RADEX-1Ø

TABLE OF CONTENTS

The International Jewelry Guild, Inc. and author have taken due care in preparing the program (including research, development, and testing) to ascertain its effectiveness and to make no expressed or implied warranty of any kind, with regard to the program or the supplementary documentation. In no event shall the author or International Jewelry Guild, Inc. be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance or use of RADEX-10 programs or program modules.

(iv) TRANSFERRING RADEX-10 TO A DISK WITH A DOS

Your RADEX-10 is supplied on a formatted disk without a DOS. Before you use RADEX-10, the six program modules must be copied to a disk with a DOS. It is recommended that you use the APPARAT ENHANCED DOS rather than TRS-DOS 2.1. The reasons for this recommendation are fully discussed in Section XXIV.

To transfer RADEX-10 modules to a 'system' diskette, use the 'COPY' function of your DOS to transfer the programs to your DOS diskette. You may also 'LOAD' the programs from BASIC and 'SAVE' them to a diskette with DOS, on drive 0.

Once you have transferred the programs, use that as a master diskette for making additional 'COPIES' of RADEX-10 with a DOS.

(v) HARDWARE REQUIREMENTS

Because of the ability of RADEX-10 to span diskettes and maintain its own end-of-file records, two disk drives are required. The following are the minimum hardware requirements needed to support the program:

Memory.....32K
Disk drives.....2
Line printer.....NOT required

I INTRODUCTION

1.0 USE RADEX-10 may be used to maintain any type of file or record requiring fast access, maximum use of disk 'space' and ease of operation. You can keep information files that are trivial in nature, files with important financial data or sensitive information requiring immediate retrieval.

Business and scientific applications programs, for instance, have great need for a database --- many programs require that several programs be able to access the data; since each programs' data is structured differently, each would normally require "hard coded" program material for each file to read and write to each file.

RADEX-10 solves this problem by providing "common code" that is able o read the file parameters and then handle the fielding, naming, input, output and various other tasks on its own.

To use 'RADEX-10', you should be familiar with disk operation. It is recommended that you read and be familiar with "TRSDOS & DISK BASIC REFERENCE MANUAL", published by Radio Shack for the TRS-80 disk operating system, TRSDOS VERSION 2.1 and DISK BASIC VERSION 1.1.

It will not be necessary to know all there is to know about the disk system commands, since 'RADEX-10' will manage your files for you; however, the specifications for file names, extensions, passwords, etc., will be of value. The LEVEL II manual will also provide additional explanation as to 'DATATYPE' (integer, single precision, etc.).

If you plan to use 'RADEX-10' as a "STAND-ALONE" program, the program is almost self-explanatory, even though the code and internal operation is complex. In use, you will find it is simple and 'transparent'; in other words, you will not be aware of the various complexities of the program because it is managing itself.

The easiest way to learn anything is to do it. Before you actually put up files onto RADEX-10, read this manual, create a file, do some file maintenance, create and run some reports; THEN put actual records on file.

Including 'RADEX-10' modules in your applications will require the addition of the "APPLICATIONS INTERFACE MODULE". This module will provide you with all the facilities to pass information back and forth between your program and 'RADEX-10'.

Additional modules that may be added to 'RADEX-10' will be discussed at the end of this documentation. See Appendix A.

1.1 IMPORTANCE OF DATA ORGANIZATION Most people confuse 'data' with 'information'. For instance, the numbers "18, 32, 36", by themselves are meaningless. They are just numbers. In other words, 'Raw Data'. To be meaningful they must be organized into some scheme that can be interpreted. If we rearrange them (36-18-32) they might suggest something familiar to most (males, at least). See, just by putting the numbers into a familiar pattern, they cause an image to jump into our mind.

We can also give them meaning by giving them labels and by always putting the proper number beneath its assigned label:

Quantity On Hand	Quantity Back Ordered	Minimum Order Quantity
36	18	32

By giving each number a label, such as "Quantity on Hand", the number suddenly acquired a meaning. NOW, we have INFORMATION; before, all we had was data. The point is, that with the organization of data, we can see information that is valuable to us.

By organizing the data in the computer, we have a powerful tool at our disposal. If the program will ALWAYS retrieve the data in the correct format, ALWAYS 'write' the data to the correct place, ALWAYS get the right group of data and ALWAYS keep it organized, no matter how the data was stored;, THEN we have a RADEX-1Ø MANAGER. Your RADEX-1Ø program is a database manager.

1.2 The International Jewelry Guild,, Inc. has a funny name to be in the computer software business, but it was no accident. We write programs that deal with the jewelry business --- jewelers are, for the most part, small businesses. In addition to specific programs that deal with diamond prices, gold prices and the like, they need accounts receivable, payroll, general ledgers and all the other programs that go with running a business.

We looked at a lot of software packages we thought we might be able to sell, but they were lacking in one important area --- poor file structure. Files are the cornerstone of every business program. To solve that problem, we decided to tackle the files problem first. It also became obvious that if we could solve our problem, we could solve it for others too.

RADEX-1Ø is the result; and as a result of the result, here we are ...in the software business.

1.3 CURRENT PRODUCT RADEX-1Ø is the primary stand alone program module. Its code is very compact (no sub-module requires more than 6K of computer memory). You can create your files, add data, remove data, write reports, make labels and do other tasks without writing a single byte of code!

1.4 FUTURE PRODUCT RADEX-1Ø is only the beginning. Many additions are scheduled for release over the comming year. Each addition (or MODULE, as we call it), can be added to RADEX-1Ø to either enhance its operation, provide additional utility or to do another job involving the data management process. A description of each modules' function is provided in Section XIX.

II PROMPTS, MENUS AND MESSAGES

2.Ø We have avoided, as much as possible, the use of obscure 'computer' terms in RADEX-1Ø. Plain English, as commonly used in everyday life, is used throughout the program. RADEX-1Ø is made up of

six subprograms:

1. DATABASE.....Control program
2. CRATE.....Creates the data base file
3. WFL.....Handles all file I/O
   Consists of 7 submodules
4. REPORT.....Creates reports & report file
5. OUTPUT.....Runs reports
6. FILCHK.....Prints file parameters

Each subprogram has a menu, with the exceptions of FILCHK and CRATE. FILCHK simply prints the file parameters without any additional input from the user and returns you to the RADEX-10 menu. CRATE leads you through the process of creating a data base file and requires no menu, as such. (See figures 1, 2, 3 and 4.)

```
==========================================================
        1.  CREATE DATA BASE
        2.  FILE MAINTENANCE
        3.  CREATE REPORTS
        4.  RUN REPORTS
        5.  PRINT FILE PARAMETERS
        6.  END PROGRAM
==========================================================
```

(Figure 1) RADEX-10 control program menu

```
==========================================================
  TOTAL RECORDS ON FILE =>  1245  (INCLUDES INACTIVE RECORDS)
==========================================================
        1.  ADD A RECORD TO THE END OF THE FILE
        2.  ADD A RECORD TO THE FIRST OPEN SPOT
        3.  DISPLAY A RECORD TO THE VIDEO
        4.  CHANGE ENTIRE RECORD
        5.  UPDATE A RECORD
        6.  MAKE RECORD INACTIVE (DELETE RECORD)
        7.  CLEAN UP FILE (MAKE OPEN SPOTS OF INACTIVE RECORDS)
        8.  RETURN TO RADEX-10 MENU
        9.  END PROGRAM
==========================================================
```

(Figure 2) FILE MAINTENANCE menu

```
==========================================================
                1.  HORIZONTAL HEADINGS
                2.  VERTICAL HEADINGS
                3.  LABELS FORMAT
==========================================================
```

(Figure 3) CREATE REPORTS menu

```
==========================================================
                ENTER '↑' TO RETURN TO MENU
==========================================================
        1.  (Your report #1)
        2.  (Your report #2)
        3.       .
        4.       .
        5.    (Etc.)
WHICH REPORT WOULD YOU LIKE TO EXECUTE?
==========================================================
```

(Figure 4) Report menu

You will find that the menus, prompts and error messages are in plain English. "Computerese" has been avoided as much as possible. A typical error message is as follows:

```
----------------------------------------------------------
YOU HAVE THE WRONG FILE NAME OR THE WRONG DISK ON DRIVE 1
THE DISK ON DRIVE 1 DOES NOT HAVE THE CORRECT FILE PARAMETERS
FOR THE FILE NAME YOU HAVE SPECIFIED
PLEASE CHECK THE DIRECTORY ON DRIVE 1
----------------------------------------------------------
```

This message is telling you, in plain English, that either you have used the wrong file name, or you have the wrong diskette mounted on disk drive 1. The message will be displayed for several seconds; and then, you will be returned to the RADEX-10 menu.

Of course, it is not possible to catch every error that can be made. Some of the 'non-trapped' errors are discussed in a later section of this manual.

III DEFINITIONS

The following definitions apply to the 'RADEX-10', as used on the TRS-80.

ACCESS The operation of seeking, reading or writing data on a storage unit (in this case, the diskette).

ACCESS TIME The time that elapses between any instruction being given to access some data and that data becoming available for use.

ACTIVE RECORDS TABLE (ART) A table of binary values in which the relative position of a single value determines the status of a record with the same relative position; i.e., the Nth binary number determines the status of the Nth record. EXAMPLE: If the 8th binary number in the table is a zero, then the 8th record is inactive.

conversely, if the 8th binary number in the table is a one, then the th record is active.

ADDRESS An identification (number, name, or label) for a location in which data is stored.

ALGORITHM A computational procedure.

ALPHANUMERIC (CHARACTERS) A generic term for numeric digits, alphabetic characters, punctuation characters and special characters.

ALPHANUMERIC STRING A group of characters which may include digits, alphabetic characters, punctuation characters and special characters, and may include spaces. (NOTE: a space is a 'character' to the computer, as it must generate a code for spaces as well as symbols.)

BOOLEAN A form of algebra applied to binary numbers which is similar in form to ordinary algebra. It is especially useful for logical analysis of binary numbers as used in computers.

'BOOT' - BOOTSTRAP A machine language program file that is put onto every diskette by the 'FORMAT' routine. This routing is invoked when reset or power-on occurs. It automatically loads the necessary programs (SYSØ/SYS) to cause the computer to respond to the DOS commands; i.e., the machine is 'BOOTSTRAPPED' or 'BOOTED' into operation.

BUFFER A small area of memory used for the temporary storage of data to be processed.

DATA BASE A collection of interrelated data stored together with controlled redundancy to serve one or more applications. The data are stored so that they are independent of programs which use the data. A common and controlled approach is used in adding new data and in modifying and retrieving existing data within a data base. A system is said to contain a collection of data-based information if they are disjoint in structure.

DATA-BASE MANAGEMENT SYSTEM The collection of software required for using a data base.

DATA ELEMENT Synonymous with 'DATA ITEM' or 'FIELD'.

DATA TYPE The form in which data is stored; i.e., integer, single precision, double precision, 'alphanumeric' character strings or 'strings'.

DIRECT ACCESS Retrieval or storage of data by a reference to its location on a disk, rather than relative to the previously retrieved or stored data.

DIRECTORY A table giving the relationships between items of data. Sometimes a table or an index giving the addresses of data.

DISPLACEMENT A specified number of sectors, at the top or beginning of the file, in which the 'bookkeeping' and file parameters are stored for later use by the various 'RADEX-10' program modules.

DISTRIBUTED FREE SPACE Space left empty at intervals in a data layout to permit the possible insertion of new data.

DOUBLE PRECISION A positive or negative numeric value, 16 digits in length, not including a decimal point (EXAMPLE: 99999999999999.99).

DYNAMIC STORAGE ALLOCATION The allocation of storage space by a procedure based on the instantaneous or actual demand for storage space by that procedure, rather than allocating storage space to a procedure based on its anticipated or predicted demand.

EMBEDDED POINTERS Pointers in the data records rather than in a directory.

ENTITY Something about which data is recorded.

EOF Initials for 'END OF FILE'. It is common practice to say that the EOF is record number nn or that the EOF is byte 15 of sector 12. Hence, it is a convenient term to use in describing the location of the last record or last byte in a file.

EXTENT A contiguous area of data storage.

FILE PARAMETERS The data that describes or defines the structure of the file.

FIELD See 'DATA ITEM'.

GRANULE Unit of 5 sectors. On the TRS-80 disk operating system, a 'granule' is the basic unit of disk storage allocation. The diskette 'DIRECTORY' file keeps track of free and assigned disk space in terms of 'granules'.

HEADER RECORD A record containing common, constant or identifying information for a group of records which follow.

INDEX A table used to determine the location of a record.

INDIRECT ADDRESSING Any method of specifying or locating a storage location, whereby, the key (of itself or through calculation) does not represent an address. For example, locating an address through indices.

INSTRING (INSTRING SEARCH) Refers to the capability of locating a substring of characters that may exist in another character string. An example would be: Substring = "THE" String = "NOW IS THE TIME". An INSTRING routine would locate the substring and return its starting position within that string. In this example, it would return a value

of eight.

INTEGER A natural or whole number. In the TRS-80 integer values may not exceed the range of +32767 to -32768.

INVERTED FILE A file structure which permits fast spontaneous searching for previous unspecified information. Independent lists or indices are maintained in records' keys which are accessible according to the values of specific fields.

INVERTED LIST A list organized by a secondary key --- not a primary key.

KEY A data item used to identify or locate a record or other data grouping.

LABEL A set of symbols used to identify or describe an item, record, message or file. Occasionally, it may be the same as the address in storage.

LIST An ordered set of data items. A 'chain'.

LOGICAL An adjective describing the form of data organization, hardware or system that is perceived by an application program, programmer, or user; it may be different than the real (PHYSICAL) form.

LOGICAL DATA-BASE DESCRIPTION A schema. A description of the overall data-base structure, as perceived for the users, which is employed by the data base management software.

LOGICAL FILE A file as perceived by an application program; it may be in a completely different form from that in which it is stored on the storage units.

LOGICAL OPERATOR A mathematical symbol that represents a mathematical process to be performed on an associated operand. Such operators are 'AND', 'OR', 'NOT', 'AND NOT' and 'OR NOT'.

LOGICAL RECORD A record or data item as perceived by an application program; it may be in a completely different form from that in which it is stored on the storage units.

MAINTENANCE OF A FILE (1) The addition, deletion, changing or updating of records in the database. (2) Periodic reorganization of a file to better accommodate items that have been added.

MULTIPLE-KEY RETRIEVAL Retrieval which requires searches of data based on the values of several key fields (some or all of which are secondary keys).

ON-LINE An on-line system is one in which the input data enter the

computer directly from their point of origin, and/or output data are transmitted directly to where they are used. The intermediate stages such as writing tape, loading disks or off-line printing are avoided.

ON-LINE STORAGE Storage devices and especially the storage media which they contain under the direct control of a computing system, not off-line or in a volume library.

OPEN RECORDS TABLE (ORT) A table of binary values in which the relative position of a single value determines the status of a record with the same relative position; i.e., the Nth binary number determines the status of the Nth record. EXAMPLE: If the 8th binary number in the table is a zero, then the 8th record is open. Conversely, if the 8th binary number in the table is a one, then the 8th record is on file.

OPERATING SYSTEM Software which enables a computer to supervise its own operations, automatically calling in programs, routines, language and data as needed for continuous throughput of different types of jobs.

PHYSICAL An adjective, contrasted with logical, which refers to the form in which data or systems exist in reality. Data is often converted by software from the form in which it is physically stored to a form in which a user or programmer perceives it.

PHYSICAL DATA BASE A data base in the form in which it is stored on the storage media, including pointers or other means of interconnecting it. Multiple logical data bases may be derived from one or more physical data bases.

PHYSICAL RECORD A collection of bits that are physically recorded on the storage medium and which are read or written by one machine input/output instruction.

POINTER The address or a record (or other data groupings) contained in another record so that a program may access the former record when it has retrieved the latter record. The address can be absolute, relative, symbolic; hence, the pointer is referred to as absolute, relative or symbolic.

RANDOM ACCESS To obtain data directly from any storage location, regardless of its position, with respect to the previously referenced information. Also called 'DIRECT ACCESS'.

RANDOM ACCESS STORAGE A storage technique in which the time required to obtain information is dependent of the location of the information most recently obtained.

READ To accept or copy information or data from input devices or a memory register; i.e., to read out, to read in.

RECORD A group of related fields of information treated as a unit by an application program.

RELATIONAL OPERATOR A mathematical symbol that represents a mathematical process to perform a comparison describing the relationship between two values (< less than....> greater than... = equal.... <> not equal... and combinations thereof (see TRS-80 LEVEL II manual, Section 1, Page 5). On the TRS-80, relational comparisons may be made on string values as well as numerical values.

RELATIVE (as pertains to position) An address or position that is referenced to a point of origin; i.e., X+20 is a specific positon, 20 places from the reference point. If the reference point was at 50, then the absolute position would be at 70 (50+20=70). Also, 50 (since it is the starting reference point) is at relative position 0.

SCHEMA A map of the overall logical structure of a database.

SEARCH To examine a series of items for any that have a desired property or properties.

SECONDARY INDEX An index composed of secondary keys rather than primary keys.

SECTOR The smallest addressable portion of storage on a diskette (a unit of 256 bytes on a TRS-80 diskette).

SEEK To position the access mechanism of a direct-access storage device at a specified location.

SEQUENTIAL ACCESS Access in which records must be read serially or sequentially one after the other; i.e., ASCII files, tape.

SINGLE PRECISION A positive or negative numerical value of 6 digits in length, not including a decimal point (EXAMPLE: 99999.9).

SORT To arrange a file or data in a sequence by a specified key (may be alphabetic or numeric and in descending or ascending order).

SUB-STRINGS SUB-STRING SEARCH See INSTRING

TABLE A collection of data suitable for quick reference, each item being uniquely identified either by a label or its relative position.

TRACK The circular recording surface transcribed by a read/write head on the disk. On the TRS-80 a track contains 10 sectors (2 granules).

TRANSACTION An input record applied to an established file. The input record describes some "event" that will either cause a new file record to be generated, an existing record to be changed or an existing record to be deleted.

TRANSPARENT Complexities that are hidden from the programmers or users (made transparent to them) by the software.

VECTOR A line representing the properties of magnitude and direction. Since such a 'line' can be described in mathematical terms, a mathematical description (expressed in numbers, of course) of a given 'direction' and 'magnitude' is referred to as a "vector".

WORKING STORAGE A portion of storage, usually computer main memory, reserved for the temporary results of operations.

WRITE To record information on a storage device.

IV VERSIONS AND RELEASES

4.0 A new "VERSION" of RADEX-10 would contain substantial changes and/or enhancements of the program.

A new "RELEASE", however, would only involve corrections and/or very minor changes in the program.

At present, the version is "1" and the release is "1". The version-release is "1.1".

V CONVENTIONS

5.0 In descriptions of syntax, commands and dialog with the RADEX-10 program, we'll use the following conventions in this documentation:

<ENTER>	Press the white 'ENTER' key.
<SPACE>	Press the space bar on the keyboard.
'filespec'	Substitute your own file name for 'filespec'.
<< .... >>	Double brackets indicate that the material is optional and may be omitted.
....	Horizontal or vertical dots indicate that the preceding material may be repeated.

5.1 FILE SPECIFICATION Specifications for file names follow the conventions as set fourth in the Radio Shack Disk Operating System Manual, Section 3, Page 6.

5.2 DATA TYPES Data types, INTEGER, SINGLE PRECISION and DOUBLE PRECISION are explained in DEFINITIONS, Section III of this manual. In specifying the data type in the CREATE program, all that is necessary for your entry is to type the number beside the description of the data type The program will automatically handle the byte allocation for the data type you have entered. The byte allocation for data types is as follows:

Integer ..... 2 Bytes

Single Precision ... 4 Bytes
Double Precision ... 8 Bytes

5.3 LSET - RSET These terms describe how an alpha field is to be justified within the field. LSET means that your input will be justified to the left side of the allotted spaces for that field; RSET will justify to the right. In the following example, 20 spaces are allotted for the field.

```
RSET = .........B.GOLDWATER
LSET = G.MC.GOVERN.........
```

Empty spaces are indicated by a period (.). RSET has the name justified to the right and LSET has the name justified to the left. The CREATE program will ask you which type of justification is required for the fields you have defined as alpha fields. Your response will be to enter a number next to the description (on the menu) of the type justification you desire.

VI DISK ORGANIZATION

6.0 PHYSICAL PROPERTIES OF THE DISKETTE The diskette is a circular sheet of plastic, coated with a magnetic coating similar to recording tape used on your cassette tape recorder. Diskettes, however, are manufactured to very exacting standards. The disk is sealed inside a plastic jacket with holes in the appropriate places so that the machine may 'read' the information recorded on the disk.

A standard TRS-80 35 track diskette is formatted as follows:

Tracks	35
Sectors per track	10
Sectors per diskette	350
Bytes per sector	256
Usable bytes per sector	255
Bytes per disk	89,600
Usable bytes per diskette	85,425
Usable sectors for data storage	335
Granules	70
Usable Granules (formatted disk)	67

The BOOT and DIRECTORY take 15 sectors of disk space. BOOT is physically located on TRACK 0 and occupies sectors zero through four. The DIRECTORY is on track 17 and occupies sectors zero through nine. (NOTE: All track and sector numbering begins with 0 (zero) as the first number.)

6.1 ORGANIZATION OF THE PHYSICAL RECORDS (SECTORS) When your computer reads a record into memory (even if it is a one-byte record) it actually reads an entire sector at a time. When that sector is updated and written back to the disk, the entire sector is written

back to the disk. In other words, no matter how big or small your record, a sector is read and written each time. For that reason, a sector (when referred to as part of a file) is called a PHYSICAL RECORD.

6.2 ORGANIZATION OF LOGICAL RECORDS When we create a RADEX-10 file, it may be as small as a single byte per record or as large as 255 bytes per record. These records are referred to as LOGICAL RECORDS in the Radio Shack TRS-DOS Disk Manual, as SUB-RECORDS. The number of logical records you may put into a sector will depend on your requirements as the RADEX-10 is created.

Each logical record, as it is stored on the disk, will adjoin the previous logical record without any delimiters or spaces between logical records. This is not true of ASCII files which require commas between records.

6.3 ORGANIZATION OF 'FIELDS' WITHIN LOGICAL RECORDS Fields within records also do not have delimiters or spaces between them. Each field adjoins the previous field. They are taken apart, so to speak, by the program and presented to you in a more readable form. Figure 6 is a 16 X 16 byte representation of a PHYSICAL RECORD with three LOGICAL records.

VII FILE STRUCTURES

7.0 THE FILE CONCEPT (Matrix) In order to visualize the records, we must first make it clear in our own mind how they are physically organized and how we should view them in a logical sense. The LOGICAL RECORDS in figure 6 are fielded and named as follows:

Field Number	Field Name	Size in bytes
1	NAME	20
2	ADDRESS	25
3	CITY	15
4	STATE	2
5	ZIP	4
6	H PHONE	8
7	W PHONE	8
8	AGE	2

For purposes of illustration, unprintable characters will be represented by a period (.), numeric unprintable characters by an asterisk (*), unused space by the plus sign (+) and the unusable byte by the pound sign (#).

```
THOMAS EDISON...
....555 INVENTOR
S.CIRCLE.DR..MEN
LO.PARK.....NJ**
****************
****NIKOLI.TESLA
.........1000.ZAP
.COIL.ST........
.NEW.YORK.......
NY**************
********ALBERT E
INSTEIN.....1,DI
SCOVERY.WAY.....
.....CAMDON.....
....NJ**********
************+++#
```

This is the record as it is physically organized on the disk.

(Figure 6)

As you can see in the above figure, there are no delimiters to tell you where each 'field' starts and ends. All this would be very confusing if we tried to visualize the records as they are physically stored on the disk.

For that reason, it is easier to think of the records as a matrix; figure 7 is such a matrix. In order to fit the matrix onto the page, we have had to shorten it, somewhat, from the above example.

RECORD	NAME	ADDRESS	CITY	STATE	H PHONE
1	EDISON	555 INVENTORS	MENLO	NJ	3127295506
2	TESLA	1000 ZAP	NEW YORK	NY	3125578977
3	EINSTEIN	1 DISCOVERY	CAMDON	NJ	3127784523

(Figure 7)

By thinking of the file organized in this manner, it is much easier to visualize. The records are horizontal and the fields are vertical --- in other words, a MATRIX.

7.1 THE ASCII FILE RADEX-10 is a random file structure; ASCII files are not. ASCII files require that the data be read sequentially until the file you want is found. Random files allow you to 'GET' or retrieve the file directly. For instance, if we wanted file number 500 in an ASCII file, we would have to start with file number 1, read that into memory, read file number 2 into memory and so on until we had done that 500 times. If each read takes 1/2 second it would require 250 seconds (4.166 minutes) to get file 500. With random access, that file would be up on your display in 1/2 second!

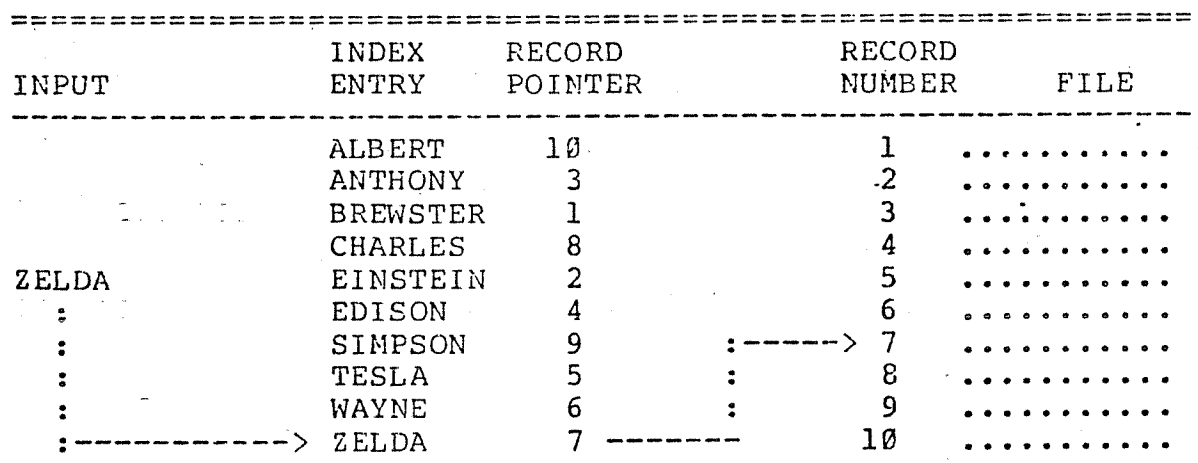
ASCII files do have some advantages for some applications requiring

short files, variable length records and simple structuring. A typical use for an ASCII file would be a temporary 'scratch' file or a small 'transaction' file.

VIII DATABASE STRUCTURES

8.0 There are an infinite number of ways to organize a file. You will, no doubt, find a few hundred ways yourself. Each 'FILE' you create will be a RADEX-10. How you structure that data base will depend upon your application and needs. No matter how that file was created, however, the program will read it and maintain it just as if it were especially written for that particular data file. Since we have already discussed the physical and logical structure in the previous sections, we will not review these aspects of the file again; instead, we'll discuss LINEAR and INDEXED structure.

8.1 RADEX-10, as you have it in this release, is LINEAR or record oriented; in other words, we must know which record number we want in order to retrieve a file. With an indexed structure, we can ask the program to match a data input with an index file that will 'point' to the record number. Figure 8 is a simple index scheme illustrating the method.



(Figure 8) In this example, the file we wanted was ZELDA; but we didn't know the record number for ZELDA. The program then searched the index until it found ZELDA; then, in that index, it found the RECORD NUMBER (pointer) for ZELDA, which was 7; and using that, got record number 7.

Indexes may be alpha, numeric or attributive. The index in figure 8 is an alpha index. A numeric index would use numbers placed into ascending or descending order. There are many types of index schemes possible.

An attributes index is one which indexes an attribute of the file. Suppose that we had an inventory file and in that file one of the fields contained the color of the inventory item. Further, let's imagine that the colors were limited to RED, YELLOW and BLUE. Figure 9 is one method of building an index of attributes.

	<-----INDEX------->			RECORD	
INPUT	RED	YELLOW	BLUE	NUMBER	FILE
	1	Ø	Ø	1	.....
	Ø	1	Ø	2	.....
	1	Ø	Ø	3	.....
	Ø	Ø	1	4	.....
	Ø	1	Ø	5	.....
	Ø	Ø	1	6	.....
	Ø	Ø	1	7	.....
	1	Ø	Ø	8	.....

(Figure 9) Attributes index

Here you can see that a "1" indicates that the file has the attribute of a particular color. Record 1 has the attribute 'RED', record 2 is 'YELLOW' and so on. This index could be a bit record index, since a byte has 8 bits consisting of ones or zeros. In other words, we can keep track of eight records, for one attribute, in one byte.

This type of index is called a BOOLEAN VECTOR index. RADEX-1Ø uses this type of index to keep track of 'DELETED' (inactive) records and 'OPEN' or 'KILL'ed records. Additional modules that may be added to RADEX-1Ø will allow you to access records by ALPHA, NUMERIC or ATTRIBUTE indexes. This module is scheduled for release sometime later in this year.

Composite indexes, made from two or more of the above examples, may also be used. In fact, an index can index another index; and on very large computers with large memory, there might be literally hundreds of indexes that 'point' to each other before one will finally point to a record or a file.

Of course, the examples we have given have been very simple in scope and application, and reading this will not tell you all there is to know about indexing. See SUGGESTED READING, Section XXIV, if you desire to know more about RADEX-1Ø structures and indexing schemes, especially James Martin's excellent book, COMPUTER DATABASE ORGANIZATION.

IX RADEX-1Ø PROGRAM ORGANIZATION

9.Ø PRIMARY PROGRAM MODULES The primary RADEX-1Ø program modules and their functions are:

1. RADEX ..... CONTROL PROGRAM
   (a) Initializes the system.
   (b) Sets the file specification to be used.
   (c) Displays the RADEX-10 menu.

2. WFL ..... FILE MAINTENANCE PROGRAM
   (a) Opens the specified file.
   (b) Displays the FILE MAINTENANCE menu.
   (c) Adds records to the end of the file.
   (d) Adds records to the first open spot.
   (e) Changes an entire record.
   (f) Updates any number of fields in a record.
   (g) Makes any record inactive (delete).
   (h) Makes inactive records OPEN records.
   (i) Performs all record keeping functions in the file parameters tables, ART and ORT sectors.
   (j) Manages disk spanning operations for adding and deleting records.
   (k) Closes specified record & returns to RADEX-10 menu upon completion of file maintenance operations.

3. CRATE ..... CREATES THE RADEX-10 FILE
   (a) Opens the specified file name on drive 1.
   (b) If no file has been specified or a different file name is requested; and a file is opened in that name on drive 1.
   (c) Displays the create instructions to the video.
   (d) Requests and sets file displacement.
   (e) Displays instructions and requests number of fields per record.
   (f) Displays instructions and requests 'HEADING NAME'.
   (g) Displays instruction and requests a definition of data type.
   (h) If data type is ALPHANUMERIC, requests length of field in bytes (1 byte per character).
   (i) Reiterates process until specified number of fields are defined.
   (j) Creates file with file parameters on drive 1.
   (k) Returns to RADEX-10 menu.

4. REPORT ..... CREATES REPORTS
   (a) Displays notice to remove write protect tab from program disk mounted in drive '0'.
   (b) Displays HEADING FORMAT menu and requests type of headings desired (horizontal, vertical or labels).
   (c) Requests the number of relational searches

you wish to perform.

(d) Displays record headers with field numbers; requests field to be searched.

(e) Displays menu of relational operators to be applied to search.

(f) Requests input for constant value to be applied to relational search. If input is defaulted, input will be requested when a report is output.

(g) Displays the RELATIONAL operators' menu and requests the relational operator to be used between each 2 searches; i.e., if 3 searches were made, 2 relational comparisons are made-
one between search 1 and 2 and one between search 2 and three.

(h) Reiterates process until the specified number of searches have been defined.

(i) Displays instructions and requests the number
of fields to be output.

(j) Displays instructions and requests user to specify which field is to be output.

(k) Displays instruction and requests length for output field. (Vertical and Label formatted reports do not request this information.)

(l) Displays minimum and maximum number of spaces that may be output for specified field if an incorrect response is made by user.

(m) Reiterates process until specified number of output fields are defined.

(n) LABELS FORMAT ONLY - Displays labels output parameters and creates labels output report, using first five fields of record without further user inputs.

(o) Creates and stores report output parameters in report output file on program disk mounted in drive zero.

(p) Closes all open files and returns to RADEX-10 menu.

5. OUTPUT ..... 'RUNS' REPORTS

(a) Opens specified data file and report file.

(b) Displays menu of user-created report file names.

(c) Requests report to be run.

(d) Requests user to specify output mode (Printer or video display).

(e) Requests user to specify forms length if out-put is to be to printer.

(f) Requests user to specify variables to be

used in relational searches if they were not specified when the report was generated.

(g) Outputs report. Program will not output deleted records even though a record actually exists in the file.

(h) Displays current record being processed to video display.

(i) Printer output prints header with report name, date and page number. Report is paged according to user input, specifying page length.

(j) Upon completion, program queries user if another report is to be run.

(k) User is returned to RADEX-10 menu if no more reports are to be run.

(l) Report output or menu may be terminated, with an up-arrow entry, at any time.

6. FILCHK ..... PRINTS FILE PARAMETERS

(a) Upon selection from main RADEX-10 menu, requests users' ready printer and prints all file parameters without further user inputs.

(b) File is closed and user is returned to RADEX-10 menu.

9.1 INTERLEAVED PROGRAM CONCEPT RADEX-10 is unique in several respects; one is the interleaved program concept. Each program line may be used in several program modules. If all of the database programs were 'MERGED' together, you would find that the sum of the whole is LESS than the sum of the parts.

This is due to "reusing" the code in several program modules. Each program function will "stand alone". This feature will play an important role when you wish to attach the APPLICATIONS MODULE to various parts of the program to use them in your own program.

X CREATING THE DATA BASE

10.0 Before we proceed with the actual creation, we must make sure that everything is ready to go. First, we will need a data disk to put the RADEX-10 file on. The following MUST be observed carefully:

1. Data disk must not have any locked-out tracks.
2. Data disk must not have any other files.*
3. Data disk must have 67 granules 'FREE' (see DOS manual, Section 4, Page 19).
4. Files accidentally created on the data disk MUST be 'KILL'ed (see DOS manual, Section 7, Page 28; and Section 4, Page 19).

* There is an exception to this rule. IF you are going to put up files of short length, and do not need to span disks, you may put as many

files on a disk as you can fit onto it.

10.1 Now that you have transferred your RADEX-10 programs to a disk with DOS and have also prepared the data disk, we are ready to set up the main program to run.

As of this writing, there are two Operating System choices: (1) TRS-DOS from Radio Shack, and (2) APPARAT ENHANCED DOS from International Jewelry Guild, Inc., or APPARAT of Denver, Colorado. There are three ways to invoke the RADEX-10 program.

1. USING TRS-DOS.
   (a) Load DOS. When 'DOS READY' prompt appears, enter BASIC and <ENTER>.
   (b) Respond to 'HOW MANY FILES?' with 6 and <ENTER>.
   (c) Respond to 'MEMORY SIZE?' with current memory size you normally set minus 255 and <ENTER>. If you do not set memory, you may use these values:

       32K ..... 48717 (Smaller value OK)
       48K .... 65486 (Smaller value OK)

       To make it easier to remember the value to set memory to, smaller values are OK ... for instance, instead of 48717 you could use 48500.
   (d) Type: RUN "RADEX-10" and <ENTER>.

2. USING TRS-DOS WITH AUTOMATIC BOOTSTRAP PROGRAM.*
   (a) 'COPY' DATABOOT/CMD from IJG disk to RADEX-10 program disk.
   (b) Invoke DOS. When prompt 'DOS READY' appears, type: AUTO DATABOOT and <ENTER>.
   (c) Press reset button.
   (d) Press reset to invoke program in all future operations.
3. USING APPARAT ENHANCED DOS.
   (a) Initiate DOS. When prompt 'DOS READY' appears, type:
       AUTO BASIC,6,<<insert memory size>>, RUN "DATABASE" and <ENTER>.
   (b) Press Reset button.
   (c) Press reset to invoke program in all future operations.

* DATABOOT is available from IJG, on disk for $12.95, including disk and one dollar for shipping. On the order blank included with this manual, specify the desired memory size you wish to reserve (also see Appendix A).

10.2 Now, you should have loaded the program, the notices will have been displayed and the RADEX-10 menu will be on the screen. For purposes of illustration, we will create a data base file, print the

parameters, do some file maintenance, create and run a report, delete some records and then clean up the files. By the time you are through with this, you will have a pretty good idea of how to use RADEX-10.

The Message you should have on your screen is:

```
============================================================
PLEASE READ TRS-DOS DISK OPERATING SYSTEM MANUAL, SECTION 3,
PAGES 6, 7, 8 AND 9  (FILE SPECIFICATION).
ENTER THE FILE NAME WITH AN EXTENSION (IF DESIRED) AND A
PASSWORD  (IF DESIRED).
REMEMBER, THE DATA DISK MUST BE IN DRIVE:1, AND IT SHOULD CONTAIN ONLY
ONE FILE.
```

TYPE OUT NAME WITH DESIRED OPTIONS

For this example, let's use the file name: DATATEST

Enter DATATEST (quotes are NOT necessary) and the RADEX-10 menu will be brought to the display. Select CREATE DATA BASE as your choice and hit <ENTER>.

The CREATE program will load and display its messages; and when you have responded to them, will display the CREATE menu and some instructions and comments. The file we are going to create will have the following inputs:

(1) Number of fields .....8.
(2) Headings, data types, length of each field.

FIELD #	HEADING	DATA TYPE	LENGTH
1	NAME	LSET ALPHA NUMERIC	25
2	ADDRESS	LSET ALPHA NUMERIC	25
3	CITY	LSET ALPHA NUMERIC	20
4	STATE	LSET ALPHA NUMERIC	2
5	ZIP	SINGLE PRECISION	*
6	H PHONE	DOUBLE PRECISION	*
7	W PHONE	DOUBLE PRECISION	*
8	AGE	INTEGER	*

* It is not necessary to know the length in bytes for these data types, as the program will automatically determine these parameters from your inputs.

*** WARNING ***
HEADING NAMES ARE LIMITED TO 12 CHARACTERS,
AND THE SUM <= 255 CHARACTERS.

Enter the first heading name and hit <ENTER>. The program will then ask you for the data type. LSET ALPHA NUMERIC is selection number 1. Enter '1' and hit <ENTER>. Now the program will prompt you to enter

the length in bytes. Refer to the above chart and enter the proper length for the heading.

*** CAUTION ***

AS YOU PROCEED THROUGH THE VARIOUS FUNCTIONS, PROMPTS AND ENTRIES, READ THE VIDEO DISPLAY. THIS WILL HELP YOU TO PREVENT ERRORS IN YOUR INPUTS.

Continue to enter heading and data types. You will notice, as you proceed, that the parameters of the DATABASE you are creating are displayed as you 'build' them. Each time you return to the heading input prompt, they are displayed. Although it is not a good idea from a procedures point of view, this will allow you to create a DATABASE as you 'shoot from the hip', so to speak.

Once you have entered all the heading names and have defined each data type, the program will create the file and the file overhead. It will then ask you if you wish to have the file parameters printed (or not printed, if that is your choice). In either case, you will be returned to the RADEX-10 menu.

Since it is so easy to create a RADEX-10 file, don't be satisfied with the first one you create. Study your file structure carefully for optimum utilization of disk space. If, by making your file a few bytes smaller, you could get one more record per sector, you will get a tremendous increase in the number of records per disk. This will become more important when you have to span disks to enter records.

10.2 Once the data disk has been created, it is a good idea to use the original as a master data disk. BACKUP or COPY the disk and put the original in some safe place. Later on, if anything happens to your data file, the original can be invaluable in recovering 'lost' data. How can you lose data? There are as many ways as there are people. If you ever have to use your master disk, it'll be there when you need it.

XI USING RADEX-10

11.0 Now that we have created a file, let's use it and become familiar with its operation. First, add 10 or so names to the file. RADEX-10s' menu should be up on your display, so enter a '2'; and the program will load the FILE MAINTENANCE module. As soon as the file maintenance menu is displayed, we are ready to go.

Select 'ADD A RECORD TO THE END OF THE FILE' as your first choice (1). Immediately, your field header will appear on the display with a graphic representation of the number of spaces you have allowed for this field.

Directly above your input prompt is a description of the data type, along with any pertinent information you might need for your data

entry. In the middle of the display is a number. As you begin to input information, this number will increment, showing you the number of characters that have been input. By pressing the <SHIFT> and "left-arrow" key, your entry will be deleted; and you may immediately re-enter data in the field. You may also backspace an entry.

When you have completed your entry, hit <ENTER> and the next field will appear with explanations of data type. Continue to make entries until no more prompts appear.

Upon completion of all entries, you will receive the following message on the display:

```
------------------------------------------------------------
RECORD ADDED AND IS # 1 ON THE FILE
HIT '↑' TO CONTINUE
------------------------------------------------------------
```

*** NOTICE ***

YOU MAY 'ABORT' THE DATA ENTRY AT ANY TIME BY PRESSING THE 'UP-ARROW' KEY. YOU WILL BE IMMEDIATELY RETURNED TO THE FILE MAINTENANCE MENU AND NO DATA WILL BE WRITTEN TO THE DISK.

11.1 Before we continue to add more records to the file, select number 2 (DISPLAY A RECORD TO THE VIDEO) and see how the program displays the records. With a file of this size, all of the fields will fit onto the screen with one display. There is a built-in 'page' so that, if your records are large, you will have to hit the '@' symbol to get the next 'page'. This will give you an opportunity to view the record without it rolling off the screen.

When you have added about 10 records to the file, we will see how a record is made 'INACTIVE'.

11.2 Select, as your next operation, number 6. MAKE A RECORD INACTIVE (DELETE RECORD). The prompt will be:

```
------------------------------------------------------------
ENTER THE RECORD NUMBER YOU WISH TO DELETE?
------------------------------------------------------------
```

Enter a '6' and you will get the message:

```
------------------------------------------------------------
RECORD # 6 IS DELETED
HIT '↑' TO CONTINUE
------------------------------------------------------------
```

After you have deleted a record and returned to the maintenance menu, select number 2 (ADD A RECORD TO THE FIRST OPEN SPOT).

The input is exactly the same as number 1. So, enter that record and

see what happens. If you had originally put 10 records on file, your OPEN SPOT entry should have been added as record 11. What happened? Didn't we 'DELETE' record number 6? Yes, but remember, we only made it INACTIVE; so, the first open record in this case was at the end of the file!

Now, let's see what happened to record number 6. Select number 3 from the menu (DISPLAY A RECORD TO THE VIDEO) and enter 6 as the record you wish to see. You will get this prompt:

```
------------------------------------------------------------
RECORD # 6 HAS BEEN DELETED
SHALL I PUT IT BACK ON THE FILE?
------------------------------------------------------------
```

Respond with a "Y" and you will get this message:

```
------------------------------------------------------------
RECORD # 6 IS BACK ON THE FILE, YOU MAY NOW USE IT FOR I/O
HIT 'A' TO CONTINUE
------------------------------------------------------------
```

Just so you may see for yourself that the record was left intact, display it to the video again. Make record 6 INACTIVE again and we'll play with it some more. Now that record 6 is inactive again, we'll CLEAN UP THE FILE.

11.3 Select number 7 (CLEAN-UP FILE (MAKE OPEN SPOTS OUT OF INACTIVE RECORDS)) and hit <ENTER>. The computer will, without further prompts or input, proceed on its own. When it has completed its task, you will get the file maintenance menu back.

Let's review what we have done to record 6, so far. (1) We put it on file, along with other records. (2) We made it INACTIVE. (3) We put a record out to the first open spot and the program put it up as record number 11. (4) We asked to see it, but since it was deleted, we got a message wanting to know if we wanted it put back on file. (5) We put it back on file. (6) We asked to see it again and, sure enough, it was still there. (7) We deleted it again. (8) Finally, we cleaned up the file. Now, record number 6 should be gone. Right? Wrong! It's still there.

Select number 3 from the menu and answer the prompt with 6 as the record number you would like to see. Again, you got the message that it had been deleted and would you like it put back on the file? Answer "Y" and then try 3 again. Yep, there it is.

OK, let's make 6 inactive again, and let's clean up the file again. After you have done this, select number 2 (ADD A RECORD TO THE FIRST OPEN SPOT) and input a record.

AH-HA!! This time, it put the record to open spot number 6! The

purpose of this exercise is to demonstrate the difference between an inactive record and an open spot. Since we don't have to really 'KILL' a record when we want it deleted, this gives you a chance to get it back if you change your mind. (Don't you wish you could do that to your programs sometimes?)

In a business application, this can become a valuable tool. Suppose you would like to make records inactive but can't use the record number again to reference customer data because it would mix-up the bookkeeping with duplicate numbers. This feature will allow you to 'age' a record; and then, after the appropriate time, reuse the number. This will also help you to keep your files as small as possible by reusing early record numbers.

11.4 So far, so good. The next project will be to UPDATE a record. Select UPDATE A RECORD (number 5 on the menu). The first prompt is:

```
------------------------------------------------------------
ENTER THE RECORD NUMBER TO BE UPDATED?
------------------------------------------------------------
```

Enter poor ol' number 6 again. First, the program will show you the entire record and then prompt:

```
------------------------------------------------------------
HOW MANY FIELDS DO YOU WANT TO UPDATE?
------------------------------------------------------------
```

If you are working with a large record, you will have to 'page' through it before you get the prompt. This is to give you the opportunity to check the data beforehand.

If you respond with a zero, the update will be cancelled; and you will be returned to the FILE MAINTENANCE menu. If you enter an up-arrow at any time during the data input, you will be returned to the FILE MAINTENANCE menu; and the entire update for that record will be cancelled. Try it a couple of times and become familiar with the procedure.

After you specify the number of fields you wish to update, you will be shown the entire record again; and the prompt will read:

```
------------------------------------------------------------
UPDATE # 1  ENTER THE FIELD # TO BE UPDATED ?
------------------------------------------------------------
```

Enter your update field number, and you will be prompted exactly in the same format as the ADD A RECORD TO THE END OF THE FILE routine. The process of showing you the entire record, after each field is updated, is continued until you have entered the specified number of updates.

If you find that you have specified more fields that you wish to update, you may respond to "FIELD TO BE UPDATED ?" with a zero; and you will proceed to the next update.

When the update is completed, the display will read:

```
------------------------------------------------------------
RECORD # 6 HAS BEEN UPDATED
HIT 'A' TO CONTINUE
------------------------------------------------------------
```

11.5 The next operation we will perform will be to CHANGE ENTIRE RECORD. You guessed it. We are going to change number 6! The CHANGE function appears exactly like ADD A RECORD TO THE END OF THE FILE. You may CHANGE any active record in the file. When a record is changed, ALL fields are changed. Upon completion, the display message will be:

```
------------------------------------------------------------
RECORD # 6 HAS BEEN CHANGED
HIT 'A' TO CONTINUE
------------------------------------------------------------
```

You will then be returned to the FILE MAINTENANCE menu. So far, there are only two things we haven't done on this menu. One is return to the RADEX-10 MENU, and the other is END THE PROGRAM. Both of these are self explanatory; however, END THE PROGRAM is more important than it appears. Read the following caution carefully.

*** *** *** *** *** *** *** ** CAUTION ** *** *** *** *** *** *** ***

ALWAYS END THE PROGRAM FROM THE MENU ENTRY. FAILURE TO DO SO CAN CAUSE PROGRAM MALFUNCTION AT A LATER TIME. ENDING THE PROGRAM UNDER PROGRAM CONTROL WILL ASSURE THAT ALL FILES HAVE BEEN CLOSED WITH THE PROPER UPDATED FILE PARAMETERS.

*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

11.6 Operations involving the spanning of disks are identical in every respect to those carried out on a single disk- with one exception --- the disk swapping involved in keeping the file parameters updated. When a file becomes too large for one disk, the program detects the requirement for an additional disk and issues the prompts and instructions to carry out the implementation of initializing the new disk.

*** CAUTION ***

DISK MAY NOT CONTAIN ANY OTHER DATA OR FILES. DISK MUST BE FORMATTED.

The prompting messages for disk spanning are as follows:

LOAD <file name> DISK NUMBER 2 IN DRIVE 1 AND HIT ENTER.

If the wrong disk is loaded, or the disk is not changed, the message will be repeated until the correct disk is loaded in drive one. If the operation being performed is a WRITE operation, you will be instructed to reload disk number 1 in drive one so the file parameters may be updated.

If a read operation is being performed, you will NOT be instructed to reload the disk number 1 until you request a record residing on that disk.

We are ready to return to the RADEX-10 menu. Select choice number 8, called amazingly enough, RETURN TO RADEX-10 MENU.

XII REPORT GENERATOR

12.0 An important part of any database is in transforming data into information. The REPORT GENERATOR is a tool for performing that task. Once again, the prompts are in English, as are the error messages. The report generator has many outstanding features. The following list will give you an overview of its capabilities:

1. Horizontal output to display or printer
2. Vertical output to display or printer
3. Label format output to printer
4. Free form formatting of output
   a. Any field used any number of times in a single output
   b. Any field may be output in any order
5. The record number is an added field for output and searches
6. Any relational operater may be applied to any field or number of fields
7. Every field is a key field
8. Perform any logical operation between any two searches
9. Search values may be constants or variables
10. Automatic form header with report name
11. Variable length forms paging
12. Inactive records not output as blanks
13. Up to 30 reports per data file
14. Any field searched any number of times up to 30 searches (TOTAL) per report.

12.1 Upon entering the report generator program, you will be cautioned to remove write protect tabs from the main program disk before proceeding.

*** CAUTION ***

"REPORT" MAY NOT BE RUN WITHOUT FIRST RUNNING THE 'RADEX-10' CONTROL PROGRAM TO INITIALIZE THE PROGRAM.

The only menu, as such, is the HEADINGS menu. (See figure 3). Select

the type of heading format for your report. For our first illustration, we'll select the horizontal format. Enter '1' and then the program will prompt with:

```
------------------------------------------------------------
ENTER THE NAME OF THIS REPORT (UP TO 8 CHARACS) ?
------------------------------------------------------------
```

The protocol and conventions are the same as for file names. Then you will be prompted with:

```
------------------------------------------------------------
ENTER THE NUMBER OF RELATIONAL SEARCHES YOU WISH TO PERFORM ?
------------------------------------------------------------
```

In this first example, we'll keep it simple. This report will be searched for record numbers greater than some yet unspecified number. We will also search the record for a zip code greater than some value and less than some value.

Now we know what we want to do. How do we do it? Actually, all you must do is follow instructions. The program will display the field names and field numbers. When you have decided which field will be searched, then enter the number of the field to be searched in search number 1. We want to search the record number because we may not always want to start with record one; we might want to start with the 100th record or the 526th record.

12.2 For our example, respond with field 'zero' (Enter: '0'). By now, you should have noticed that field 'zero' (the RECORD NUMBER) has been added to the fields we specified when we created the data base. The next prompt to appear will be the following:

```
------------------------------------------------------------
        1.  EQUAL TO  (=)
        2.  GREATER THAN  (>)
        3.  LESS THAN  (<)
        4.  SUB STRING-(INSTR)-FOR ALPHA FIELDS ONLY

ENTER THE TYPE OF RELATIONAL SEARCH FOR RECORD #
------------------------------------------------------------
```

Since we want to search for a record number greater than some number, type '2' and <ENTER>. At this point, we must decide whether or not we want this to be a constant search value or a variable search value. The prompt will now read:

```
------------------------------------------------------------
ENTER THE CONSTANT VALUE THAT YOU WISH TO SEARCH FOR.
(HIT 'RETURN' IF YOU WANT THE VALUE TO BE ENTERED AT REPORT TIME)
REMEMBER THIS IS AN INTEGER FIELD ?
------------------------------------------------------------
```

Here we have to make a decision. If we enter '0', the report will always start searching at record one. If we enter any number, as a matter of fact, the report will always start at the number we enter. If, however, we don't enter anything, just hit the 'ENTER' key; then we can enter whatever we want WHEN WE RUN THE REPORT!

This is a pretty powerful feature. It allows us to format a basic report and tailor it to suit our needs when we run it. We can specify some searches to be constants and others to be variables.

After this input, the computer will display all the fields again and will then ask which field is be be searched for search number 2. This time, specify the ZIP field for the search field. For the relational operator, specify GREATER THAN (>) and make it a variable input to be input at report time.

At this point, the program needs to know which LOGICAL operator to apply between search one and two. The prompt will be:

```
------------------------------------------------------------
        1.  AND
        2.  OR
        3.  AND NOT
        4.  OR NOT
ENTER THE TYPE OF LOGICAL OPERATOR BETWEEN SEARCH # 1 AND # 2 ?
------------------------------------------------------------
```

In this case, we want AND as the logical operator (record number greater than 'N' AND zip greater than 'NN').

For search number three, specify the ZIP field again; and this time, make the relational operator LESS THAN (<); and again, make it a variable to be input at report time.

Now we have to specify the LOGICAL operator between search number two and three. Again, specify 'AND'. The logic looks like this:

Record number > n AND zip > nn AND < nnn.

When we run the report, we'll input 'n', 'nn' and 'nnn'. Let's see how that will work. Suppose we had a file with 3000 records on it and that somewhere (we don't know exactly where) after about record 1000 we had put up all the entries for Pasadena. It would be a waste of time to start searching at record one, when we know we could start at record 1000 to find all the Pasadena entries.

*** CAUTION ***

LOGICAL OPERATORS AND RELATIONAL OPERATORS ARE EXECUTED IN THE ORDER THEY WERE SPECIFIED WHEN THE REPORT WAS CREATED. THEY DO NOT FOLLOW ANY HIERARCHY.

Further, we know (by consulting our Zip Code Directory) that all Pasadena zip codes start at 91101 and end at 91109. So, when we run the report, we can specify record number >1000: (AND) zip >91100 (AND) zip <91110. This will cause the program to list every person with a record number greater than 1000 with a Pasadena zip code. If we had chosen a LABEL format, we could have printed out our mailing labels, presorted for the post office.

12.3 Once the report parameters have been defined, we must define the output format for HORIZONTAL HEADINGS. Vertical and label formats do not require all the formatting specifications that Horizontal headings do. Since horizontal headings require the most information, we have chosen to demonstrate this format in the example. It is suggested that you create a report in each format to see the various possibilites with each type of format.

You will be shown all of the fields in the file; you will then be asked to:

ENTER THE NUMBER OF FIELDS YOU WISH TO OUTPUT ?

You must remember that you may have more fields than will fit across the page, so only output those that are necessary. For this reason, it is a good idea to have the file parameters printed out before you reach this stage; however, if you did not, you can still 'shoot from the hip'. If you specify more fields than will fit, the program will inform you of your error and will start you over.

12.4 Now that we have specified the number of fields we wish to output, we must specify which ones and in what order. First, you will be shown the file headings and asked which field you wish to output as field number 1. ANY FIELD MAY BE OUTPUT IN ANY ORDER IN A REPORT OUTPUT. You may make field #5 output, field 1 and field #0 output, field 2 and 7. Here again, you have the choice of formatting any way you wish.

If you don't know how many spaces each field needs for output, enter a zero; and you will get an error message telling you the minimum and maximum number of spaces the field will require. You may choose any number that you think will make your report easier to read or more attractive.

Once your report has been formatted, the report parameters will be written to the disk; and you'll be returned to the RADEX-10 menu.

XIII REPORT OUTPUT (RUNNING THE REPORT)

13.0 To run a report, select RUN REPORTS from the RADEX-10 menu. The report program will be loaded, and the REPORT menu will be

displayed with the names of the reports you have created. The following is an example of a typical report menu:

```
------------------------------------------------------------
                ENTER 'A' TO RETURN TO MENU
============================================================
        1.  DUMP
        2.  MAIL1
        3.  MAIL2
        4.  DISTSALE
WHICH REPORT WOULD YOU LIKE TO EXECUTE ?
------------------------------------------------------------
```

Enter the number of the report you want to run, and the program will ask you if you want the output to the printer or the display. If the output is to go to the printer, it will ask you to input the length of your form.

Next, you will input the variables that you specified when you created the report. You will then be prompted to ready the printer (if printer output was specified) and the report will be run.

You will notice that as the report runs, the current record number being processed is displayed on the lower portion of the display. When the program encounters INACTIVE records, this counter will increment, thereby giving you a clue that the program is doing something and has not 'hung' somewhere.

*** NOTICE ***

YOU MAY ABORT THE OUTPUT, WHILE THE REPORT IS RUNNING, BY PRESSING THE 'UP-ARROW'. YOU WILL BE RETURNED TO THE REPORT MENU.

When the report is complete (the last record read), you are returned to the REPORT menu. To return to the RADEX-10 menu, enter an "up-arrow".

XIV USING TABLES, FLAGS & OTHER TRICKS

14.0 (Reserved)

XV USER INPUT ERRORS

15.0 (Reserved)

XVI RESERVED VARIABLE NAMES & EXTENSIONS

16.0 As of this release (RADEX-10 Verson 1.1), variable names have NOT been compressed into one letter range. The principle difference between this release and 1.2 will be the compression of all variable names into variables beginning with "I".

16.1 With the release of version 1.2, a list of key variable names will be included in additional documentation which will accompany the release.

16.2 The reserved extensions to file names are:

1. /GEN ..... Report parameters file
2. /TAB ..... I/O tables parameters file
3. /IND ..... Index tables file
4. /TRD ..... Reserved

These extensions to file names may NOT be used by the user. The program will generate files with these extensions and will place the files onto the proper disk.

XVII FUTURE PROGRAM MODULES

17.0 RADEX-10 is designed to be used in many applications. Some of these applications will require additional features not included in the primary RADEX-10 software package. In addition, other modules will be released that will expand the capabilities of RADEX-10 or provide additional functions.

Future module releases will include:

1. I/O TABLES ..... Allows the user to create tables of input data that are stored as one byte fields. An example would be a list or 'table' of city names. With I/O tables, 'LOS ANGELES' would be stored in one byte.

2. APPLICATIONS INTERFACE ... Allows the user to include RADEX-10 or portions of RADEX-10 in an applications program.

3. FILE LINKER ..... Will allow the 'Linking' of different files for report output and the arithmatic manipulation of RADEX-10 record fields for output as pseudo fields for reports.

4. SUPER SEARCH ..... Will find any numeric or alpha string anywhere on the file, regardless of data type. It will require one input and will output to the display or printer.

5. CREATE READ/WRITE INDEXES...Creates alpha, numeric and attributes indexes. Requires a minimum 3 drive system. Will allow the use of 2 drives for data storage; will store and retreive records using index entries.

6. SORTS & MERGES - FILE CONVERSIONS ... Will sort and merge any file structure created by ANY data base, whether ASCII or RANDOM files. Will convert ASCII files to Random files or Random to ASCII. Will delete or add fields to existing records. This is a machine language

program and may be used as a stand-alone program; it does not require the use of RADEX-10 but may be used with RADEX-10.

7. CHANGE REPORTS ..... Allows user to 'KILL' a previously defined report or change a report.

8. RPG REPORT EMULATOR ..... Simulates RPG (Report Program Generator) language statements for formatting of reports. Will allow the user to create pseudo-fields for output, using mathematical calculations on actual stored data. Will also allow for cumulative totals, page totals, sub-totals, line and level totals. (May be used with FILE LINKER.)

9. RADEX-10 FILE CONVERSIONS ... Will add or delete fields to RADEX-10 records and will change file parameters. Requires a minimum 3 disk drive configuration

10. LABELS OUTPUT ..... Will output any field, in any order, for any label format. Used for special labels, parts labels, mailing labels, inventory labels, etc. The program will request the labels layout and will output to the printer any desired label layout.

11. ELECTRIC PENCIL LINKER .... This program will take files created by the ever-popular ELECTRIC PENCIL by Michael Schrayer and will output the file just the same as you would in PENCIL; EXCEPT, you may access RADEX-10 to insert names and addresses. You may also insert other data at selected places in your PENCIL output.

XVIII TYPICAL APPLICATIONS

(Reserved)

XIX PROGRAM OPERATION

(Reserved)

XX LIMITATIONS

20.0 The following is a list of limitations encountered in using RADEX-10:

(a) Total number of records per file. The ART and ORT sectors index up to 10,199 records. This is the maximum number of records you may have on a RADEX-10 file.

(b) There are 67 granules of available space on a formatted diskette. Each granule consists of 5 sectors. To determine the number of records per disk, use the following formulas:

20.1 DISK NUMBER 1:
67 grans X 5 = 335 - displacement = available sectors for records.
Available sectors X number of records per sector = records on disk 1.
DISK NUMBER 2 TO N:
67 X 5 = 335 sectors -1 = available sectors for records (there is one sector of overhead on each data disk after disk 1.)

20.2 NUMBER OF DISKS:
Since the maximum record size is 1 sector (255 bytes), and the maximum number of records is 10,199 then, allowing for a minimum displacement of 16 sectors on disk 1, a total of 31 diskettes is the maximum number that a file may occupy.

20.3 SIZE OF FIELD NAMES:
Field names may not be longer than 12 characters.
Total field name lengths may not exceed 255 characters in any one file.

20.4 MINIMUM RECORD SIZE:
2 bytes.

20.5 MAXIMUM RECORD SIZE:
255 bytes.

20.6 MINIMUM NUMBER OF FIELDS PER RECORD:
2.

20.7 MAXIMUM NUMBER OF FIELDS PER RECORD:
127.

20.8 MAXIMUM NUMBER OF SEARCHES IN A REPORT:
May not exceed 31 searches in a single report.

20.9 MINIMUM NUMBER OF SEARCHES IN A REPORT:
1.

20.10 MAXIMUM NUMBER OF FIELDS IN A REPORT OUTPUT:
Any number up to the number of fields, plus the record number, AND may not exceed 132 spaces in a horizontal report. Vertical reports may output up to 132 spaces per field, including field name. Labels formats are restricted to the first 5 fields of the record and may not exceed 35 characters on any one output field NOT including field name.

*** CAUTION ***
LABEL OUTPUT WILL "HANG" IF ANY FIELD EXCEEDS 35 CHARACTERS ON LABELS OUTPUT.

XXI (RESERVED)

XXII (RESERVED)

XXIII OPERATING SYSTEM LIMITATIONS

23.Ø TRS-DOS Version 2.1 is, as of this writing (4/14/79), not a fully debugged operating system. All rumors not withstanding, it appears that Radio Shack will not release a version with corrections to 2.1 for some time to come. As a result of the many errors generated by 2.1, International Jewelry Guild recommends the use of the APPARAT enhanced DOS. Some of the known errors of TRS-DOS are:

(1) 'LOC' command in DOS-BASIC fails due to erroneous call.

(2) 'APPEND' command does not work.

(3) 'VERIFY (ON)' turns off the verify and 'VERIFY (OFF)' turns on the verify function.

(4) 'ATTRIB' uses the wrong error code.

(5) 'CLOSE' causes a major system disaster when it releases an FXDE by not preserving the contents of register DE, containing count of +1 sectors yet to be freed, when freeing a no longer-needed byte. This error is compounded by the branch at 4ED9 not implicitly ending deallocation when the file is known to have no more granules assigned.

(6) The above errors cause all writable main memory from 3ØØØ - 42XX to be set = 'FF', where XX is the relative position within the sector of the last byte of the FPDE pointed to by the last FXDE released.

The corresponding sector in the directory is also filled with 'FF' to that relative point.

(7) As a result of (5) and (6), the GAT directory sector is modified to free-up granules at random in tracks ØØ-FF, with most tracks below 8ØH. This will cause granules, previously allocated to other files, to be allocated again in subsequent file allocations. This includes reallocation of BOOT/SYS and DIR/SYS granules, eventually destroying those files and resulting in a "NO SYSTEM" error message and rendering the data file unrecoverable by less than heroic methods.

(8) In addition, files whose FPDE preceded the destroyed FPDE in the directory entry sector will disappear from the system. If a file's FXDE was so destroyed, the user would have extreme trouble from that file and should be considered lucky if the system detects an