG BB .... Terry Jensen

REMark

DBMS's in HDOS

When Heath computers were first introduced, computer systems were primarily created for and used by the hobbists. The hobbists had to become proficient programmers just to own their computers, because very little software was available. In essence, these beginning programmers "broke the trail" to creating the software products and utilities that we use today.

Experimenting with games was the earliest and most common use of microcomputers. As programs became more complicated, utilities for aiding the hobbists were developed and became popular because they made operating a computer system easier and more efficient. With the addition of more sophisticated utility programs, other uses for the microcomputers became more prevalent. No longer was it necessary to be an avid programmer to own and operate a micro.

The small business community caught on and began to see the use of the microcomputer for doing their accounting books, payroll, and maintaining an up-to-date inventory. Large businesses realized that word processing, done on their mainframes, used up valuable time and memory which could be accomplished just as effectively with micros.

Within the last year, the microcomputer price tag has become attractive to the home market. Besides the many popular games that are available, the introduction of home finance and bookkeeping programs has added utilities that are useful to the home. Now virtually anyone can put practical use to the microcomputer.

Software demands continue to be made from the microcomputer users. The programmer is now confronted with the user wanting an efficient means of handling large amounts of data, quickly and effectively. That in essence is what data management is. Database management is probably the microcomputer trend of the day, and is the subject that is to be discussed in this article.

DBMS Defined

A Data Base Management System (DBMS) is "the function of controlling the acquisition, analysis, storage, retrieval, and distribution of data", (American National Dictionary for Information Processing). In the Winter 1982 edition of CREATIVE COMPUTER BUYERS GUIDE, Glenn A. Hart, author of "CP/M Database Management Systems", gives a definition for DBMS as "a program system which manipulates collected data, facilitating information entry, updating, additions, deletions and the generation of useful reports". Simply stated a DBMS is any system that controls the handling of data.

Is that what an inventory program is? How about a business accounting package? Yes, but they are a specialized type of a DBMS. Those of you who are familiar with the SOURCE or CompuServe time-share systems should recognize them as two very large database management systems. There is quite a difference between an inventory package and the very large DBMS of the SOURCE or MicroNET but yet both are a type of database management system. This article will deal with DBMS's as they relate to the handling of data that is peculiar to existing and defined software products, such as an inventory or accounting package.

Each database management system has a distinct appearance to the user and each controls data differently. A few DBMS's are "keyword" oriented which, like the SOURCE, means that by entering the proper word or series of words the user will get the results he intended to see. Most are menu driven which, like CompuServe, make it easy for the user to use the DBMS without knowing anything about the system or computers in general. Some DBMS's are a combination of both.

The possible uses of DBMS's are about as varied as the number of DBMS's that are available: Libraries have incorporated a type of DBMS which will take a keyword that is entered and return the title of the books or articles that has reference to said topic. What about a means of logging all contacts and conversations that are made via amateur radio? Another example is a name and address file that not only gives the address information but any other bits of data about the individual or company that is pertinent to same. How about being able to enter any paragraph of information and having that paragraph returned at some future date just by entering a keyword, subject or category of the paragraph? A means of alphabetizing and sorting of a mailing and telephone list would give a nice option to the normal mailing program. Lastly, it would be nice to create a file of any data that is to be stored and retrieved as needed.

A Data Base Management System must have several principle parts or topics to be considered useful. All DBMS have variations between them but they must cover similiar ideas or options. The following list is an outline of the main areas that need to be considered before purchasing a DBMS package:

I. Human Engineering

II. Data Manipulation

A. Creating a file
B. Editing a file
C. Output of data

III. Language Used

IV. Documentation

V. Evalution of Strengths vs. Weaknesses

Heath Users' Group is releasing six database management systems in this issue of REMark. It is important to keep this outline in the back of your mind, while reading this article. The six DBMS's will be explained in detail, later on, as part of this article.

DBMS Outline Explained

I. Human Engineering

Human Engineering simply refers to the careful design of a program with the end user in mind. The DBMS should allow for simple training and ease of operation for the user of the program without reference to documentation or instructions. This means that an ideal DBMS will be self-documenting, clearly displayed, and in some way, menu driven.

The human design factor of a DBMS is extremely difficult to judge, because what is written as "ease of operation" or "clearly displayed" for one person is not to another. Programs that use the special features of the H19/H89 terminal, e.g. reverse video and the 25th line enabled to explain the position status, will greatly enhance the users ability to make his way through the DBMS.

II. Data Manipulation

When data is entered, changed, or printed then it can be said that the data has been manipulated. Is a DBMS really that simple? It can be, but not always. The way data is handled is so diversified, that two programs written in the same language, using the same data, could easily accomplish the same results, but yet handle the files differently.

It is beyond the scope of this article to explain the different ways of handling files. For information on sequential and random data file handling refer to "DBMS" issue 14 of REMark, page 10, by William N. Campbell, M.D.

A. Creating a file

The first step in the handling of data is obviously to create files for the storage and retrieval of information. Each DBMS may accomplish this differently, but somehow the DBMS must create a file.

A desirable feature of creating a file in a DBMS, is giving the user the option of defining the file structure, i.e. record and field size. Allowing the user to define the record and field size will greatly enhance the usefulness and capability of implementing the DBMS for use with other types of data. Another selectable item might include the ability to set the data field to alphanumeric or numeric fields and to be able to define them in a variety of formats. These user selectable options are not available on all packages.

B. Editing a file

Editing the files that are created by the DBMS is as important as creating the file. If data is entered incorrectly or needs to be updated, the modification must be accomplished smoothly and efficiently by editing the file. Updating, changing, adding and deleting are the obvious editing commands that must be available. Some DBMS's may have basic editing commands available to allow the user to edit specific fields of the file.

C. Output of data

The last option in the handling of a data file is the ability to produce useful and accurate reports of the stored information. The output will determine how the DBMS is to be used effectively, e.g. a telephone and mailing format could not produce an accurate list of articles obtained from a keyword but may be implemented to keep a logbook containing names, places and a brief description.

Several reports may be necessary to produce an accurate description of the data in the DBMS. Sorting routines can arrange the datafiles to display the data in a variety of formats. The important consideration of the reports is, will it meet the needs of the user.

III. Language Used

The language, of which the DBMS is written in, may be one of the most important determining factors in choosing among any number of database management systems. If the user is not a programmer and does not care to learn, then the DBMS language may seem trivial. However, it is not likely that one DBMS will meet everyone's needs without making some modifications to the software. By choosing a DBMS that has a familiar language, the user will be more comfortable with studying and modifying the existing programs.

The steps for choosing a DBMS based on the language used, include the following suggestions: determine the language of the DBMS and be sure which operating system it is written under. Then determine if the language needs an interpreter or compiler. If it is assembly language it will need neither but may need a special microprocessor to execute. DBMS's are all written differently and therefore it is important to understand this information to determine if the product is intended to run on a particular computer.

IV. Documentation

When anyone is unfamiliar with a software product, the documentation is there to help the user become comfortable with the product. The same is true with any database management system. If written properly the DBMS should need limited documentation on the use of the system, but may need an extensive explanation on how to modify the system for a particular use.

The documentation of the DBMS must be explicit but easy to read for the beginner. Examples of the menu's, if any; instruction of the use of special function keys, if used; and the explanation of the file structure all must be edited carefully for the reader. Any other special features of the DBMS should also be included as part of the documentation. It is most important that the use of the system be as self-documenting as possible.

V. Evaluation of Strengths vs. Weaknesses

Generally, DBMS's do not have a self-evaluation contrasting the strengths and weaknesses. When available, the evaluation will be helpful in determining which DBMS may meet the users needs. Whether or not an evaluation is included, the user is responsible for making a determination of the strengths and weaknesses as it relates to his needs. (Comments, by the reviewer of each respective DBMS, are included and are intended to be as objective as possible.)

DBMS from HUG

The following list of Data Base Management Systems is of newly released products of Heath Users' Group. There is a total number of six DBMS's contained on five disk Part Numbers (four under HDOS and one under CP/M). Under HDOS, one disk product has two Benton Harbor Basic DBMS's, a second has two Microsoft BASIC DBMS's, a third is one DBMS (two disk set) written using a combination of Z80 Assembly Language code (8080 modification included), BHBASIC and PASCAL, while the last HDOS database management system has been written in Z-80 Assembly Language code (only). The CP/M disk contains a data base management system in MBASIC, which is a modification to the Logbook DBMS written in BHBASIC. Here is the list:

885-1107 HDOS DBMS in BHBASIC
   Logbook: An Amateur radio operators logbook
   Transaction Data Management System (TMS): A General purpose DBMS

885-1108 HDOS DBMS in MBASIC
   Information System: A General purpose DBMS
   Telephone and Mail System:

885-1109 HDOS DBMS; Retriever: A Keyword oriented retrieval system

885-1110 HDOS DBMS; Autofile: A Filing and retrieval system in Z80 code

885-1214 CP/M DBMS; Logbook: An Amateur Radio operators logbook

The P/N, description and price will be included in the Price List on page 17 of this issue.

Each of the programs are released with no modifications or editing by Heath Users' Group. The author of each respective DBMS is responible for his work, including the programs and documentation. It has been my responsibility to review database management systems, in general, and these six submittals in particular, and then present the entire picture to you. It is my wish to introduce the DBMS submittals as the respective author has intended. You, as the purchaser of said products, will be responsible for making any changes or modifications to fit your personal needs.

None of the authors of the DBMS's were aware that their products would be introduced and released in this way, and therefore, each of the following abstracts were prepared within a reasonable and consistant format from their articles, letters and documentation. The abstracts, as presented, were written and edited for this particular article and have been followed as closely as possible to the author's submittal letter, article, or documentation. My apologies to any of the authors if the context has changed from what they originally intended. My thanks to each of the authors for their fine work and effort they put into their programs and documentation, giving HUG the depth of such fine Data Base Management Systems.

At the conclusion of each abstract, there will be a short commentary of the product including pertinent information, helpful hints, and, if necessary, a brief overview of the Data Base Management System.

DBMS's from HUG

885-1107 HDOS DBMS's in BHBasic. This disk contains two database management systems. The first is LOGBOOK, an amateur radio operator's log of contacts that are made via radio waves. The second is Transaction Data Management System (TMS), which allows the user to define his own "database". This is the program that won the HUG software contest last November. Neither of these programs can be considered "inferior" because they are written in Benton Harbor Basic! They are both very efficient DBMS's.

LOGBOOK: An Amateur Radio Logbook System, version 1.1

Author: Mr. Jerry Volpe
3620 Lakeshore Dr. Apt F-10
St. Joseph, MI 49085.

Requirements: The program is designed to operate either with the H8/H19/H17 or the H89. It will operate satisfactorily on a single drive system. A minimum of 48k is necessary to run LOGBOOK.

Introduction: LOGBOOK generates a versatile log management system for amateur radio station operations. Its design offers many features not found in similar programs, including a "literal" search function. The program is self prompting and includes a CNTRL "B" interrupt function.

Program Content: There are seven user selectable routines built into the program's menu. Six of these routines are presently defined, a seventh routine has been included as a user's programming option. Space has been provided, including a blank entry in the menu routine.

The main menu has the following commands:

SEARCH
NEW ENTRIES
LIST
HELP
EXIT
OPENING A NEW LOG BOOK

HELP: This command prints the documentation file which explains how to use LOGBOOK.

OPENING A NEW LOG BOOK: This routine opens a data file by <filename>. This must be done before any READ/WRITE operation. It is not normally necessary to go directly to this routine, as the program is designed to make the necessary prompts before a R/W function. Once you have opened a logbook by its <filename>, the program automatically returns to that <filename> for most R/W functions. However, you may return to the menu and select the "OPEN NEW LOG BOOK" subroutine at any time (by entering a CNTRL "B").

NEW ENTRIES: This routine automatically prompts you for the following options or categories:

1) Date and Zulu Time of contact
2) Band and/or frequency of contact
3) Mode of operation
4) Your input power
5) Their call sign
6) Their name
7) Their location
8) Any general remarks

When making entries it is important to remember to try for consistancy in their structure. This will make "SEARCH" operations easier. With leading and ending quotes, your strings may be up to 255 characters in length. However, the ultimate number of possible entries into your logbook, as well as the program's overall speed, improves with short, concise entries.

LIST: This program allows you to list the contents of any logbook on either your terminal or your printer. You have your choice of continuous scan (scroll), page display, or line by line operating modes.

Unless already opened in a previous subroutine, LIST will ask you to open a logbook before attempting any operations. This routine is very handy for making "hard copies" of your logbook, as well as those special files created by SEARCH.

SEARCH: This is the most powerful subroutine of LOGBOOK. With this routine you can literally search for any specified "string"; e.g. names, dates, callsigns, locations, etc. Once you select the search category and the item to be located, the program automatically finds the appropriate log entries. The string must be a perfect match to return the log entry.

The program is specially designed to give you the opportunity of "searching" one or more data files (called logbooks) for the same string. The results of the search can then be listed to the terminal, a printer, or stored as another file. You are able to search any of the categories as listed in NEW ENTRIES above.

Searching can prove a very valuable operating tool. Not only can you quickly identify if you have contacted a station previously, but you can also create condensed logs with some similar characteristics (i.e. all CW contacts).

EXIT: This simple routine insures that all files are closed, then returns to the operating system.

Special features:
1) A logbook character counter that automatically increments on each additional entry.
2) A logbook sector count on the 25th line.
3) A Full Log warning indicator at 48 sectors. This is for 5.25 inch drives and would need to be altered for 8 inch drives.
4) Mode indicator on the 25th line.

Comments: LOGBOOK was written for amateur radio operators but is not limited to that purpose. Any similiar "log" or use can easily be adapted to this program. There would need to be minor changes done to PRINT statements to change the category titles.

LOGBOOK could be easily modified to run under Microsoft BASIC. The CNTRL B function needs a character and carriage return input immediately after the CNTRL B to exit to the main menu.

It should be noted that LOGBOOK is completely contained in <u>one</u> program!! This program is so very efficient, that you may wonder why other programs have to consist of several programs! The program is excellent for beginners to modify. There are features that could be added to make it run more smoothly e.g. single keystroke to issue a command from the main menu, but these would be "appearance" modifications. The file handling of LOGBOOK executes efficiently and needs no modification.

<u>Transaction Data Management System (TMS):</u> Release 2.3

Author: Mr. Larry Towner
218 St. David Drive
Mt. Laurel, NJ 08054

Requirements: Transaction Data Management System (TMS) will run on either the H8/H17 or the H89. It will require a minumum of 24k but 32k or higher is recommended. The system is written for a two drive system. It could be modified for one drive but the file storage is quite limited.

Introduction: Transaction Data Management System is a generalized facility developed to provide the user with the capability to store information in an easily controlled form. It permits the user to store data files onto disk storage with these basic options:

1) Create new data files.
2) Add, modify, and delete records from an existing file.
3) Create duplicate database files sorted in another order.
4) Produce standard reports from data files.
5) Provide a report generator facility which helps you create reports from your data.

The data files are made up of any number of records. Each record is comprised of one to twenty-four fields which the user selects and names.

Program Content: For each database that is created the following may be specified:

1) One to twenty-four user defined data fields. Each data field may contain a variable number of characters.
2) A user selective and descriptive title for each data field.
3) One of the data fields is selected as the key field for the database. The records in the database will be stored in ascending sequence on this field.
4) A predefined report which produces mailing labels.
5) A predefined report which produces a complete listing (dump) of any database.

The following is the Menu of Valid Commands:

ADDRCDS	ADD NEW RECORDS TO FILE
DELETE	DELETE RECORDS FROM FILE
DONE	TERMINATE TMS PROCESSING
GENRPT	GENERATE NEW REPORTS
MENU	PRINT THIS LIST OF COMMANDS
MODIFY	MODIFY RECORDS IN FILE
NEWFILE	CREATE A NEW DATABASE FILE
REPORTS	PRODUCE TMS REPORTS
RPTADD	ADD A NEW REPORT TO SYSTEM
RPTLIST	LIST ALL REPORTS IN SYSTEM
SEARCH	LOCATE SPECIFIC RECORDS
SORT	SORT DATABASE INTO A NEW SEQUENCE

NEWFILE: This command tells TMS that you wish to define a new database. You have the option of selecting one to twenty-four data fields. The titles for each field are entered. Then one of the fields is user selected to be the "key" field of the file. All records stored will be sequenced in ascending order on this "key" field.

A newly created database will consist of three files on the data disk. The first is the database control file. The second is the beginning of your database. The third file is the report control file. It contains the list of the reports. The file is initialized with the label report and the "dump" report but may be updated as explained later.

ADDRCDS: This command instructs TMS to add more records to an existing database. The name of the database is entered, and TMS returns with the current number of fields as prompts for adding the new data (1 to 24 as selected in NEWFILE). If erroneous information was entered, it is possible to edit the data fields before going on. Also available is the option of saving the old data files or replacing them with the new additions.

MODIFY: This command provides the user with the ability to alter the contents of specific records within a database. It can modify one or more database records in the same update session. TMS will display the entire contents of the data record, allowing the user to review the record before and after any modifications. The user may save or delete the old files at completion.

DELETE: The DELETE command permits the user to remove an entire record from the database. The record is found by entering the description of the record in the "key" field you wish to delete.

GENRPT: The report generator option permits the user to easily create reports without writing a BASIC program. The features and limitations of GENRPT are:

1) The length of the report line may be varied from 60 to 132 characters.
2) The number of print lines on a page may also be varied.
3) Each record within the database selected for printing is printed on a single line in a columnar format.
4) The fields printed may not total more than the length of the print line.
5) You may select which fields are to be printed and their order on the print line.
6) The report has three title lines supplied by the user.
7) Page numbering is automatic.
8) A top and bottom page margin is automatically provided. Page skipping is automatic.
9) Database records may be automatically sorted into the order required by the report as part of the report generation process.

REPORTS: The REPORTS command instructs TMS to execute one or more report programs. All reports may be produced using either the "OLD" or "NEW" versions of your database. You may choose from the initialized reports or from those you have created with GENRPT.

SEARCH: This command permits the user to scan the current (old) version of the database for specific records which contain certain information. SEARCH allows the user to perform a scan on a specific field and TMS will return any data which satisfies the scan criteria. e.g. searching for the name "JOHN" will return "JOHNS", "JOHNSTONE", and "JOHNSTON".

SORT: The SORT function instructs TMS to create a special version of the database ordered in a different sequence from the normal database. The sort is done on any of the data fields (1 to 24) and is done in ascending order.

TMSRADD: This command instructs TMS to add a new report to the report control file. This option is used only for reports which are written in BASIC. Reports created by GENRPT are automatically added to the report control file.

RPTLIST: This option provides a list of all reports available for a specific database.

Restrictions (as submitted by the author): TMS has a few restrictions which must be observed:

1) A single data record (the combination of 1 to 24 data fields) may not contain more than 256 characters. The shorter the records the more possible records to store on one disk.

2) The file must not exceed 180 disk sectors (two drive system). TMS creates a new version of the database each time it performs an update and there must be room for that version on the data disk.

3) A maximum of 24 new records may be added to the database at a time. The new records may be added in any order. TMS will sort and store the data.

4) Only one data field may be defined as the "key" upon which the database is ordered.

5) All data fields are actually alphanumeric mode. This means that, while you may store numeric information in a data field, all data fields are treated by TMS as alphabetic.

6) The database may contain only one data record definition.

Comments: The greatest advantage to TMS is the excellent documentation that will come with the package. The author has written 44 pages of step by step instructions to aid the user in understanding the database system.

There are a number of programs that make up the entire package of TMS. The programs may be modified for use with MBASIC but work extremely well in BHBASIC. The program structure is too complicated for the beginner to make any major modifications to any of the programs.

The user will need to follow through step by step with the documentation in order to obtain the proper results of the system. This package allows the user to create several databases. The input format and output structure for each database is the same which make it limited in implementing it to an unusual format. However, the user can use his imagination in modifing TMS to fit his needs.

An excellent general purpose data management system.

885-1108 HDOS DBMS's in MBASIC. This disk contains two database management systems written in Microsoft BASIC version 4.82. The first is INFORMATION SYSTEMS, which allows the user to define his own database. This database is similiar to the TMS system explained above. The second is a TELEPHONE AND MAIL SYSTEM. This is much more than just a mailing label program.

<u>INFORMATION SYSTEMS:</u> A General Information Database System

Author: Mr. Maynard Mansfield
2605 Glencairn Dr.
Fort Wayne, IN 46815

Requirements: The program is written to run on the H8/H19/H17 or the H89 with Microsoft BASIC under HDOS 2.0. The system requires 48k of RAM and a line printer. A two drive system is recommended, however, it may be modified for a single drive.

Introduction: INFORMATION SYSTEM is a highly interactive, general purpose information storage and retrieval system. The user creates any number of databases. This system is broken down from the database "file" to any number of "records". The records consist of one to twenty data "fields" or "field elements". The user may select any title and length for each field element of a record in the database. The length of the field will determine how many records can be stored in the database on a single disk.

This DBMS allows the user to select the number of fields, the title of each field, the length (or number of characters) of each field element, and the type of each field element, i.e. alpha, integer, single precision, or double precision.

Program Content: Each database created by the system is contained in two files: an organizational file which is a sequential file containing the parameter values which define the database, and a random-access file containing the actual data. The parameter values defining a database include the name of the database, the number of data fields in each record, the name of each data field, the type of each data field element and the number of characters (bytes) in each data field. Also included in the organizational file are the date of creation of the database, the date of the last update, the total length of each record, and a pointer to the next available random-access file sector. Clearly the total number of records in a given database is limited to the number of records available on a random file and the record length is limited to the number of bytes in a disk sector.

INFORMATION SYSTEM leads you through the operations by a series of menus. The menus are explicit and follow a natural flow. The following are the list of major menus:

```
Start Menu
     (1)  START a new database
     (2)  work with an EXISTING DATABASE
     (3)  QUIT

Working Menu
     (1)  SEE or EDIT the database
     (2)  QUERY and generate REPORTS
     (3)  SORT the database
     (4)  EXIT

Editing Menu
     (1)  PRINT the entire database
     (2)  SEE or EDIT an existing RECORD
```

(3) ADD a new RECORD
(4) DELETE a RECORD
(5) EXIT

Query Menu
(1) PRINT the records satisfying the query
(2) DELETE the records satisfying the query
(3) EXIT

The add, change, and delete options of the DBMS are similiar to any general data management program. So they will not be discussed here. However, it is important to discuss the unique feature of this DBM system.

The most interesting program in the system is the database "querying" program. "Query" is defined as "to put as a question", (Webster). INFORMATION SYSTEM identifies records satisfying a user-generated query (or argument) and either prints them out or deletes them, as specified by the user. The query language is a variant of the PL/I structure declaration. Its syntax is the same as that described by Elmore and Agarwal in "An Information-Retrieval System," BYTE, October 1980, p.114.

A query may be either simple or compound. A simple query is of the form "NAME OP VALUE" where NAME is the name of one of the data fields, OP is one of the relational operators =,<, or >, and VALUE is a value appropriate to the data field element designated by NAME.

For example, a query of "AGE > 17" will return all records that have in the data field element "AGE" a value greater than 17. A query of "CAR COLOR > C" in a list of cars that has in its field element the color of the car, will return GREEN, PURPLE, PINK, RED, YELLOW but not APRICOT, BLACK, BLUE or CREAM. In other words, ASCII characters are treated as their number coded value i.e. 0 < 9 < A < Z < a < z. Be careful when using non-alpha characters as they have "mixed" values.

Compound queries consist of simple queries (as above) linked by one or more occurrences of the logical operators AND or OR. For example, a query of "AGE > 20 AND COLOR EYES = BLUE" will return all records that have an age greater than 20 with blue eyes.

The query menu gives the option of printing any query directly or creating another format for the printout. A subset of the data field elements may be specified for inclusion in the report, and the order in which the data field elements appear across the page may also be specified. The records to be included in the report may be sorted on any data field element so as to produce either a smallest-first or biggest-first listing.

An entire database may be sorted on any one of its data fields. The sort utilizes a highly efficient implementation of the heapsort algorithm.

The query enables INFORMATION SYSTEMS to print a number of different type of reports. The report structure will vary as much as the number of arguments that are available in using the query option.

Comments: This DBMS is a very useful package primarily due to the query option. The menus and instructions of the program make it easy to follow. There are many safety checks to allow the user to go back to a menu and start over.

The documentation for this package is not as explicit as some of the others but the menus aid the user along through the programs. In essence, it is a well self-documented DBMS by the menus.

The greatest asset to the entire package is the "query" option. The query, if used effectively, will be extremely helpful in creating and using a database management system that depends on searching for "ascending" or "descending" data or for multiple searches of AND's or OR's. The user can use his imagination to use the query efficiently to create and use the INFORMATION SYSTEM database.

TELEPHONE and MAIL SYSTEM: A Telephone and Mail Database System

Submitted by: Mr. Richard Drumb
33214 Shelly Lynne
Sterling Hgts., MI 48077

The programs of Telephone and Mail DBMS, are modifications to the programs which appeared in issue 10 of REMark by William N. Campbell, M.D. The programs were modified to the existing database system.

Requirements: The series of programs are written to run on the H8/H19/H17 or H89 and a line printer is necessary. Microsoft BASIC version 4.82 under HDOS 2.0 is used and requires two drives. For any size database the two drive system is a must. The system, as released, will need a minimum of 56k to operate all the programs. (The user will need to modify the package if less memory is required for his system.)

Introduction: The TELEPHONE and MAIL DBMS generates a customer mailing list, telephone book, mailing labels, form letter printout and form letter generation. This DBMS is much more than a mailing label list package.

Program Content: TELEPHONE and MAIL DBMS allows the user to create any number of "files". Each "record" contains a number of "fields", which consist of a name, address, telephone number, a comment or additional information and a customer code. The customer code field allows the user to establish a customer status base. This gives the option of printing labels or form letters to a specific code.

The mail list section allows the user to create and print any text to a form letter which may be mailed to any customer code (above) or all of the addresses in the mailing file. The format of the form letter is printed so as to be folded to slide into a windowed envelope. The mailing labels may be categorized (customer coded) and printed accordingly.

The telephone section allows the user to print a file of any customer code or all telephone numbers which include the name and address of the respective number. The format is printed alphabetically.

The following is a list of the options available on the main menu:

- #1 CREATE and ADD to a new file
- #2 ALTER and CORRECT existing file
- #3 PRINT Telephone Book
- #4 PRINT Mailing Labels
- #5 DELETE Records from file
- #6 PRINT "key" number list
- #7 SORT file Alphabetically
- #8 PRINT Form Letter
- #9 GENERATE and CORRECT a Form Letter
- #10 EXIT to HDOS

The "key" number of option #6, is simply what chronological order the record is within a particular file. This gives the option of printing just one label or form letter, within a customer code.

The DBMS was set up to create a separate file for each letter of the alphabet. This allows the operator to work with each file (or letter) separately instead of one large file containing all records. Any file name can be used, however the A-Z "filename" seems very efficient. This also creates an easy way of remembering the names of the files.

Comments: TELEPHONE SYSTEMS does not consist of a search program to find a name and then print same. This would have required the exact spelling of the name, which is difficult with the different spelling of names, e.g. JOHNSON, JOHNSTON or JOHNSTONE. Instead, by viewing a file (letter of the alphabet) for the last name via the terminal, at 9600 baud, it requires just a few seconds to find the correct name.

This package is limited in the appearance and usability by the user. The programs do execute properly but could use modifications to fit the users needs.

The only documentation about the package is this abstract created for this article. The menu and subsequent subcommand options allow the user to "feel" and learn his way through the programs.

The TELEPHONE and MAIL DBMS may not be the most efficient of the databases presented here but it has good potential of being a good program for maintaining a large mail and telephone directory. It therefore, is included with INFORMATION SYSTEMS on the MBASIC DBMS disk 885-1108.

The following two DBMS's, RETRIEVER and AUTOFILE are in themselves a separate package. RETRIEVER is contained on a two disk set, while AUTOFILE is released on a single disk. Both DBMS's have been released with their source codes.

<u>885-1109 DBMS; RETRIEVER:</u> A "Keyword Oriented Database", version 1.5

Author: Mr. Peter van den Hamer
Hendrinaland 138
2591TM The Hague
HOLLAND

Requirements: These series of programs require only one 5" disk drive. The programs will run on HDOS 1.5 or later. Benton Harbor BASIC is needed to run some of the programs of RETRIEVER. The programs consist of Assembly Language files, BHBasic files and two .ABS programs which have been compiled with the TINY PASCAL compiler. (The TINY PASCAL compiler is not required in order to run RETRIEVER. It would be needed if modifications were required to the two programs. The author does NOT anticipate any reason to do so.)

The programs require 48K of RAM, however it is possible to adapt the machine language programs to run using less memory (recommended minimum: 24K). An H-19 or H89 terminal is required. Three of the machine language programs require a Z-80 microprocessor, however, by setting a software "switch" at the beginning of their source files and reassembling them, they may be run on an 8080 based system.

Note: The following part of the abstract is taken directly from the authors submitted article. Only slight editing needed to be done to maintain a consistant format and a reasonable length.

Introduction: RETRIEVER can best be described as "a keyword oriented database". It consists of a number of programs to create, expand and access a specific type of database. The following is a simple application:

Suppose you have a few volumes of a couple of different computer-related magazines lying about. That means you have, say, 300 unrelated articles spread out over a few dozen issues. How do you find a specific article about A/D conversion that you're sure you have (somewhere)? Or how do you find out what you have on the language Pascal? How about the professional who has 2000 articles on his particular field of specialization and for whom it is vital that he can quickly find all the information that he has on a subject?

A few mechanical systems are commercially available to return the articles, but they are cumbersome for large systems, too expensive for small applications and they can't give you a hard copy. RETRIEVER was written specifically to handle the finding of items according to their keywords.

Program Content: Assume you want to enter the following article into the system: "An Exchange Evaluator for Computer Chess", by Dan and Kathe Spracklen, from BYTE, November 1978, page 16-28.

The system requires that each article has a number. This can be any integer between 1 and 65534. You enter the keyword information with the program, INPUT.ABS (written in Tiny Pascal and assembler) which looks the desired keywords up in a list (SUBJECTO.DAT) and stores the results in an intermediate file. Note that you must choose keywords that are in SUBJECTO.DAT (up to 254) because the keyword information is stored in a compact form. You might enter the following keywords:

```
     Article no. <finished>: 19
     Subjects:
       (1)  - chess
       (2)  - assembly language
       (3)  - z80
       (4)  - games
```

INPUT features "keyword completion" (similiar to "command completion" in EDIT): you only have to type enough letters that the program knows which keyword you mean. Typing the first two letters of "assembly language" would be enough (assuming that there are no other words present starting with "as-") and at that point the program will complete the rest for you. Unlike EDIT, the cursor position is left unaltered and you can type more if you like (you can still DELETE in the normal way). Another feature I'm rather proud of is the facility to switch back and forth between the normal display and a list of all the "legal" keywords.

The information stored in the intermediate file is transferred to MAIN000.DAT by TRANSFER.ABS. This sorts your input and merges it into the file MAIN000.DAT. This may take 10 seconds. The four entries above would take 2 bytes of disk space each.

The descriptions (optional) are entered with a separate program, ADDTEXT.ABS:

```
     Number: 19
     Title: Exchange evaluator for computer chess
                Pt 2 or 2
     Author: Spracklen
     Journal: BYTE
     Vol.: 11
     Year: 1978
     Page: 16
```

Note that although the prompts suggest the "formal" representation of an article (volume + page) I used the customary one for magazines (year + issue + page). It is easy to adapt the prompts if they are incorrect for your main application. ADDTEXT checks all input as far as possible, compresses it and stores it in a sequential file TEXT000.DAT.

The above information togther with a few "housekeeping" bytes would be stored in 60 bytes of disk space. TEST000.DAT (as well as MAIN000.DAT) is organized sequentially to reduce disk storage space while allowing for a wide range of string lengths.

The title, author or journal field may contain any printing character.

Accessing the files MAIN000.DAT and TEXT000.DAT is done with SEARCH.ABS. This program makes extensive use of the features of the H19 terminal (H89). The screen is initially filled with a "menu" of all the legal keywords. You can then select a particular keyword by moving the cursor there (keypad cursor motion keys) and pressing the f1-key. This, and other top row keys are labeled on the 25th line of the display:

```
     <set>
     <search>
     <text>
     <clear>
     <display>
     <help>
     <bye>
```

A few more commands are accessed by hitting a single letter key (<c> to check the syntax of MAIN000 before making a backup copy of it).

Using <set>, one or more of the keywords can be "switched on" (the keyword is displayed in reverse video). <search> gets you a list of the numbers of all the articles that satisfy the selected keyword combination, e.g. looking up "games" may give you 30 articles; "games" AND "basic" possibly 5. <text> supplies the descriptions of all the articles found with <search>. Both of these functions typically take seconds. A copy of the output to the screen can be sent to a printer.

The complete system (2 disks) also contains a few utilities, a set of sample data files, documentation (150 sectors) and the source listings of all machine language programs. By using a kind of "stand alone" technique, several datafiles of up to 350 sectors each can be handled on a system with one minifloppy drive.

The source listings of the three large assembly language programs start with, what I call, UMOD's (User MODifiers). These are lines in which a non-programmer can select the desired options for his system and particular application: whether you have a Z80 or 8080 CPU, how much memory you have, on which drive you want each datafile, etc. A typical UMOD might be:

```
Z80  EQU  1    CPU type: 1=Z80, 0=8080
```

The UMOD called FORMAT$ ("format string") is somewhat more complicated (there's an appendix on it in the documentation): it is a string of one letter commands that tells the program what to input (output) at what point and what kind of prompts (if any) to use. The FORMAT$ fragment " Title: TR", when used in SEARCH.ASM, would cause "Title: " to be typed on the screen, followed by the content of the title field (T) for that article and by a carriage return (R). Because the FORMAT$'s are constantly scanned while the program is running ("interpreted at run time", if you like), it enables you to program the program. This is especially important if your application has nothing to do with retrieval of articles or reprints. After all, you might want to use

RETRIEVER to keep track of your mineral or record collection.

Summary: RETRIEVER is a keyword oriented database; it can handle the question "which items satisfy a given combination of keywords". The keywords must be chosen from a predetermined (though expandable) list (up to 254). Datafiles are reasonably compact. Because most of the system is in assembly language, response times are quite short. Maximum datafile size is 45,000 entries for MAIN000 and over 1000 article descriptions for TEXT000 per 100 Kbyte disk.

Peter van den Hamer

Comments: RETRIEVER is by far the most complicated and complex database management system that is offered by HUG. It is the most difficult to learn how to use, simply because it is not a "turn-key", menu driven system, meaning the programs are not linked together. The user must load BASIC and all executable programs from the operating system as needed.

The documentation is extensive and detailed for the user to learn the system. As mentioned by the author, the source files are included with the package. The two disk set also contains example data files and allows the user to "play" with the system before creating his own database. This is extremely helpful in understanding RETRIEVER.

This DBMS is not a general purpose database system and was not intended to be one. RETRIEVER will serve its purpose, very well, as a keyword oriented database.

885-1110 DBMS; AUTOFILE: A Filing and Retrieval System

Author: Mr. Jim Tysinger
118 Shannon Hts. Dr.
Verona, PA 15147

Requirements: AUTOFILE makes extensive use of the Z-80 instructions set, and therefore will run only on H89's or H8 Z-80 based systems. It is also dependent on the H89 screen format and function keys. One floppy disk is adequate for program operation, however, an additional drive will allow the use of larger data bases. A minimum of 48K of memory is recommended.

NOTE: AUTOFILE will run on Z-80 microprocessor machines, ONLY!! This means H89's, unless the H8 has the Z80 modification. The source files are provided, and if modified, the programs must be assembled using the Z-80 assembler available from HUG.

Introduction: AUTOFILE, similar in capability but not to be compared with text retrieval systems on large-scale data processing equipment, will be for home computer enthusiasts a useful utility in organizing, filing and retrieving information on a suitable scale for personal use. The DBMS is menu-driven to provide for ease of operation to the user.

Program Content: Design objectives for AUTOFILE include the implementation of useful and efficient functions for the storage and retrieval of items of data while maintaining an uncomplicated interface to the user. It contains keyword, subject, and user-defined category search capabilities of stored data.

An "item of data" in an AUTOFILE data base consists of a subject, a date, a user-defined "category", the keywords associated with the item, and an abstract, which is the main body of text material contained in the item. Although data base searches can be performed on the subject and the user-defined category, the type of search which the program is designed around, is the keyword search. Any combination of keywords and truncated keywords using the logical operators AND and OR can be defined as the data base search argument. The logical NOT of a keyword or truncated keyword can also be used in these combinations.

The keyword search is performed using a core-resident keyword inverted file which results in a search not requiring accesses to disk. The keyword inverted file is similar to a cross-reference list which associates each keyword with the numbers of all items containing that keyword. Most keyword searches appear to the user to be instantaneous. A before/after date limit may also be applied to the search.

After any search, the number of "hits" is displayed, and the user may call for a display of the "hit list" or of any "hit item". The hit list or any hit item may also be printed as a hardcopy.

The chores of maintaining the index file and the keyword inverted file are performed automatically as a part of the functions for new item entry and item deletion. A utility function is also provided for data base re-organization after item deletions.

The very flexible screen format control provided by the H89 combined with the function keys, make the implementation of the menu-driven mode of operation easy and practical. In most cases, the user selects a function from the options displayed on the screen by pressing the appropriate function key.

An input function requiring the keying of characters to some degree is the entry of new items into the data base. The subject, date and user-defined category are saved from the previously entered item when multiple items are entered during one "store items" session. These will be the default field titles for the next group of items, therefore, the grouping of the items by one or more of these fields will reduce the keying effort.

The function keys of the H89 keyboard are used where ever practical as the "turnkey" input from the main and sub-function menus. The main menu of AUTOFILE is shown as follows:

Key	Function
f1	Search items by SUBJECT
f2	Search items by KEYWORDS
f3	Search items by "user-defined item"
f4	Store items
f5	Delete items
Blue	Data Base Organization
Gray	Return to HDOS

"Search items by SUBJECT" ("f1" key) is used to select the character string to be matched against the SUBJECT field of items in the primary item file. This field is limited to a maximum of 40 characters. An exact match must occur to get a "hit" on an item. The primary item file is read sequentially from beginning to end. When "f1" is selected, a sub-menu is displayed to show the options within the search by subject.

When the "f2" key is selected, up to 10 lines are available in which to define the keyword search argument. Up to 8 keywords in the same line are combined in a logical AND function, while the separate lines provide the logical OR capability. The logical NOT of a keyword may be specified by typing a reverse slash immediately prior to the first character of the keyword. The following are examples of keyword arguments:

Argument	Result
RED YELLOW BLUE	red AND yellow AND blue
RED YELLOW BLUE	red OR yellow OR blue
RED \BLUE	red NOT blue

A sub-menu of the available options is displayed on the H89, also, when the "f2" key is selected.

Search by "user-defined item" ("f3") will need a brief explanation. The user-definable item or category field is a useful addition to the subject and date of an item. It is called the "display field" because it is one of the fields displayed when the hit list is called for. A couple of examples of specific display field definitions could be: LOCATION, if items in the data base refer to magazine articles or books, etc., or ORIGINATOR, if items in the data base refer to pieces of correspondence. The display field is defined when a new data base is created, and the field definition supplied at that time remains associated with that particular data base. If no special category field is desired, it can be rejected at the time the data base is created.

In any of the above searches, a before/after date limit may be applied. The sub-menu function (option "f2" of the sub-menu) is used to implement the date limit prompt. When prompted, the user enters a date, after which the before or after option can be selected. Only "hits" before or after the date supplied are allowed to remain hits.

Option "f4", "Store items", does the adding and storing of new items to a data base. For each item to be stored, the user is prompted for a subject, a date, a user-specified data category, keywords, and an abstract. After each item is entered, the user will be asked if another entry is desired. If "Y" is specified, the user is prompted for the information needed for the next item to be stored. The default subject, date, and category fields are shown as the previous entry for each field, and are selected by a carriage return. The subject field is limited to 40 characters, while the category display field is limited to 25. Multiple lines may be entered when specifying keywords or creating the abstract. There is no restriction on the number of keywords or on the amount of text in the abstract.

Upon selection of the "f5", "Delete items", the user is prompted to supply item numbers one at a time. Items are deleted as their numbers are entered, until the user indicates that there are no more items to be deleted. To recover the deleted space, the re-organization function is required.

The "Data Base Re-organization" function (Blue Key) creates a new data base containing the valid items of the old data base, leaving out the space used by deleted items. The user is first prompted for a device number on which to build the new data base. Messages are displayed during the copy process to mark the progress of the action. At completion, AUTOFILE will exit, allowing the user to copy or rename the new data base.

Restrictions: Each data base is limited to a maximum of 254 items, although a user may have as many data bases as he has floppy disks to contain them. The size of any item is controlled by the person entering the item, and there is no program restriction on the number of keywords or on the amount of text in the abstract. 254 items of moderate size (4 or 5 full lines of abstract text) will come close to completely filling a normal formatted five-inch disk.

Comments: AUTOFILE is a very unique DBMS. It has a keyword, subject, and category search. AUTOFILE is unique in that it stores any text abstract of any length for retrieval.

AUTOFILE is virtually a complete "turnkey" operation. The commands from the menus and links between programs are all controlled, independent of the user, by single keystroke entry.

To explain again briefly: Each "item" of AUTOFILE contains an item number (1 to 254), a date (mm/dd/yy), a 40 character subject, a 25 character category, any number of keywords, and lastly any length abstract. The only "report" available with AUTOFILE is the formatted output structure of these six fields.

AUTOFILE is quite different from a general database management system and may have the options that will meet a unique application that requires, as part of the data, a lengthy text abstract.

885-1214 CP/M DBMS; LOGBOOK. The last disk that is available at this time is the CP/M version of LOGBOOK. It is written in Microsoft BASIC version 5.2. The structure and options are virtually identical to the BHBasic LOGBOOK as explained above. Therefore, a detailed description of this version is not required. Please refer to the abstract above for any questions concerning LOGBOOK.

DBMS's in Conclusion

The Data Base Management Systems that are presented here are obviously not the only DBMS's that are available to you. There are a number of CP/M DBMS's that are available from different sources. Heath and Softstuff have a couple of systems available. In addition, there are independent software houses that have created a number of DBMS's for different computers and operating systems.

Data Base Management Systems are available from different sources. They may range in price from $19 to $1000. There are too many to list in this article, but you may come across them in your reading of computer magazines. Some sources of database systems that were written specifically for Heath computers are:

1) Heath has the Condor Data Base Management System which operates under CP/M and sells for $899. I have only had a very brief look at the documentation of Condor and realize after glancing through it that it is a very extensive system. There are many options available to the user that are not mentioned in this article.

2) Softstuff has the Mailpro 127.1 Mailing List Package which is a type of database management system, for $40, which runs under HDOS. I have not had an opportunity to see or review any documentation on Mailpro, but can offer it as another alternative.

3) Excalibur Systems Ltd. has a general database system, DATABASE, which sells for $79. It also has 3 support packages which sell for $39 per package. For further information, contact Greg Greene at

Excalibur Systems Ltd.
207-885 Craigflower Rd.
Victoria, B.C. CANADA V91 2X4

4) Keyboard Studios has a few database management systems available. These are low priced systems and vary from dedicated to general database systems. For further information write to

The Keyboard Studio
125 Aspen
Birmingham, MI 48009

With these alternative database management systems available to you, you should have no difficulty in finding, evaluating, and deciding which database management system is the right package for your needs. You may choose from the HUG products or from elsewhere.

Take your time in determining your needs. Review several different DBMS's by examining the options as outlined earlier. Weigh the pros and cons, and choose the package that will meet your needs the best. It is important that you take your time; you just can't run down to your friendly neighborhood dime store and pick out another database management system.

None of the Data Base Management Systems mentioned in this article are the responsibility of the Heath User' Group. These DBMS's have been offered to you in this article to enable you to learn what database management systems are about and what packages are available.

It is hoped, this article will be used as reference material for information and suggestions as to relative questions of DBMS's and what to look for in same. There are many different versions and implementations of DBMS's. By studying this article and its abstracts, you should have learned the basics of a Data Base Management System.

<TLJ>

New HUG Software

DATA BASE MANAGEMENT SYSTEMS (DBMS):

For an explanation of the DBMS's refer to the indicated page of this issue.

885-1107 HDOS DBMS in BH BASIC
    Logbook:..........................Page 6
    TMS:..............................Page 7
885-1108 HDOS DBMS in MBASIC
    Information System:..............Page 9
    Telephone and Mail:..............Page 10
885-1109 HDOS DBMS
    Retriever:........................Page 11
885-1110 HDOS DBMS
    Autofile:.........................Page 13
885-1214 CP/M DBMS
    Logbook:..........................Page 15

For the price of these items refer to the HUG Product List on page 17.

885-4001 REMark VOLUME I $20.00

INTRODUCING REMark Volume I ! One of the major difficulties for the newcomer to the HUG community is catching up with the rest of the gang who have a complete collection of the REMark Library. By popular demand, HUG is now offering Issues 1 to 13 of REMark bound in a handsome single book at an introductory price of $20.00. This limited edition will be followed by REMark Volume II which will Include Issues 14 to 23.

885-1053 H11 TSTE MODEM PACKAGE $20.00

This modem package for the H-11 system requires the Heath Serial I/O Board, H-11-5, and will operate on HT-11 or RT-11 Disk Operating Systems with a normal telephone modem and console device. TSTE uses a series of control keys to direct the functions of the modem package. Some options include opening and closing the printer port and the transmitting, receiving and closing of files. One of the control functions sends a "BREAK" to the host. Another control key sequence will allow you to send special characters to the remote. TSTE will need to be reassembled for the HT-11 operating system and the source code is provided.

NEW GAMES FOR CHRISTMAS

HUG is proud to present the following game disks for your holiday enjoyment.

885-1113 HDOS FAST ACTION GAMES $20.00

How fast do you react? These games will help you find out. They will all run on an H89 or H8 with H19, and require 32k of RAM. Source code is included.
BREAK19 -- This is a variation of the "Breakout" TV game in which you try to break through barriers with a ball hit from a paddle. It features user selectable playing speed, bonus points, and weighted scoring.
SKI -- This game simulates a slalom ski race. You try to control your "skier" as he races down a zig-zag course. It has user selectable playing speed and scoring.
SNAKE -- In this game, you control the movements of a snake as he crawls around your screen. You try to reach the "food" without hitting a barrier or looping back on the snake, and the snake keeps growing! With user selectable speed and scoring.
BUGS -- This is the traditional "Life" game in which "cells" react to produce interesting patterns on the screen. This version uses H19 graphics and runs extremely fast. It will not work on an H8 that uses the H8-5 card for the console.

885-1112 HDOS GRAPHIC GAMES $20.00

This disk contains BASIC and machine code games. The BASIC games will run in either B H BASIC or MBASIC without modification. For the H89 or H8/H19.
KENO -- This game simulates Las Vegas style Keno using H19 graphics for the playing card and for drawing balls. It requires 48k if you use B H BASIC or 56k if you use MBASIC.
HORSES -- This is a horse race simulation game that several people can play together. It maintains a disk library of several "horses" from which horses are picked for each race. Payoffs are based on the horses past records, which are recorded at the end of each playing session. The name of the highest winner is also recorded. Simple animation makes the game more interesting. Requires 32k with B H BASIC, or 48k with MBASIC.
SPY -- This is a game of industrial espionage. A rival company has stolen plans from your company, and it is your job to break in and retrieve them. H19 graphics are used to show floor plans, etc. Requires 48k (B H BASIC or MBASIC).
CRAPS -- The traditional dice game -- in graphics. Requires 32k.
FANTAN -- Fan-Tan is a card game for 4 players. In this version, the computer plays 3 of the hands and you play the 4th. Can you beat 3 computer players? Requires 32k for B H BASIC, or 48k for MBASIC.
ACKACK -- This is a machine language action game in which you try to shoot down "airplanes" before they shoot you. Requires 32k, and the source is included.
BRKOUT -- Here is another variation of the "Breakout" TV game. You are scored by the number of "bricks" you knock out. No source is available for this program, which requires 32k.

885-1111 HDOS MBASIC GRAPHIC GAMES $20.00

All of the games on this disk require Microsoft BASIC and 48k of memory, except WORD, which requires 56k. For the H89 or H8/H19.
WORD -- This is a computer version of Parker Brother's "Probe" (TM) word search game. One to 3 people can play against each other and the

computer. The computer has a library of 500 words to choose from. This game is educational as well as fun.
SECTOR -- In this game you try to defend up to 5 sectors of the galaxy against invading aliens. You try to shoot the aliens and aviod hitting the allies.
MASTRMND -- This is an excellent graphics version of the traditional "Mastermind" game, in which you try to guess a coded series of colors.
POKER -- This game simulates the coin operated poker machines found in some casinos.
WINNING -- This is the child's game of "paper scissors rock" in H-19 Graphics.
ACEY -- "Red Dog" or Acey-Deucy in H19 Graphics.
CHECKERS -- If you like to play checkers, now you can do it with a computer.
ONECHECK -- A variation of checkers in which you try to jump as many pieces as possible.

885-1114 -- COLOR RAIDERS AND GOOP $20.00

These games are for the HA-8-3 Color Graphics board for the H8 computer. Requires HDOS, 32k of RAM.
RAIDERS -- In this game, the color monitor is the view from your space ship into space. You are pursuing an enemy space ship. Suddenly a small fireball of energy appears to come from the enemy ship. It grows larger and larger, and finally an explosion rocks your ship. You steer towards him and fire your phasers. Missed! Fire again! Got him! You see his ship break up into pieces, then another enemy ship appears. This game features realistic star field movement, sound, scoring, user selectable speed, and H19 keypad control.
GOOP -- In this game, you shoot at descending space creatures. It can be played from a joystick (contact type), from the H8 keypad, or from the H19 keypad. Your hit count is maintained on the color screen.

HUG BULLETINS:

The CP/M Series on getting acquainted and learning the CP/M operating system will be continued in the February Issue of REMark.

The CAT and SDUMP programs on 885-1213 do not have to be reassembled to run under non-Heath CP/M as stated in REMark Issue 22 as long as the CP/M is Version 2.0 or higher.

HUG Product List

Part Number	Description		Selling Price
CASSETTE SOFTWARE (H8 and H88)			
885-1008	Volume I Documentation and Program Listings (some for H11)		$ 9.00
885-1009	Tape I Cassette		$ 7.00
885-1012	Tape II BASIC Cassette		$ 9.00
885-1013	Volume II Documentation and Program Listings		$ 12.00
885-1014	Tape II ASM Cassette H8 Only		$ 9.00
885-1015	Volume III Documentation and Program Listings		$ 12.00
885-1026	Tape III Cassette		$ 9.00
885-1036	Tape IV Cassette		$ 9.00
885-1037	Volume IV Documentation and Program Listings		$ 12.00
885-1039	WISE on Cassette H8 Only		$ 9.00
885-1057	Tape V Cassette		$ 9.00
885-1058	Volume V Documentation and Program Listings		$ 12.00
HDOS SOFTWARE (H8/H17 or H89 -- 5-inch only)			
MISCELLANEOUS COLLECTIONS			
885-1024	Disk I H8/H89		$ 18.00
885-1032	Disk V H8/H89		$ 18.00
885-1044	Disk VI H8/H89		$ 18.00
885-1064	Disk IX H8/H89		$ 18.00
885-1066	Disk X H8/H89		$ 18.00
885-1069	Disk XIII Misc H8/H89		$ 18.00
GAMES			
885-1010	Adventure Disk H8/H89		$ 10.00
885-1029	Disk II Games 1 H8/H89		$ 18.00
885-1030	Disk III Games 2 H8/H89		$ 18.00
885-1031	Disk IV Music H8 Only		$ 23.00
885-1067	Disk XI Graphic Games .ABS and B H BASIC (H19/H89)		$ 18.00
885-1068	Graphic Games (H19/H89)	*	$ 18.00
885-1088	Graphic Games (H19/H89)	*	$ 20.00
885-1093	Dungeons and Dragons Game Requires H89 or H8/H19	*	$ 20.00
885-1096	Action Games (H19/H89)	*	$ 20.00
885-1103	Sea Battle Game (H19/H89)		$ 20.00
885-1111	HDOS MBASIC Graphic Games	*	$ 20.00
885-1112	HDOS Graphic Games		$ 20.00
885-1113	HDOS Fast Action Games		$ 20.00
885-1114	Color Raiders and Goop (HA-8-3)		$ 20.00
UTILITIES			
885-1019	Device Drivers (HDOS 1.6)		$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89		$ 15.00
885-1025	Runoff Disk H8/H89		$ 35.00
885-1043	MODEM Heath to Heath H8/H89		$ 21.00
885-1050	M.C.S. Modem for H8/H89		$ 18.00
885-1060	Disk VII H8/H89 SUBMIT, CLIST, FDUMP, ABSDUMP, etc.		$ 18.00
885-1061	TMI Cassette to Disk H8 only		$ 18.00
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM		$ 25.00
885-1063	Floating Point Disk H8/H89		$ 18.00
885-1065	Fixed Point Package H8/H89		$ 18.00
885-1075	HDOS Support Package H8/H89		$ 60.00
885-1077	TXTCON/BASCON H8/H89		$ 18.00
885-1079	HDOS Page Editor		$ 25.00
885-1080	EDITX H8/H19/H89		$ 20.00
885-1082	Programs for Printers H8/H89		$ 20.00
885-1083	Disk XVI RECOVER, etc.		$ 20.00
885-1089	MACRO, CTOH, and misc Utilities		$ 20.00
885-1090	Misc. HDOS Utilities CCAT, HPLINK, AH, MBSORT, etc.		$ 20.00
885-1092	RDT Debugging Tool H8/H89		$ 30.00
885-1095	HUG SY: Device Driver HDOS 2.0		$ 30.00
885-1098	H8/HA-8-3 Color .ABS/.ASM		$ 20.00
885-1099	H8/HA-8-3 Color in Tiny Pascal		$ 20.00

Let There Be Sound!

One thing has always bothered me about the H-89 computer. When playing games or when trying to create a special program, you are stuck with the blasted terminal horn as a means of alerting you of some malfunction, hit, error, etc. Well, no more! After a weekend of thought, and a couple of games later I disconnected the terminal "bell" as a source of the disturbing "beep-beep". I then went to work on a simple (and cheap) alternative to the situation.

After looking over the schematic for the H-89, it was found that there are two unused bits of the General Purpose Port 362 (Octal). Yup, right there in front of me was the answer I needed to construct a sound generator from existing hardware within the H-89. Data bits D2 and D3 didn't seem to go anywhere, so I had found my port and output for the new "horn". Using the "heads-or-tails" technique, I selected D2 as the output line which, fortunately, is latched or remains in the last given state of high or low. One problem did surface however. It seems that the clock can be disrupted by writing to the GP. So, you must make sure that the clock is restored after writing to the GP by outputting an 002Q before exiting your routines.

CAUTION: DO NOT PERFORM THE FOLLOWING UNLESS YOU ARE COMFORTABLE WORKING WITH ELECTRONIC EQUIPMENT AND A SOLDERING IRON. USE REASONABLE PRECAUTIONS AS YOU CONSTRUCT THIS VERY SIMPLE CIRCUIT.

Before we proceed, you might like to know what is necessary or required to modify the H-89. Simple! You will need (1) 100 ohm resistor, (1) speaker similar to that used as the current "bell" (45 ohm), and (2) suitable wires approximately 24 inches long each. As we go through the connections, you may wish to examine FIGURE 1. CAREFULLY connect one of the two wires to pin 20 of IC U552 on the foil side of the board. Connect the remaining wire to pin 6 of the same IC. Pin 6 is the output and pin 20 is the five volt supply. Once you have completed these connections, reassemble the H-89 using care to route the wires between the Terminal Logics Board and the CPU Board. I found that routing the wires over the top of the boards was easy and would cause less strain. Next, you must make the necessary connections at the speaker. To one lug of the speaker, solder a 100 ohm resistor. Solder the lead currently connected to pin 20 of the IC to the opposite end of the resistor. Connect the remaining wire to the remaining speaker lug to complete the modification.

That's all there is to it! Cheap and easy to do. With the modification completed, we can now proceed with the fun part. The following program is actually several routines that I have compiled quickly to give you an idea of how effective your new "horn" can be. These are only samples of what you can do and the big word here is EXPERIMENTATION! You can pull any of the subroutines along with "NOTES" if you wish to use just one of the sounds presented here. Remember, you must write a 002Q to the GP before exiting (SCALL 0) or you will find the system hanging very well in space!

.... Happy noise making BE:

```
              ****HORN89****
042.200       START   ORG     *
              ****
000.000       OFF     EQU     000Q
000.001       LAZF1   EQU     001Q
000.002       EXIT    EQU     002Q
000.004       ON      EQU     004Q
000.010       LAZREP  EQU     010Q
000.050       WARREP  EQU     050Q
000.075       PHAZFEQ EQU     075Q
000.200       WARFREQ EQU     200Q
000.362       PORT    EQU     362Q
000.377       LOWBY   EQU     377Q
000.377       LAZFREQ EQU     377Q
000.377       PHAZREP EQU     377Q
377.377       TIMWAIT EQU     377377A
```

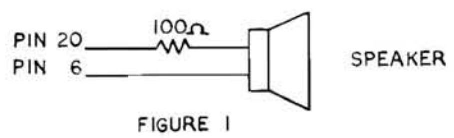


FIGURE 1

```
                       **** BODY
042.200  315 317 042                 CALL    LAZER
                       ****
042.203  315 241 042                 CALL    WAIT
                       ****
042.206  315 277 042                 CALL    WARBLE
                       ****
042.211  315 241 042                 CALL    WAIT
                       ****
042.214  315 370 042                 CALL    REZAL
                       ****
042.217  315 241 042                 CALL    WAIT
                       ****
042.222  315 350 042                 CALL    PHAZER
                       ****
042.225  315 241 042                 CALL    WAIT
                       ****
042.230  315 010 043                 CALL    SCALE
                       ****
042.233  076 002                     MVI     A,EXIT
042.235  323 362                     OUT     PORT
042.237  377 000                     SCALL   0
                       ****
042.241  052 050 043   WAIT          LHLD    TEMPTIM
042.244  353                         XCHG
042.245  033           TLOOP         DCX     D
042.246  173                         MOV     A,E
042.247  262                         ORA     D
042.250  302 245 042                 JNZ     TLOOP
042.253  311                         RET
                       ****
042.254  076 004       NOTES         MVI     A,ON
042.256  323 362                     OUT     PORT
042.260  170                         MOV     A,B
042.261  075           ONLOOP        DCR     A
042.262  302 261 042                 JNZ     ONLOOP
042.265  076 000                     MVI     A,OFF
042.267  323 362                     OUT     PORT
042.271  170                         MOV     A,B
042.272  075           OFFLOOP       DCR     A
042.273  302 272 042                 JNZ     OFFLOOP
042.276  311                         RET
                       ****
042.277  016 050       WARBLE        MVI     C,WARREP
042.301  006 200       WO2           MVI     B,WARFREQ
042.303  315 254 042   WO1           CALL    NOTES
042.306  005                         DCR     B
042.307  302 303 042                 JNZ     WO1
042.312  015                         DCR     C
042.313  302 301 042                 JNZ     WO2
042.316  311                         RET
                       ****
042.317  016 010       LAZER         MVI     C,LAZREP
042.321  006 377       LO2           MVI     B,LAZFREQ
042.323  315 254 042   LO1           CALL    NOTES
042.326  005                         DCR     B
042.327  302 323 042                 JNZ     LO1
042.332  041 377 377                 LXI     H,TIMWAIT
042.335  042 050 043                 SHLD    TEMPTIM
042.340  315 241 042                 CALL    WAIT
042.343  015                         DCR     C
042.344  302 321 042                 JNZ     LO2
042.347  311                         RET
                       ****
042.350  016 377       PHAZER        MVI     C,PHAZREP
042.352  006 075       PO2           MVI     B,PHAZFEQ
042.354  315 254 042   PO1           CALL    NOTES
042.357  005                         DCR     B
042.360  302 354 042                 JNZ     PO1
042.363  015                         DCR     C
042.364  302 352 042                 JNZ     PO2
```

Merry Christmas

```
042.367  311          00088            RET
                      00089   ****
042.370  016 010      00090   REZAL    MVI     C,LAZREP
042.372  006 001      00091   RO2      MVI     B,LAZF1
042.374  315 254 042  00092   RO1      CALL    NOTES
042.377  004          00093            INR     B
043.000  302 374 042  00094            JNZ     RO1
043.003  015          00095            DCR     C
043.004  302 372 042  00096            JNZ     RO2
043.007  311          00097            RET
                      00098   ****
043.010  041 052 043  00099   SCALE    LXI     H,DATA
043.013  106          00100   SCALE2   MOV     B,M
043.014  043          00101            INX     H
043.015  126          00102            MOV     D,M
043.016  043          00103            INX     H
043.017  036 377      00104            MVI     E,LOWBY
043.021  257          00105            XRA     A
043.022  270          00106            CMP     B
043.023  312 047 043  00107            JZ      STOP
043.026  315 254 042  00108   SCALE1   CALL    NOTES
043.031  033          00109            DCX     D
043.032  173          00110            MOV     A,E
043.033  262          00111            ORA     D
043.034  302 026 043  00112            JNZ     SCALE1
043.037  345          00113            PUSH    H
043.040  315 241 042  00114            CALL    WAIT
043.043  341          00115            POP     H
043.044  303 013 043  00116            JMP     SCALE2
043.047  311          00117   STOP     RET
                      00118   ****
043.050               00119   TEMPTIM  DS      2
043.052  010 062 020  00120   DATA     DB      010Q,062Q,020Q,031Q,037Q,015Q
043.060  076 007 173  00121            DB      076Q,007Q,173Q,004Q,377Q,002Q
043.066  000 000      00122            DB      0,0
                      00123   ****
043.070  000          00124            END     START
```

B H BASIC vs MBASIC

"Tell me one good reason why I should spend $150 for another version of BASIC?"

How many times have I heard that question when Microsoft BASIC (MBASIC) is recommended over Benton Harbor BASIC (BH BASIC) for the Heath/Zenith computers.

I'll let you be the judge as to whether the advances of MBASIC make the $150 worth spending.

Just a thought as to whether MBASIC is worth $150; how much was your personal or business income last year? Did you make more than $9,999.99? If you can answer yes to the second question then MBASIC is for you.

Why should I spend $150 if I made $10,000 in one year? If you don't mind loosing pennies at $10,000 and dollars at $100,000 then continue to use BH BASIC. BH BASIC only has six digits of precision where MBASIC has sixteen digits of precision, that is $99,999,999,999,999.99 (using double precision). That is about ninty-nine times the limit of national debt. To most of us that is a little more than just pocket change.

Following are three lists of commands that are gone, changed or new when going to MBASIC from BH BASIC.

MBASIC vs. BH BASIC

Commands gone when going to MBASIC from BH BASIC.

CHAIN	LOCK	SEG (H8 Only)
CIN	MAX	UNLOCK
CONTROL	MIN	UNFREEZE
FREEZE	PAD (H8 Only)	
LNO	PAUSE	

Commands changes.

BH BASIC		MBASIC
BYE	=	SYSTEM
BUILD	=	AUTO
MATCH	=	INSTR
OLD	=	LOAD
PIN	=	INP
REPLACE	=	SAVE
SCRATCH	=	NEW
SPC	=	SPACE$
UNSAVE	=	KILL
USE	=	FRE

NEW MBASIC Commands.

CINT	EQV	MERGE	STRING$
CSNG	ERASE	MKD$	SWAP
CDBL	ERL	MKI$	TROFF
CVD	ERR	MKS$	TRON
CVI	ERROR	MOD	USING
CVS	FIELD	NAME	USR
DEFDBL	FIX	NULL	VARPTR
DEFINT	GET	OCT$	WAIT
DEFSNG	HEX$	PUT	WIDTH
DEFSTR	IMP	RENUM	XOR
DEFUSR	LOC	RESET	
EDIT	LOF	RESUME	
EOF	LSET	RSET	

Take a close look at the commands that are gone and try remembering the last time you used one of these commands in a BASIC program. Yes, you may have used one, two or maybe three of these commands such as CHAIN, CONTROL and PAUSE, but there are ways around all three of these commands.

First the CHAIN command may be replaced by using RUN"FNAME", however the values of ALL the variables will be lost. But, if you need values, send them to a file and retrieve them in the next program. NOTE: All dimensions are also erased, requiring that all arrays be redimensioned in the new program.

The CONTROL command is not needed anymore because ALL of MBASIC is loaded into memory before you start programming.

The PAUSE command had two functions in BH BASIC, the first being the wait for the RETURN key to be hit and the second being the time delay. The wait for the RETURN key may be replaced by using the A$=INPUT$(1) function which allows you to hit any key on the keybroad, making the string A$ equal to the key struck. If the RETURN key is struck the value of A$ is CHR$(13). Checks for different keys may also be done for other functions within the program.

The time delay function of the PAUSE command may be replaced by a FOR-NEXT loop such as: FOR I=1 to 1000:NEXT I

CHANGED COMMANDS

A few commands have been renamed to other commands, however they still function basically the same.

NEW MBASIC COMMANDS

The NEW MBASIC commands add a lot of power to MBASIC that BH BASIC did not have. Some of the new commands are very powerful, those commands are EDIT, ERROR-RESUME, RENUM, RESET, SWAP and TRON-TROFF.

EDIT

The EDIT command has the ability to make your programming efforts much faster and easier. To activate the editing of a line, simply type the word EDIT, the line number of the line to be edited and the RETURN key .

EXAMPLE: EDIT 10

This will place you in the EDIT mode of MBASIC to allow the adding, changing and deleting of characters within the current line.

Several commands are available while in the EDIT mode of MBASIC, they are:

A	Restore to the original line.
C	Change the next character to the following character.
D	Delete the next character.
E	End the editing.
ESC	Exit the insert mode.
H	Kill to the end of line and place in the insert mode.
I	Enter the insert mode.
K	Kill to the next character pressed.
L	List the line (look at).
Q	Quit the EDIT.
S	Search to the next character.
X	Goto end of line and enter the insert mode.

The commands C, D and S may be preceded by a number which allows the changing, deleting or searching a number of characters.

Need to change a line number? Yes, you may list a line and then type control-A which enters the line edit mode allowing the change of the line number. If the line number is changed the old line remains intact and a new line is added to the program.

ERROR-RESUME

The ERROR function is not really a command but rather a way for a program to recover from an error. This is very useful for recovering from an error caused by not finding a file, the end of a file, an illegal input or other MBASIC error.

Example:

```
10 '      SAMPLE OF ERROR RESUME
20 ON ERROR GOTO 2000
30 I=100:DIM A$(I),B$(I)
50 PRINT "Enter file name? <END> ";
60 LINE INPUT FN$
70 IF FN$="" THEN SYSTEM
100 OPEN "I",1,FN$+"DAT"
110 FOR I=1 to 100:IF EOF(1) then 160
120 LINE INPUT #1,A$(I)
130 LINE INPUT #1,B$(I)
140 PRINT A$(I)B$(I)
150 NEXT I
160 CLOSE
170 PRINT "OK, END OF FILE"
180 GOTO 50
200 PRINT "File not found try again."
210 GOTO 50
220 PRINT "BAD File Name!"
230 PRINT "Re-enter it."
240 GOTO 50
2000 IF ERR=53 THEN 2100
2010 IF ERR=63 THEN RESUME 160
2020 IF ERR=65 THEN RESUME 220
2090 PRINT "Error"ERR"in line"ERL:END
2100 IF ERL=100 THEN RESUME 200
```

This program is a simple example of the use of the ERROR-RESUME functions and also the EOF, ERR and ERL where EOF is the end of file, ERR is the error number and ERL is the line where the error occurred.

Other commands shown in this example are the SYSTEM command to return to the operating system and in line 10 the first character is the apostrophe used in place of the REM command. The REM command may still be used.

One other very important addition to MBASIC is shown in line 60 where the variable name is two letters (FN$).

RENUM

The RENUM command is almost self explanatory except it has several options that make it very powerful.

The command RENUM by itself renumbers the entire program with first line starting at line 10 and in increments of 10.

The first option is that you may specify what the first line of the program will be.

Example: RENUM 100

This command renumbers in increments of 10 starting with the first line at 100.

The second option for the RENUM command is to specify the line within the current program where to start the new line.

Example: RENUM 100,20

This will renumber all lines after line 19, starting the first new line at 100 and in increments of 10.

The third option is to specify the size of the increments.

Example: RENUM 2000,100,15

This will renumber all lines above 99 in increments of 15 and starting with the first new line as 2000.

RESET

The RESET command allows the resetting of all the drives while in either command mode or in the body of a program. This command is very useful in using multiple data disks giving a very large amount of data using a limited number of drives.

When using the RESET command the disk that is placed in the drive when requested need <u>ONLY</u> be initialized, not SYSGENed. The fact that the disk is only initialized saves approximately one hundred sectors of space on the disk.

The disk may only contain data if desired.

EXAMPLE: Need to change SY0:.

20 RESET "SY0:"

A good example of the use of the RESET command is the Small Business Package III which may be used with one, two or three drives, but still allows all features. It would be best if all three drives were available but that is not always possible, therefore the RESET command proves very useful.

SWAP

The SWAP command is used to interchange two variables saving time and programming space.

EXAMPLE: Interchange variables A$ and B$.

BH BASIC	10 B$=C$:A$=B$:B$=C$
MBASIC	10 SWAP A$,B$

TRON-TROFF

The last of the new commands to be described here is the TRON-TROFF command. The TRON command stands for trace on and TROFF is for trace off. What is a trace? A trace is the ability to watch the line numbers of the program as they are executed. What this means is that the line numbers of the program will appear on the screen as they are called by the BASIC program. This feature is very useful in trouble shooting a program, especially when a program has a lot of GOTOs or GOSUBs.

SUMMING IT UP

That's a lot of information to try to inhale in only a few paragraphs and it was not intended to make you an MBASIC expert, but to only expose you to some of the main features of MBASIC. For further information on MBASIC, the manual that comes with MBASIC will help some, but is only a reference manual not a course to teach MBASIC. As a strong suggestion, the NEW course offered by Heath Company, called Programming in Microsoft BASIC will offer the needed information to allow you to program in MBASIC. This course was written on the Heath/Zenith computers, however it applies to ALL computers using MBASIC. The course model number is EC-1110 and sells for $99.95.

:GK:

DS Patch for HDOS ASM 104.06.00

by Ted Eitel III
16313 Redington Dr.
Redington Beach, FL 33708

If you are an assembly language programmer, this patch will make it easier for you to fill blocks of memory with a constant at assembly time. For example, suppose you need to fill 10 bytes with zeros. The usual way to do it is with a DB statement as follows:

```
        DB      0,0,0,0,0,0,0,0,0,0
```

After making the following patch, you can do it with a DS statement:

```
        DS      10,0
```

The first number in the operand field is the number of bytes to fill, and the second is the constant. The constant can be any number less than 256 or any expression that evaluates to an 8-bit number. If you leave off the comma and constant, the DS statement works in the old way, and just reserves space. The patch to ASM is listed below. Use PATCH.ABS to make the changes. Note that the old data in the second section may be different, since this is a patch area.

```
ADDRESS   OLD DATA   NEW DATA   MNEMONIC
045172    042        303              JMP  PATCH
045173    176        054
045174    072        073

073054    040        032 PATCH  LDAX D
073055    061        376        CPI  ','
073056    071        054
073057    070        312        JZ   DS2
073060    060        070
073061    012        073
073062    040        042        SHLD ORG
073063    050        176
073064    142        072
073065    171        303        JMP  ASM11
073066    040        367
073067    107        044
073070    101        023 DS2    INX  D
073071    103        325        PUSH D
073072    050        305        PUSH B
073073    151        315        CALL E8B
073074    156        054
073075    040        057
073076    162        171        MOV  A,C
073077    145        315        CALL OBB
073100    155        354
073101    145        060
073102    155        301        POP  B
073103    142        321        POP  D
073104    162        013        DCX  B
073105    141        170        MOV  A,B
073106    156        261        ORA  C
073107    143        302        JNZ  DS2+1
073110    145        071
073111    040        073
073112    157        303        JMP  ASM11
073113    146        367
073114    040        044

                                EOF
```

A Tiny SY.DVD

If you are in need of more space on your HDOS 2.0 system disks, here is yet another method. It was suggested to us by HUG member Ted Andrews. The disk device driver (SY.DVD) is actually two programs joined together into one file: the operating program and the initialization program. If you have a system disk that you never use for initializing other disks, all you really need on it is the operating part. To get it, assemble SYDVD.ASM on your HDOS 2.0 Software disk. Then delete SY.DVD from your system disk (yes, you can delete it even though the L flag is set), then copy the assembled SYDVD to your disk as SY.DVD. This gives you a disk driver only 4 sectors big. If you are using the HUG SY: device driver (885-1095), you can use DKH17.REL as your SY.DVD for a 6-sector driver, and you don't even have to assemble it.

PS:

Pascal Corner Part II

by Henry E. Fale
2918 S. 7th. St.
Sheboygan, WI 53081
(414) 452-4172

Welcome to part two of the Pascal Corner. I hope you gathered your reading material and started to get into this on your own. Like I said last month, reading this column will do very little for you if you do not take the incentive to work along with me. From this point on, the print will not be two column, but single column. The reason for this is mainly so program listings will not be broken up and confused, since program format is one of the beauties of Pascal.

I assume by now you have your Pascal system at least up and running and understand how to execute and compile programs for your system. Since I will be using the Lucidata Pascal, it may differ from yours, so it's up to you to know any differences. So you know what the limits and features of the Lucidata Pascal are, I print the following. Much of this may not mean too much to you at the present time, but it will as we go on. It may be a good idea to keep this handy.

DATA TYPES	COMMENTS	BYTES OF STORAGE
INTEGER	+/- 32767	2
BYTE	0..255	1
REAL	+/-1.7E+/-39 (9 DIGIT PERCISION)	5
BOOLEAN	TRUE/FALSE	1
CHAR	7 BIT ASCII	1
ALFA	ARRAY (1..6) OF CHAR	6
SET	64 MEMBERS	8
ENUMERATED	USER DEFINED	1
ARRAY	7 SUBSCRIPTS	1..32767
RECORD	TAGGED & UNTAGGED VARIANTS	1..255
FILE	ANY DECLARED TYPE	
LABEL	0..2047	

STATEMENTS

IF..THEN..ELSE, CASE..OF, FOR..TO/DOWNTO..DO, WHILE..DO, REPEAT..UNTIL, GOTO, BEGIN..END

STANDARD PROCEDURES

RESET, REWRITE, POSITION, READLN, WRITELN, READ, WRITE, HALT, POKE

STANDARD FUNCTIONS

ABS, SQR, TRUNC, ROUND, ORD, CHR, SUCC, PRED, ODD, EOF, EOLN, PEEK, USER, UNPACK, CARD

OPERATORS

*, /, +, -, OR, NOT, AND, DIV, MOD, IN, =, <>, <=, >=

IDENTIFIERS

UNLIMITED LENGTH--FIRST 6 CHARACTERS SIGNIFICANT (Pascal Standard is 8)

RUN-TIME CONFIGURATION OPTIONS

HIGH MEMORY LIMIT
STACK SIZE
AUTOMATIC MACHINE LANGUAGE USER FUNCTION LOAD
LINE BUFFERED OR CHARACTER MODE TERMINAL INPUT

ADDITIONAL FEATURES

FILE I/O TO USER SUPPLIED ASSEMBLY LANGUAGE CODED ROUTINES
RANDOM FILE ACCESS TO PROGRAM DEFINED LOGICAL RECORDS
VIRTUAL MEMORY OPERATION
HEX LITERAL CONSTANTS

HARDWARE REQUIREMENTS

24K HDOS SYSTEM (40K RECOMMENDED)
SINGLE DISK DRIVE (2 DRIVES RECOMMENDED)

Since I covered the general Pascal program format last month, we will now jump right into a simple program. I will list the BASIC program and the Pascal to do the same thing, for a comparison. This program will simply output two lines to the console. It is a simple program requiring just the basic Pascal structure without procedures or variables.

```
10 REM WRITTEN IN B.H. EXTENDED BASIC
20 REM PROGRAM TITLE SAMPLE ONE
30 REM THIS PROGRAM WILL OUTPUT THREE LINES TO THE CONSOLE
40 PRINT "HERE IS MY FIRST";
50 PRINT " ATTEMPT AT PASCAL"
60 PRINT
70 PRINT "SEE HOW EASY IT IS?"
80 END
```

Note lines 10 through 30 are remark statements used only for documentation purposes. Line 40 will print HERE IS MY FIRST ,and since a semicolon is used at the end of the print statement, the next line will print right after the word FIRST since a new line is not generated. Line 50 will print ATTEMPT AT PASCAL on the same line as the first. Line 60 causes a blank line to be printed. Line 70 will start a new line and output SEE HOW EASY IT IS? Line 80 tells BASIC there's no more. Here's how it will look.

```
HERE IS MY FIRST ATTEMPT AT PASCAL

SEE HOW EASY IT IS?
```

Now the Pascal version.

```
(* PROGRAM LANGUAGE--LUCIDATA PASCAL

   PROGRAM TITLE--SAMPLE ONE

   PROGRAM SUMMARY--OUTPUT THREE LINES TO THE CONSOLE  *)

PROGRAM SAMPLEONE;

BEGIN
  WRITE ('HERE IS MY FIRST');
  WRITELN (' ATTEMPT AT PASCAL');
  WRITELN;
  WRITELN ('SEE HOW EASY IT IS?')

END.
```

In Pascal, comments to be ignored by the program are enclosed with (* *), or in some Pascal's the curly braces {* *}, at the start and end of the comment. Some also support nested comments such as (*{ }*). As you can see, it does not have to be on the same line. This is good form for documentation as we can see at a glance the program language, name, and summary--sweet and simple. The actual Pascal program starts with the program name, which must be followed by a semicolon, in this case PROGRAM SAMPLEONE;. Note that although 6 characters are significant, the title and other Pascal words and identifiers can be any length, just so the first six are unique. In this case, SAMPLE is the first six characters which must be unique. Every Pascal program MUST have a program name, and always followed by a semicolon. In fact, most Pascal statements, or lines, must have a semicolon, which is called a statement separator.

The program name is only for documentation and is not used by the compiler for anything other than an indication of where the source code begins. The actual load module (.BIN or .ABS) need not have any relationship to this internal name.

Before I get too far, let me point out that in this program, the words PROGRAM, BEGIN, and END are reserved words. This means they have a special meaning in Pascal and cannot be used for variable names or identifiers. So what's an identifier? An identifier is a name (could be a variable, constant, program name, procedure name) which a programmer uses to 'identify' various sections of a Pascal program, or hold a quantity. The identifier may be any length, although only the first six (8 in most Pascal's) characters are significant. They must start with a letter, and may contain only letters and numbers, special characters (* / % # etc) are not allowed.

Following the program name we have the main program block, or body, since in this case no procedures are used. It is everything between and including the reserved words BEGIN and END. Of course, "BEGIN" tells Pascal this is where the program body begins and "END" tells Pascal it has reached the end of the program and can return to the operating system.

Two common Pascal reserved words used for output are WRITE and WRITELN. WRITE causes the output device to stay on the same line after output (like the semicolon after a PRINT in BASIC. WRITELN causes the output device to advance to the next line after the output. Whereas in BASIC you simply enclose your output, or string, in quotes " ", in Pascal you use parenthesis and single quotes (' '), although some Pascal's may require double quotes (" "). Note again, that at the end of each WRITE or WRITELN command the semicolon is placed as a statement seperator. This is required, except usually just before the line containing the END statement. There are other cases where you may not use semicolons, and these will be discussed as they come up.

When Pascal encounters WRITE ('HERE IS MY FIRST'); , it will output that line to the console and stop without advancing to a new line. The next WRITELN causes the words ATTEMPT AT PASCAL to be output right after the word FIRST, that's why there was a space left before the A in ATTEMPT. It also causes the console to move to the next line. The following line, WRITELN; causes a blank line, to be outputted. The following WRITELN prints SEE HOW EASY IT IS? on the following line, and moves to the next before the END.

Note the period after END. It is required to tell Pascal this is the actual program end. This is important since you have BEGIN and END in procedures also, but the END of a procedure, for instance, contains a semicolon and not a period.

You can see just by this simple program the structure and modularity of Pascal. We will see how you can have many of these BEGIN...END's in a program in modules, and call them as you need them. It makes writing, understanding, and debugging of the program much simpler.

So you say, big deal, all that work and learning to print a few lines on the screen? Well, you have to start somewhere. So moving right along, an important symbol to learn in Pascal is the ASSIGNMENT SYMBOL which is about equal to the equal sign in BASIC. It looks like this := . In Pascal, a value is assigned to a variable with this assignment statement, which means assign the value to the variable. For example, TOTAL := 29.35; means the identifier TOTAL now has the value 29.35. That's easy enough. The same rules apply to this variable name (TOTAL) as mentioned before for identifiers.

Not to get too confusing, but the ASSIGNMENT symbol is used for variables and not for constants. For constants, the equal sign is used, such as PI = 3.14; . Just as with MBASIC, variables and constants must be DECLARED before they can be used. If you noticed in the chart at the beginning of this article, real numbers require more space than integers, and execute slower. When Integers can be used, it is wise to use them rather than REAL. The reason for DECLARING variables and constants is so all variables, procedures and functions will be clearly 'typed' so the compiler can check on incompatabilities and the reader will know what is going on.

How do you declare a variable or constant? Easy. Remember from part one that constants must be declared before variables, and this all happens after the program title. Below is an example I will use as we start building a program.

```
PROGRAM TAXRATE;

CONSTANT TAXRATE = 0.04;
         ESC     = 27;

VAR      PRICE, SALESTAX, TOTAL      : REAL;
         A                           : CHAR;
         FINISHED                    : BOOLEAN;
```

As Pascal format requires, the program title is first, followed by the constant and variable declarations. We are declaring TAXRATE to be a constant and equal to 0.04 (4 % sales tax in Wisconsin), and ESC to be a constant equal to 27 (escape sequence code for H19 terminals). Note the leading zero in the constant TAXRATE. It is required for decimal usage. The REAL variables are PRICE, SALESTAX, and TOTAL. These values can and will change during the program, thus they are variables. I assume you all know what REAL and INTEGER is, what about CHAR and BOOLEAN? CHAR describes an identifier (A in this case) which will hold one ASCII character, good for inputting Yes and No answers for example. BOOLEAN describes an identifier (FINISHED in this example) that holds a value of "false" or "true"--a logical value.

Not bad so far, if you get boggled, slow down and read it over because these are the basics for most every Pascal program. OK, now we have to have a way to manipulate data to get results. That's were the Arithmetic Operators come in, much like BASIC with a few differences.

* Addition +
* Subtraction -
* Multiplication *
* Division / (used when the result is of type REAL i.e. REAL := INTEGER/INTEGER.
* Division of two integers producing an integer result DIV
* Integer remainder of an integer division MOD

Now that you know these, you can begin to write some simple programs that do something. It would be more helpful, however, to be able to input from the terminal so we can run real time programs. There are two common commands for this, just as there were for output, READ and READLN. We will deal mainly with READLN. READLN reads a line of information terminated by a RETURN from the keyboard. For now that's all we will be concerned with since the difference between the two becomes confusing.

So now we're about all set. We can read information, write it, assign variables and constants, and preform mathematics. I will show an example of mathematics now.

```
SALESTAX := TAXRATE * PRICE;
TOTAL := PRICE + SALESTAX
```

That was easy and does not require much explanation. First we calculate the value computed by TAXRATE multiplied by PRICE, and assign it to the variable SALESTAX. Next we compute and assign to the variable TOTAL the sum of PRICE plus SALESTAX. No problem. Now let's write a procedure to do it.

A procedure is a module which all by itself preforms a small part of a program, structured something like a mini Pascal program itself. Note the example.

```
PROCEDURE CALCULATE;
    BEGIN
      SALESTAX := TAXRATE * PRICE;
      TOTAL := PRICE + SALESTAX
    END;
```

That's it! This procedure named (or identified) as CALCULATE will preform the mathematical functions when called from the main program. Note the BEGIN and END;, and the indenting to make it more readable. It's not required, but is part of it's structured readable form. Get into this habit and make life easier. Note the use of the assignment symbol as previously discussed, and the placement of the semicolons.

Here's another PROCEDURE which will be used often. It will clear the screen of the H19/H89 terminal when it's called.

```
PROCEDURE CLEARSCREEN;
    BEGIN
       WRITE(CHR(ESC),'E')  --note no semicolon here because before END
    END;
```

Note we have already defined ESC as 27 under constant declaration. All we are doing is sending the escape sequence to the terminal to clear the screen. Now perhaps you can see the beauty of modularity. It's simple, and once it is done, we can use it over and over again for other programs knowing that it's been tested and debugged. You can build major programs with small modular procedures. Of course some of these can get large and complex, but the idea still holds. A procedure or function is actually a complete program within a program--truly a full subprogram!

I guess this is all for this month, look it over and play around with it. If you are getting interested and want to jump in, it's not too late, get yourself a version of PASCAL and meet me next month. We'll finish this taxrate program and introduce some new concepts. See you then.

The "Sector Zapper"

USING HARD SECTOR DISKETTES
IN A SOFT SECTOR ENVIRONMENT

By: RICHARD L. KING
8 Van Dyke Rd. Box 505
Hollis, NH 03049

I was just sitting there minding my own business at a HUG-EM meeting in Wellesley Hills Mass., when someone asked the guest speaker (Gerry Kabelman), if it would be possible to use the existing 10 sector diskette media on the newly announced soft sector diskette controller (H-37)? When Gerry said that he could not answer that question, I stated that it could easily be done by making a small modification to the INDEX/SECTOR line, and it could probably be done with two IC's. Gerry challenged me to prove it by writing an article for REMark, so here it is.

It is important to note that this circuit will NOT allow you to recover the data on a 10 sector diskette, but it will let you use the media on a soft sectored system.

The problem is that the INDEX/SECTOR line contains only INDEX pulses when running a soft sectored diskette, and contains 10 equally spaced SECTOR pulses in addition to each INDEX pulse when running a 10 sector diskette. A typical 5" diskette drive manual (I used the SIEMENS FDD 100-5B) shows what the signal on the INDEX/SECTOR line looks like. (See Figure 1.)

The best way to separate the INDEX pulse from the SECTOR pulses, is to generate a signal to mask the SECTOR pulses and prevent them from getting to the output. Another signal will be needed to position the masking signal around the SECTOR pulses. We can do this with two one-shots or timers. Due to the relatively long time intervals needed, we will use the 555 timer circuit.

The SECTOR pulses will fire timer T1, and the end of T1 will fire T2. Notice that in Figure 2 negative pulses that occur while TIMER 2 is OFF are the INDEX pulses. Each SECTOR pulse causes the timers to cycle and mask the next SECTOR pulse.

Figure 3 shows that a soft sectored diskette will still work properly when using this scheme. Even though the INDEX pulse itself sets T1, T2 comes on later and masks -- nothing.

Figure 4 shows how the timers are connected to perform the required masking.

1. INDSEC- fires T1, which in turn fires T2.
2. IC3a inverts INDSEC- making the final logic a bit simpler.
3. IC3b inverts T2+ and allows SW1 to disable T2, so we can disable the curcuit for test purposes. When the switch is closed, IC3b is disabled, and its output can never go to ground. Therefore, the output can't be affected by the T2 pulses.
4. IC3c ANDs together INDSEC+ and the negation of T2. The only time the output of IC3c can go low is when INDSEC+ and T2S- are both high

(during an INDEX pulse).

Figure 5 shows how to wire a 555 timer to cause it to act as a one-shot. It is AC coupled, and triggers on a negative going pulse. It is not re-triggerable, that is it will not restart its cycle if it receives another input trigger before its cycle is complete. This is necessary, because we do not want the INDEX pulse to affect it when working with 10 sector media.

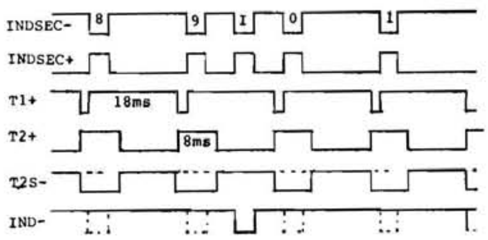
The circuit used for the 555 timers is standard, and can be found in several articles using 555s. All components are the same in both timers, except for the timing resistor (Rt).

You will need an oscilloscope to adjust the unit. The easiest way is to close SW1 while running the system, and adjust T1 and T2 for the proper times. If you have a dual trace scope, you can actually adjust the positioning of T2 to bracket the SECTOR pulses.

If you happen to have a lot of 16 sector disks lying around, this circuit can be made to work with them, by changing T1 to 11 ms, and T2 to 6 ms. However, it will NOT work with 10 and 16 sector diskettes intermixed.

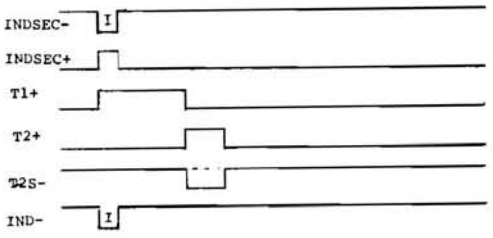
The total cost to build this unit should be less than $15.00, even if your junk box is currently empty. If there is sufficient interest, I would be willing to build and sell finished units.

I can't tell you how to connect this unit to an H37, because I don't own one. All I can say is that this unit will remove the SECTOR pulses from the INDEX/SECTOR line (I breadboarded it on my H17).



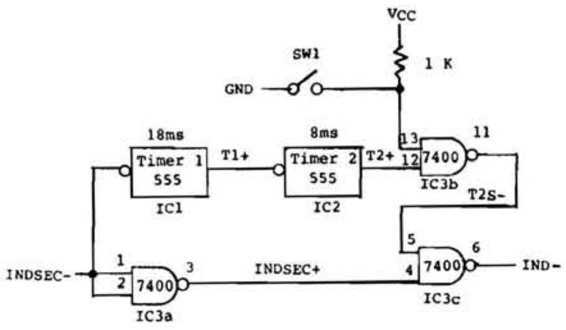
HARD SECTOR DECODING

Figure 2



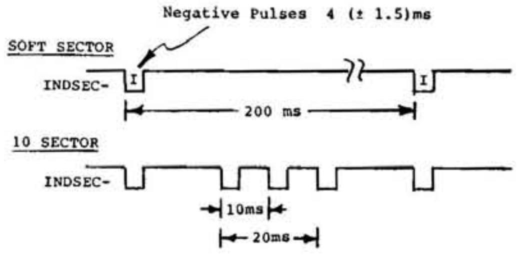
SOFT SECTOR DECODING

Figure 3



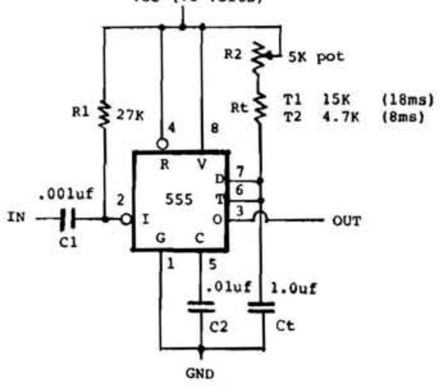
CIRCUIT DIAGRAM

Figure 4



INDEX/SECTOR WAVEFORM

Figure 1



BASIC DIAGRAM OF TIMERS 1 AND 2

Figure 5

Disk Options for the H89/Z89

With the introduction of two additional types of drives it is high time someone sat down and made selecting drives for the H89/Z89 simple.

You may use any two of the four different controllers available in one machine at a time. That means both hard and soft sectored disks may be used at the same time using the proper controllers and separate drives for each type of disk (hard or soft sectored).

In just a few words the above paragraph tells the entire story. Now let me try to explain the words of that paragraph in detail.

The easiest way to look at the possible options is to first take a look inside the H89/Z89 and look at the upper right three slots P504, P505 and P506. (P510, P511 and P512 are the lower portions of the above connecters, respectively.) These three slots are the interface slots of the H89/Z89 computer.

The H89/Z89 may contain any combination of two different types of disk interface cards such as the Z-89-37 and Z-89-67 or H-88-1 and Z-89-47.

There are a couple of restrictions that need to be brought out into the open and must be observed when adding additional interface cards to the H89/Z89.

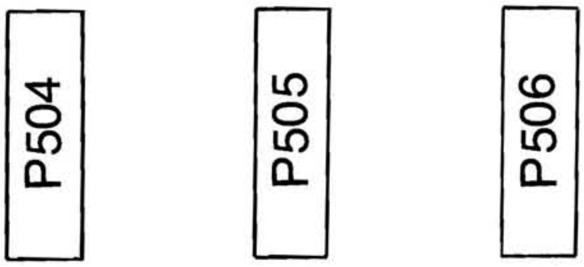
First using the H88-1 single-sided single-density controller card requires that it always be installed in the slot labeled P506.

Second, when using the Z-89-37 double-sided double-density controller may only be used in slots P504 or P505.

Also the serial input/output board may only be used in slots P504 or P505.

The other two controller cards (Z-89-47 or Z-89-67) may be used in any of the three slots.

The above information should be reviewed in addition to the information provided within the Heath Catalog.



P504	(or)	P505	P506
Z-89-37		I/O	H-88-1
Z-89-47		I/O	H-88-1
Z-89-67		I/O	H-88-1
Z-89-37		I/O	H-88-1
Z-89-37		I/O	Z-89-47
Z-89-37		I/O	Z-89-67
Z-89-47		I/O	H-88-1
Z-89-47		I/O	Z-89-67
Z-89-67		I/O	H-88-1
Z-89-67		I/O	Z-89-47

Local HUG News

Richmond Heath Users' Group at 1724 Blakemore Rd., Richmond, VA 23225 meets the second Wednesday of each month at various locations. Contact Jim Scott at (804) 323-2925.

Len Bateman of Illinois Heath Users' Group (I-HUG), has announced the new officers of the group: President, Len Bateman; Vice-President, Jim Sossong; Secretary, Bob Bileski; Treasurer, Jim Jones. I-HUG meets the third Wednesday of each month, except December, at the Downers Grove Illinois Heath Electronic Center.

The newest formed HUG group is the Rochester Heath Users' Group (RHUG) in Rochester, New York. The group meets on the last Tuesday of every month at 7:30 p.m. at the Rochester Heathkit store on Henrietta Rd. Joanne Lang, the Membership Chairperson, can be contacted at 663-0193, or write to RHUG, 937 Jefferson Rd., Rochester, NY 14623.

OOPS! The contact information given for the PittsburgHUG in Issue 19 (August) of REMark was incorrect. You may contact John Schultz by calling (412) 793-6781. Sorry about that John! Hope this didn't cause too many problems on your end.

BUGGIN' HUG

From the desk of Ted Benglen of Micro-Interface (for the H8/H89 Computers:

Dear Bob,

It has been some time since we talked on the phone about M.I.-8 and the products we offer to the Heath Computer users. Next year, we will be developing new products for the H-89, the first of which will be a parallel clock board with 6 - 9 parallel ports and a real time clock. Also to be announced soon, is a temperature acquisition and A/D converter with 1/2 degree resolution. We have been hard at work supporting the H-8 and plan to continue with several new products next year for both the H-8 and H-89.

M.I.-8 also makes special prices available to HUGs, and will continue to follow this practice in the future. Any HUG (Heath User Group) wishing to discuss a product and a special price should contact me (just ask for Ted). We are offering a free heat sink extension to any H-8 owner that writes and requests one. The heat sinks have shown to lower the temperature on the boards in the H-8 up to 65% and require no modification to the H-8 enclosure. I sincerely hope this information will be worthy of passing on to the Heath Computer users.

Ted Benglen,II
M.I.-8
822 E. County Rd. 30
Ft. Collins, CO 80525

Phone: (303) 669-4116

Note: Give M.I.-8 a call about the existing hardware and software they have made available for the H-8 and H-89 Computers. It's GOOD stuff!!

Dear HUG,

Recently, I purchased the replacement Monitor Rom for the H-89 offeredby UltiMeth Corporation. I found it to be a very good product. The additional features that it gives to the H-89 are surely needed, and the increased processor speed is welcome. For people who do Assembly Language programming, it is a must. If you have any other questions, you can call Dean Gibson at (213) 539-4276 between 9:00 AM and 12:00 PM Pacific. He is more than willing to answer any questions. I thought that your readers would like to know.

George E. Orndorff
Rt2 Box 20A
Charles Town W. VA 25414

Changing your address? Be sure and let us know since the software catalog and REMark are mailed bulk rate and it is not forwarded or returned.

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

IS THE INFORMATION ON THE REVERSE SIDE CORRECT?
IF NOT FILL IN BELOW.

Name ____________________

Address ____________________

City-State ____________________

Zip ____________________

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

RENEWAL RATES		NEW MEMBERSHIP FEE IS:
US DOMESTIC	$15 ☐	$18 ☐
CANADA	$17 ☐ US FUNDS	$20 ☐
INTERNAT'L*	$22 ☐ US FUNDS	$28 ☐

* Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

PROGRAMMING LANGUAGES

885-1038	WISE on Disk H8/H89		$ 18.00
885-1042	PILOT H8/H89		$ 19.00
885-1059	FOCAL-8 H8/H89		$ 25.00
885-1078	HDOS Z80 Assembler		$ 25.00
885-1085	PILOT Documentation		$ 9.00
885-1086	Tiny Pascal H8/H89		$ 20.00
885-1094	HUG Fig-Forth H8/H89 2 Disks		$ 40.00

BUSINESS, FINANCE AND EDUCATION

885-1047	Stocks H8/H89		$ 18.00
885-1048	Personal Account H8/H89		$ 18.00
885-1049	Income Tax Records H8/H89		$ 18.00
885-1051	Payroll H8/H89		$ 50.00
885-1055	Inventory H8/H89	*	$ 30.00
885-1056	Mail List H8/H89	*	$ 30.00
885-1070	Disk XIV Home Finance H8/H89		$ 18.00
885-1071	SmBusPkg III 3 Disks H8/H19 or H89	*	$ 75.00
885-1091	Grade and Score Keeping	*	$ 30.00
885-1097	Educational Quiz Disk H89 or H8/H19	*	$ 20.00

DATA BASE MANAGEMENT SYSTEMS (DBMS)

885-1107	Amateur Radio Logbook and TMS		$ 30.00
885-1108	Telephone/Mail Info. System	*	$ 30.00
885-1109	Retriever (2 disks)		$ 40.00
885-1110	Autofile		$ 30.00

AMATEUR RADIO

885-1023	RTTY Disk H8 Only		$ 22.00
883-1106	Morse-89 H8/H19 or H89		$ 20.00

* Means MBASIC is required

H11 SOFTWARE

885-1008	Volume I Documentation and Program Listings (some for H11)		$ 9.00
885-1033	HT-11 Disk I		$ 19.00

CP/M SOFTWARE (5-inch only)

885-1201	CP/M (TM) Volumes H1 and H2	%	$ 21.00
885-1202	CP/M Volumes 4 and 21-C	%%	$ 21.00
885-1203	CP/M Volumes 21-A and B	%%	$ 21.00
885-1204	CP/M Volumes 26/27-A and B	%%	$ 21.00
885-1205	CP/M Volumes 26/27-C and D	%%	$ 21.00
885-1206	CP/M Games Disk	%%	$ 21.00

The above CP/M products are 2 disks each.

885-1207	TERM and H8COPY		$ 20.00
885-1208	HUG Fig-Forth H8/H89 2 Disks		$ 40.00
885-1209	Dungeons and Dragons Game MBASIC and H89 or H8/H19		$ 20.00
885-1210	HUG Editor		$ 20.00
885-1211	Sea Battle Game for CP/M		$ 20.00
885-1212	CP/M Utilities I		$ 20.00
885-1213	CP/M Disk Utilities		$ 20.00
885-1214	Amateur Radio Logbook		$ 30.00

% Means CP/M 1.43 only (ORG-4200)
%% Means CP/M 1.43 or 2.2 (Heath)
Other CP/M disks are for 2.2

MISCELLANEOUS

885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-4	HUG Binder	$ 5.75

CP/M is a registered trademark of Digital Research Corp.

Heath Users' Group
Hilltop Road
St. Joseph MI 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable, please do not return.

885-2023