DOCUMENT RESUME

ED 094 779 IR 000 915

| | |
|---|---|
| AUTHOR | Cohen, Malcolm S. |
| TITLE | On the Feasibility of a Labor Market Information System. Volume 3. Final Report for Period July 1, 1970-June 30, 1974. |
| INSTITUTION | Michigan Univ., Ann Arbor. Inst. of Labor and Industrial Relations. |
| SPONS AGENCY | Manpower Administration (DOL), Washington, D.C. Office of Research and Development. |
| REPORT NO | DLMA-71-24-70-02-3 |
| PUB DATE | Jun 74 |
| NOTE | 177p.; For related documents see IR 000 913 and 914 |
| AVAILABLE FROM | Institute of Labor and Industrial Relations, Publications Office, 401 South Fourth Street, Ann Arbor, Michigan 48106 ($3.00) |
| EDRS PRICE | MF-$0.75 HC-$9.00 PLUS POSTAGE |
| DESCRIPTORS | *Computer Oriented Programs; *Feasibility Studies; *Information Retrieval; *Information Services; Information Theory; *Manpower Utilization; Programing Languages |
| IDENTIFIERS | Benchmarking; *Labor Market Information System |

ABSTRACT

Volume 3 contains supporting appendixes of a three-volume study designed to evaluate the need for new labor market information. The study also sought to improve methods for making better use of currently collected statistical and administrative data. The appendixes support Volume 1's chapter 2 "A Flexible Data Retrieval System" and chapter 3 "Data Access and Quality." Volume 3 appendixes are: (1) improvements in benchmarking, (2) automated graphics for benchmarking, (3) a manpower information service, (4) a set theoretic data structure and retrieval language, (5) MICRO information retrieval system technical reference manual, (6) the LMIS (Labor Market Information Service, (7) the needs of users of a labor market information system, and (8) minority employment data sources. (WCM)

Report DLMA 71-24-70-02-3



# ON THE FEASIBILITY OF A LABOR MARKET INFORMATION SYSTEM

## Volume III

Malcolm S. Cohen

Institute of Labor and Industrial Relations
The University of Michigan-Wayne State University
Ann Arbor, Michigan 48104

June 1974

Final Report for Period July 1, 1970-June 30, 1974

Prepared for

U.S. Department of Labor
Manpower Administration
Office of Research and Development
Washington, D.C. 20210



This report was prepared for the Manpower Administration, U.S. Department of Labor, under research and development contract No. DL 71-24-70-02. Since contractors conducting research and development projects under Government sponsorship are encouraged to express their own judgement freely, this report does not necessarily represent the official policy of the Department of Labor. The contractor is solely responsible for the contents of this report.

ADDITIONAL COPIES MAY BE OBTAINED FROM

Institute of Labor and Industrial Relations
Publications Office
401 S. 4th Street
Ann Arbor, Michigan 48106

This report is available in three volumes. The price is $3.00 per volume.

Order from: The Institute of Labor and Industrial Relations
Publications Office
P.O. Box B-1
University of Michigan
Ann Arbor, Michigan 48106

TABLE OF CONTENTS

Volume III

TABLE OF CONTENTS (Continued)

Volume III

TABLE OF CONTENTS (Continued)

Volume III

TABLE OF CONTENTS (Continued)

Volume III

# LIST OF FIGURES

## Volume III

APPENDIX H

# A SET THEORETIC DATA STRUCTURE AND RETRIEVAL LANGUAGE

William R. Hershey and Carol H. Easthope*

## 4.0 Introduction

The development of a data structure for use with labor market information has led to a system which is capable of handling many types of data bases. The Labor Market Information System (LMIS) Project is being funded by the U.S. Manpower Administration to study the feasibility of implementing a nation-wide information system for the storage and retrieval of labor market data and to build a labor market model. The characteristics of a good labor market information system are not unlike the characteristics of data bases in many other applications. Therefore many of the features discussed here should be of widespread interest.

The goal of an accurate model of the labor market almost demands an efficient and accurate information system to supply the model's input data. A second purpose for developing an information system is to provide an extremely simple retrieval system to be used by the state Employment Services for the access of data that is not available through their "canned" report generating programs.[1]

Two of the primary criteria that our information system must satisfy are generality and compatibility to handle existing data files for different surveys. An interactive system is highly desirable, since it allows our economists to ask questions of the data bases quickly, and since we can support and maintain our system on one computer which can be called easily by the states that are participating in our project. The Michigan Terminal System (MTS) on which the program is run is also a very good interactive environment.

---

* The Labor Market Information Systems Project is sponsored by the Manpower Administration, Office of Research and Development, United States Department of Labor, under contract no. 71-24-70-02. The views represented in this paper are the sole responsibility of the authors and do not necessarily reflect the views of the Department of Labor.

1 We are presently supporting our systems for three states: Michigan, Wisconsin and Colorado.

The uses to be made of our retrieval program require the utmost simplicity at the user level. Hence, the instructions are entered as English sentences rather than in fixed formats. There is an option to correct spelling mistakes, and the program is self-documenting via commands from the user.

## 4.1 General Organization and Function of the Program

The traditional way to implement a retrieval system has been to determine what questions are to be asked of the data and then to write a program and data structure to answer those questions. Thereafter one is locked into that class of questions. However, in our applications, and indeed in many other applications, it is not known what questions will be posed. Nor is the data in any particular format. We had to look into new ways of designing our system, so that any question could be posed with any data.

We have available on the University of Michigan computer a generalized retrieval program called Set Theoretic Data Structure (STDS), developed by Set-Theoretic Information Systems Corp., Ann Arbor, Michigan. This program is composed of a number of very efficient routines which treat the data bases as sets and perform set operations on them, e.g., union, intersection, restriction, etc. STIS Corporation holds the view that there exists an information enviornment to which questions can be directed, a machine environment in which the data resides, and that "Any data structure is actually an isomorphism between a machine environment and an information environment preserving the functional aspects of each" [3]. They feel, however, that the usual data structures do not preserve the functional differences; the information environment is made to look like the machine environment.

The problem is to find a data structure that can map the myriad relationships of the information environment into the algorithmic, procedural world of the machine. Such a structure could be a set-theoretical model. STIS has extensively investigated the proposition the general information requests can be abstracted to set operations. For example, if an information request is expressed as a set operation, $\circledast$ , given data sets A and B, with the retrieved result as set C, then

the abstraction can be stated:

$$A \circledast B = C$$

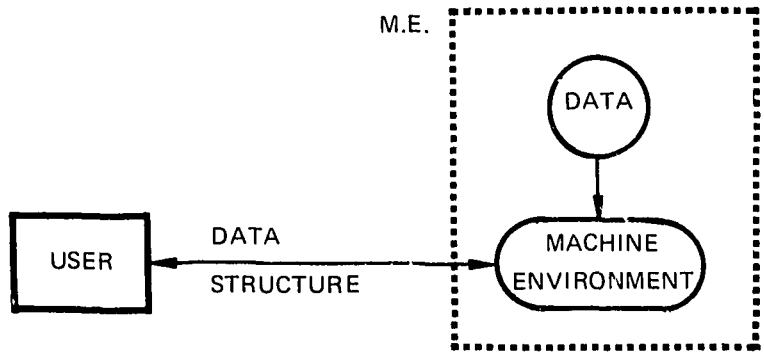
To define this abstraction and thereby prove that $\circledast$ is a valid set-theoretic operation, it is further stated that an element x is a member of C if and only if there is a truth function relating A, B, and x, i.e.,

$$C = (x \mid \psi(A,B,x))$$

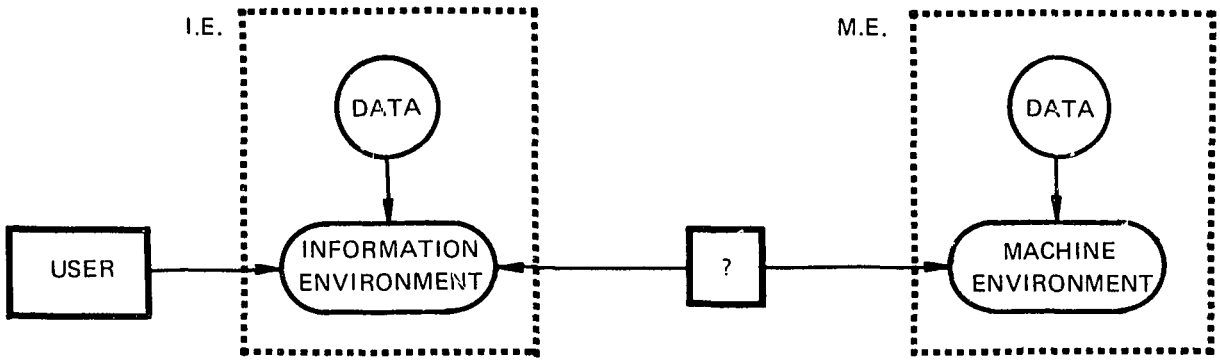
It is the fact that the function is decidable that makes $\circledast$ a set theoretic operation and assures that C is defined. Any algorithm or procedure that decides $\psi$ is valid. Therefore any convenient and/or economic machine representation of data can be mapped into the set-theoretic information environment. (See Figure 7) hence the STDS routines are essentially machine independent, although the initial versions require the paging facility for virtual memory with the MTS operating system on the IBM 360/67.

Our retrieval system, which we have named MICRO, calls on the STDS routines to do set-theoretic retrieval operations on the data. Essentially then, our program is an interface between the user and the STDS routines which do tha actual manipulation of the data (see Figure 8). This interface includes all the facilities needed to make data retrieval a simple task for users who are totally unfamiliar with computers. There is a syntax analyzer to parse the input commands, a dictionary to associate fields and records in the data with words that the user understands, 1/O and file manipulating routines that prepare the data in the form needed by STDS, error diagnostics and even automatic error correction, self-documentation to aid the confused or forgetful user, and facilities for preparing the output for the terminal, for peripheral storage, or in a format suitable for input to statistical routines. It is with this MICRO interface program and its affiliated data structures that this paper concerns itself; literature describing the STDS routines is available from STIS Corp. [14]. (See also [4] and [5].) In our experience thus far it has been shown that the user interface is an extremely important part of the total retrieval package, since with a more difficult program the unsophisticated user would be helpless and not likely to utilize the data most effectively.

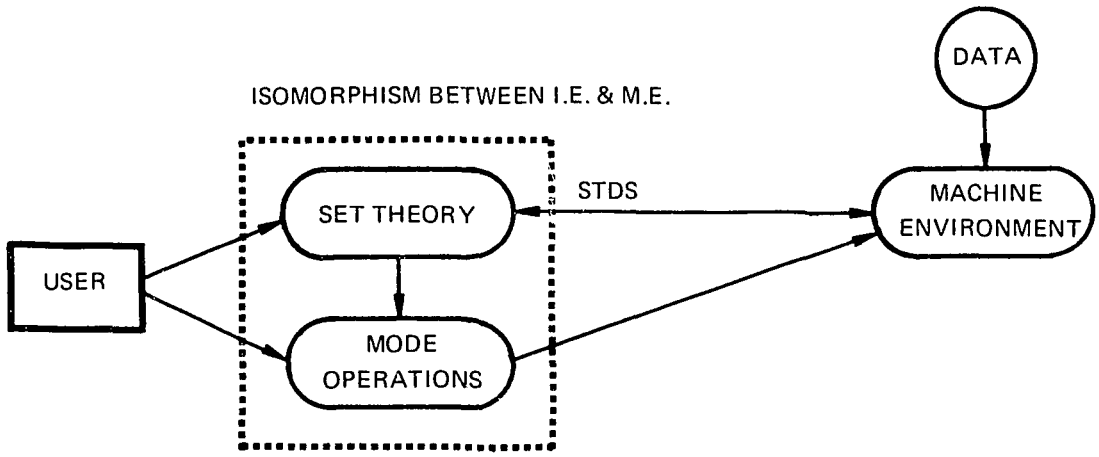
## Figure 7 Contrasting Data Structures




USUAL DATA STRUCTURE - User can only make retrieval requests as structured by the machine environment.

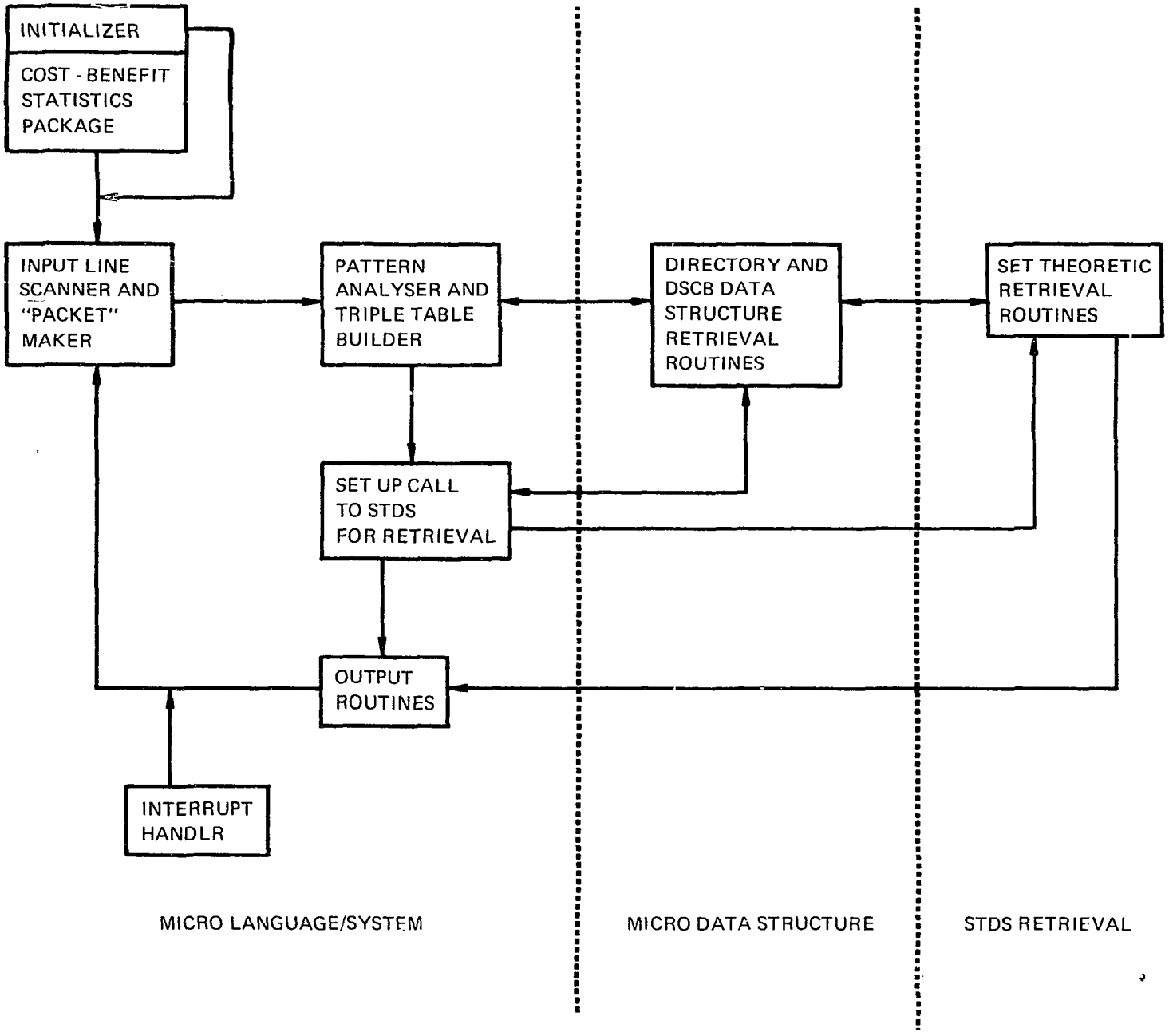



IDEAL DATA STRUCTURE - User can make any request, limited only by and inherent limitations in the relationships within the data itself. There is the need for an isomorphism between I.E. and M.E.




SET THEORETIC DATA STRUCTURE - User expresses request as set operations, which give him full retrieval power. The STDS and mode operations isolate the user from M.E.

Figure 8 MICRO Interface To Set Theoretic Operations



## 4.2 Structure and Content of the Data Bases

The STDS routines work equally efficiently on any data. Therefore, retrieval can be done in any mode. But since our data bases tend to be large (up to one million bytes), we convert data from characters to binary numerical form before using it with the program. While the cost of the one-time conversion is sometimes high, the conversion effects considerable savings in virtual memory and execution time charges for the retrieval program.

The MICRO representation of the data can be visualized as a matrix in which the rows represent different records or "cases", and the columns are fields in which are recorded characteristics for each record; however, the actual data representation is in set-theoretic form. In labor market applications the rows typically represent job applicants or employees and columns (called Fields) designate fields such as age, sex, and income or type of industry, number of employees, and payroll. Each column, i.e., Field, can have certain "Categories" which stand for different values of the specified Field. For example, the Field SEX would have the categories MALE and FEMALE.

The Categories are coded in the data structure as numerical values, and they are translated to words that the user understands at retrieval time via a "dictionary". Field names are also included in the dictionary, since STDS does not currently keep track of these names. The retrieval is one by using the byte position and field length of the Field in the record; hence each Field name in the dictionary has associated with it the length of the Field (one to four bytes) and the Field's byte position within the record.

Some of the information in a data-base may be meaningful as numerical information, i.e., not translated into a Category word. Examples would be age or income. In these cases no Category recodings are provided, and MICRO converts the binary number to decimal for printout when necessary.

## 4.3 Use of the Program and Data-Bases

The set-theoretic approach allows great flexibility in the retrieval operations. Four major retrieval operations will be discussed briefly

here. The MICRO command keywords for these operations are FIND, XTAB, SELECT, and RESTRICT. All of these operations result in the formation of a RESULT set, i.e., a subset of the original set. This RESULT set can be renamed for further manipulation or storage by use of the NAME command.

The FIND command extracts a subset of the data-base by matching specified Category information under a specified Field or Fields. Hence the resulting subset consists of selected rows of the data matrix. Logical combinations of Fields are possible, as shown in the following examples with a data-set called "SOC-SEC".

FIND IN SOC-SEC WHERE SEX IS FEMALE.

FIND IN SOC-SEC WHERE SEX IS MALE AND RACE IS WHITE.

FIND IN SOC-SEC WHERE AGE IS BETWEEN 20 AND 30.

FIND IN SOC-SEC WHERE OCCUPATION IS CARPENTER OR PAINTER OR PLASTERER.

After finding the RESULT subset, MICRO stores the subset temporarily and prints out a count of the selected cases, i.e., the number of records whoe Category values under the specified Fields conform to the logical combination designated in the FIND command. The RESULT subset has the same record length as the original set, i.e., all Fields in the original set are included.

The XTAB command can do a cross tabulation of frequence of co-occurrence of codes under specified Fields. The number of Fields XTABed can range from two to six, and with this command the entire RESULT set is printed out. The following example will illustrate the command and a typical resulting printout.

XTAB IN SOC-SEC SEX BY RACE.

Printout:

| SEX | RACE | COUNT |
|---|---|---|
| MALE | WHITE | 25,235 |
| MALE | NEGRO | 5,243 |
| MALE | OTHER | 451 |
| FEMALE | WHITE | 27,457 |
| FEMALE | NEGRO | 6,105 |
| FEMALE | OTHER | 347 |

The SELECT command selects a subset by picking out specified Fields in all records. In other words, the subset consists of designated columns of the data matrix. Note that the logical OR operation is illegal in SELECTing Headers.

SELECT IN SOC-SEC SEX AND RACE AND OCCUPATION.

SELECT IN EMPLOYERS PAYROLL AND INDUSTRY.

The RESTRICT command is usually used to match records between two different data-sets. Both data-sets will have Fields with ID numbers that match for corresponding records. The program finds all records in a data-set whose values under a specified Field match any of the values under a specified Field in another data-set. For example,

RESTRICT IN SOC-SEC WHERE IDNO IS ID IN JOB-APPS.

In this example SOC-SEC is a data-set containing records of all individuals residing in a given area, and IDNO is a Field name for a unique identification number for each individual. JOB-APPS is a mythical file of job applicants at a State Employment Service, and ID is the Field for the unique identification number for each applicant. With the command shown one could find records in the SOC-SEC data-set for only those individuals who are in the JOB-APPS data-set. The cardinality of the RESULT set is printed out, i.e., the number of records in SOC-SEC whose values for IDNO match values of ID in the data-set of JOB-APPS. The length of records in the RESULT set will be the same as the records in SOC-SEC, the RESTRICTed data-set.

Other MICRO commands allow the user to SAVE a RESULT set permanently for future reference, to PRINT all or part of a permanent or temporary data-set, to DESTROY data-sets that will no longer be used, to DELETE data-sets temporarily to save on the cost of core storage during operations on other data-sets, and to ask for documentation about the contents of data-sets as well as the MICRO commands themselves.

## 4.4 Program Control and Use of the Data Structure

Having looked at MICRO's general structure and operation, we can proceed to study the data structure itself in more detail. The data structure is divided into three parts: the directory, the dictionary, and the data itself. We will treat each separately. All three components are stored on disk all the time, and they are brought into core

separately when needed. There is no difference between the representation on disk versus that in core, except that the disk addresses (note pointers) are replaced by core addresses when the information is read into core.[2]

## 4.5 Directory

The directory is simply a list of available MICRO data-sets. It contains the names of data-sets and the location of their dictionaries (see Figures 9 and 10). It also informs MICRO whether the data-set can be destroyed. Typically, several individuals have access to the same data-sets, and not to those of another group. There is also a master directory which lists datasets that anyone may use. These master data-sets either are for demonstration purposes, or contain non-confidential information that is to be shared among different user groups.

When MICRO begins execution, it reads the user's directory into core and then the master directory. It combines the two into a simple forward ring (see Figure 11), creating a "universe" to which the user has access. Whenever a temporary data-set is created with MICRO, a directory entry for it is inserted as the second element of the ring. The premise is that a temporary data-set is more likely to be referenced again and therefore should be near the beginning of the ring.

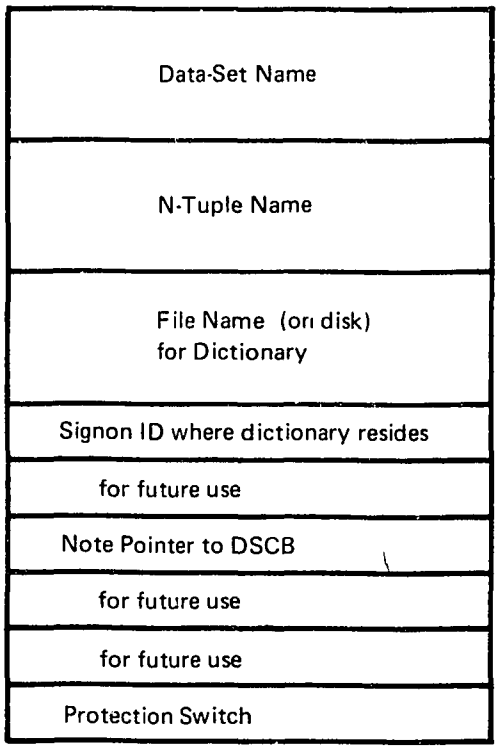
A directory entry for a temporary data-set is not written to the disk directory list unless a SAVE command is issued. In that case it is simply written as the last record of the user's disk directory. Therefore, there is no correspondence in order between a disk directory and the in-core directory.

When a temporary data-set is DELETEd, the entry is removed from the ring. (A permanent data-set only has its data deleted from core, not its directory entry.) Hence there are never any directory entries for temporary data-sets on disk. However, when a permanent data-set is

---

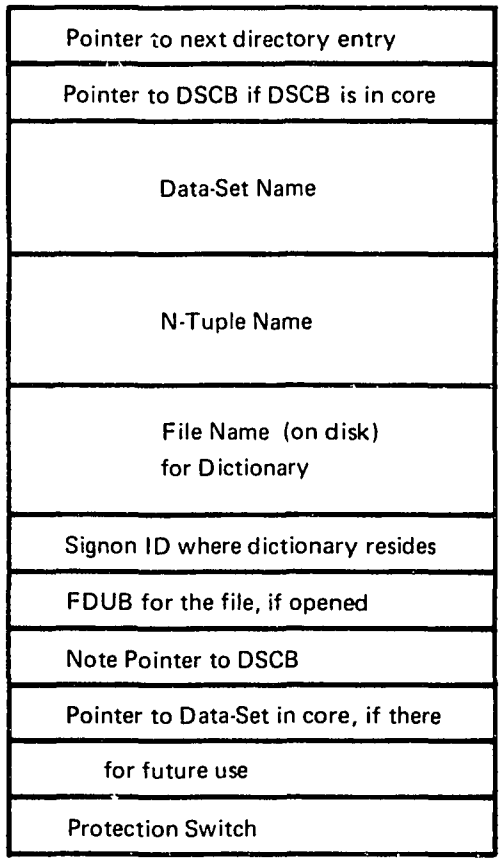
[2] By core we mean really virtual memory. Our computing system has a paging facility to continuously copy core to drum and vice versa according to user demand for core. The combination of core and drum storage is called virtual memory.

Figure 9 Directory Entry Format On Disk



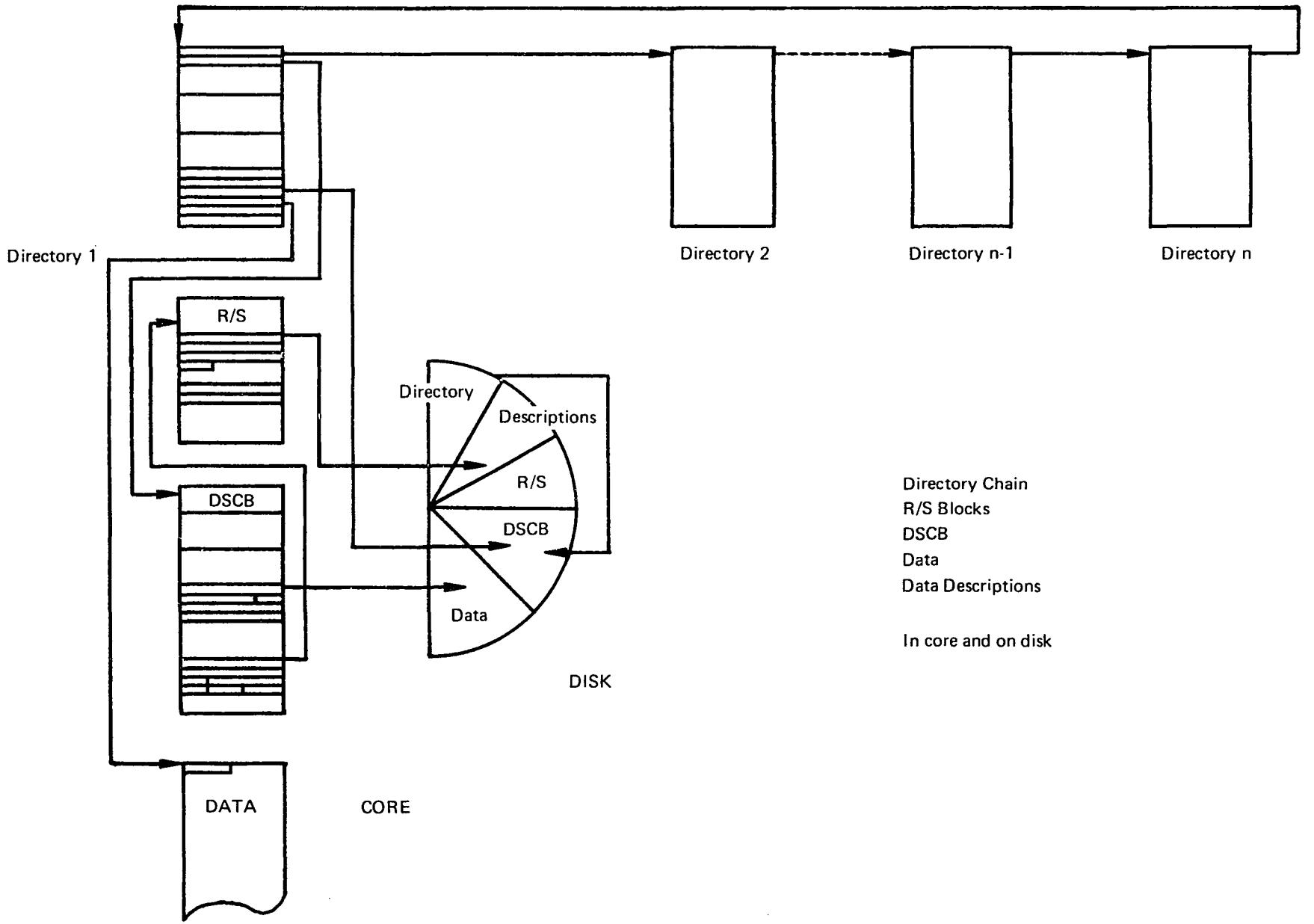

(One entry for each data-set)

Figure 10 Directory Format In Core




(One area for each data-set)

Figure 11 MICRO Combines Dictionary And Directory



DESTROYed (a DESTROY of a temporary defaults to a DELETE), the disk directory entry is removed, along with the in-core directory area, the dictionary, and the data.

## 4.6 The Dictionary

The dictionary itself has three parts: the data set control block (DSCB), the R Blocks and S Blocks, and the data descriptions (English language descriptions of the data-set and its Fields and Categories). (See Figures 12, 13 and 14.) There are only slight restrictions on the order of the dictionary components. They are usually arranged with all data descriptions first, followed by R and S Blocks, and then the DSCB and tape mounting information (if the data is on tape rather than disk). The directory entry for a data-set dictionary contains a note pointer (disk address) to the location of the DSCB record in the file where the dictionary resides. The DSCB in turn has note pointers to the R/S Blocks, and R/S Blocks have note pointers to the data descriptions.

MICRO reads in a DSCB and it associated R/S Blocks on demand, either explicitly when a USE or ACQUIRE command is issued, or implicitly when a retrieval command such as FIND is issued. Once in core, the DSCB remains there for the rest of the session, unless a DESTROY or SAVE command is issued.[3] The directory area for the data-set has a pointer which is set to point to the DSCB in core (see Figure 11). The note pointers in the DSCB are also changed to core addresses for its R/S Blocks. However, the data descriptions are never in core permanently. The note pointers in the R/S Blocks remain as disk addresses all the time. MICRO, in response to a DESCRIBE... Command, does a direct-access read from disk with the appropriate note pointer.

When the structure and format of the directory and dictionary were first designed, it was felt that room should be left for after-thoughts and future expansion. That is the reason for the abundance of "spare"

---

[3] The DESTROY command releases core for the DSCB and data-set, as well as destroying the dictionary and data-set files. The SAVE command, while creating dictionary and data-set files, releases the current core for DSCB, R/S Blocks, and data-set. The SAVEd data-set's DSCB is read in again on demand if it is again referenced.

Figure 12 Data Set Control Block (DSCB)

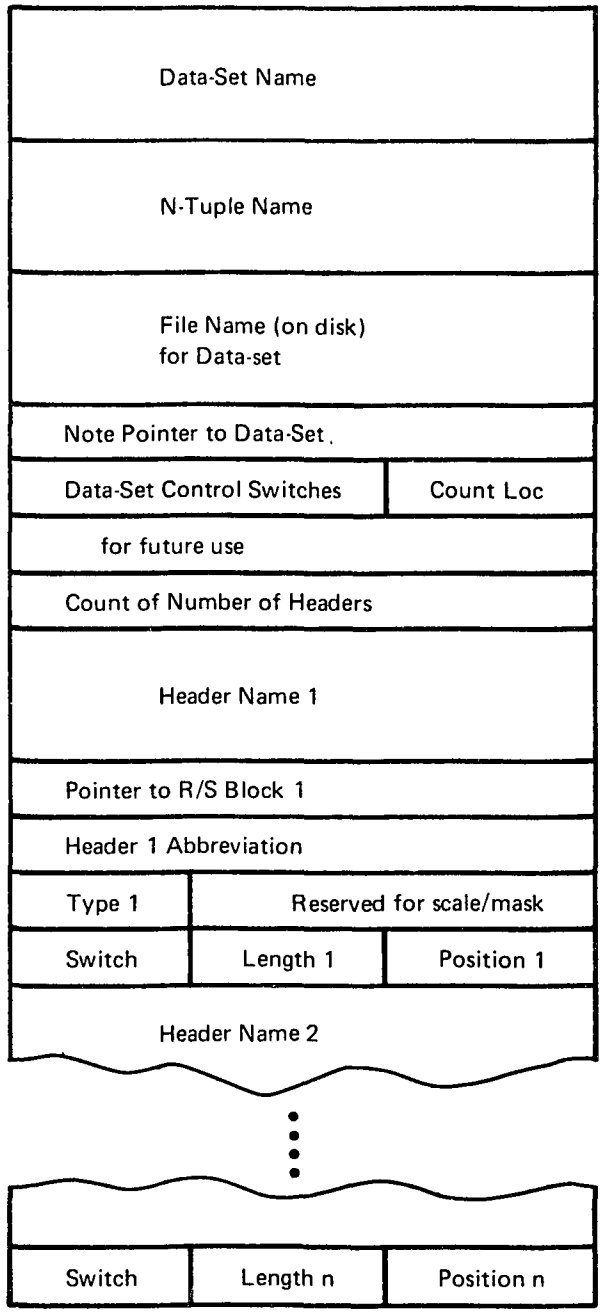
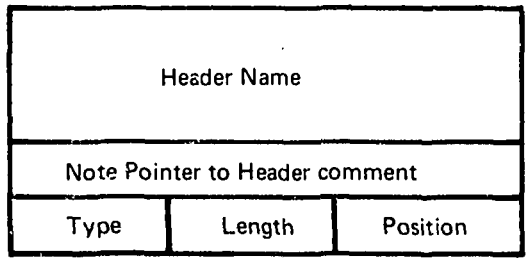
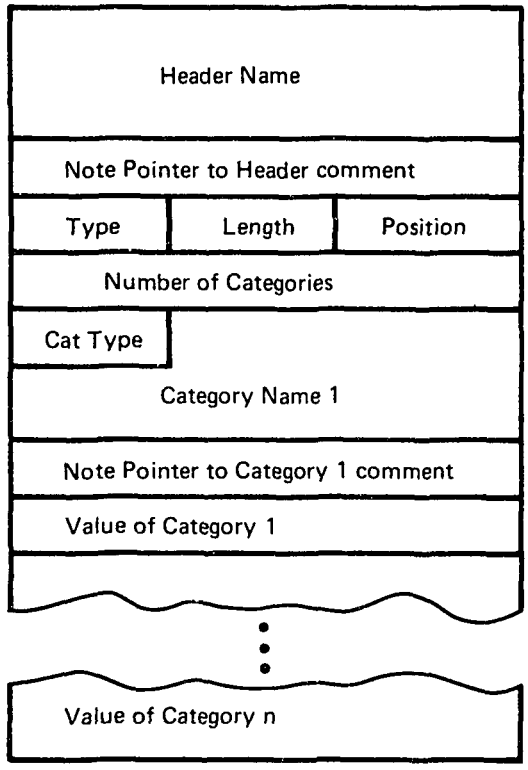


**Figure 13 The R Block**




(For Headers with numerical information only)

**Figure 14 The S Block**




(For Headers with coded Category information)

bits and extra spaces in the various control blocks. In fact, since the first demonstration version (late March, 1971), about one third of the original "spare parts" have been utilized. At the same time, data-sets created at the beginning of the project are still compatible with new and improved versions of MICRO.

## 4.7 The Data-Set

The structure of the MICRO is extremely simple. It can exist on disk or tape. The first record is a specially encoded fullword that specifies the data-set's n-tupple length (record size) and its cardinality (number of records). The data immediately follows in 32K-byte records (rows of the data matrix are stored contiguously).

A data-set record can be a mixture of binary and character fields, where the field length can be from one to four bytes. The field length is a limitation of the current STDS routines. (A more recent version of STDS will allow a length from one bit to 32767 bytes.) A great deal of effort is put into the analysis stage of preparing the data as a set form with an interactive version of STDS.

The data-set is not read into core until needed for a retrieval. Once in core, the data-set will remain there until it is DELETEd or DESTROYed, (or in the case of temporaries, SAVEd.) If MICRO needs more core than is available, the program will, if possible, release the core of permanent data-sets (they can always be read in again), and use that. If there is not enough core to release, MICRO will ask the user to DELETE or SAVE some temporary data-sets.

In core the data-set is one contiguous block containing a double word specifying the original size request for the data and how much space is not being used, then the fullword specifying the length and cardinality of the data-set, and then the data.

Data-sets that are confidential can be created and stored in "scrambled" form. This process involves the use of a password which is used in scrambling the data. A dump of the datafile would show only meaningless numbers. MICRO will request the password from the user when it reads in the data, and the data will be unscrambled at the time it enters core. If an incorrect password is given (the user has only

one chance), retrieval requests proceed with garbage results.

Presently the size of a data-set cannot exceed 1,044,480 bytes because of STDS program restrictions and MTS operating limitations. But a newer version of the STDS routines will allow buffered retrieval operations for larger data-sets. The data-set can exist on the same disk file as the dictionary or on a different file. Usually it is the latter case, since this allows more efficient use of disk space.

Temporary data-sets that are created as the result of a retrieval operation exist as regular MICRO data-sets with their own directories, DSCB's, and R and S Blocks. Much of the information in these structures is the same as that of the parent data-set. Whenever a temporary set is SAVEd as a permanent set, this information is used to generate the disk equivalents for directory, DSCB, R/S Blocks, and data descriptions. A temporary data-set which is a descendent of a scrambled data-set will be saved as a scrambled set, i.e., a password is requested to be used in encoding the output set.

## 4.8 Limitations and Plans for Improvement

One of the limitations of the MICRO system is the inability to handle arithmetic operations on numerical data values. While this capability is desired for a future version of the program, it is possible now to achieve the same results by using the MICRO command, WRITE FOR ANALYSIS... This command prepares the RESULT data-set in a form suitable for input to a statistical program on the U. of M. computer called CONSTAT. The desired operations can then be specified through CONSTAT.

A second limitation is in the editing and updating capabilities of the program. But we have immediate plans to implement a COMBINE command (union in set theoretic terms). This command will allow updating, modification, and merging of data-sets. Another modification further in the future will provide even greater editing capabilities.

Data manipulation on the bit level, an improved syntax analyzer, and facility with especially large data-sets are other desired improvements.

The program is designed so that routines for data structure control, I/O, space allocation, and retrieval of elements from the directory,

DSCB, and R/S Blocks are separate from the routines for the syntax analyzer, printing, and retrieval of data-sets. By keeping the two major parts of the system separate, we are more easily able to continually upgrade and improve the system.

# APPENDIX I

# MICRO INFORMATION RETRIEVAL SYSTEM (VERSION 3.9) TECHNICAL REFERENCE MANUAL

Michael A. Kahn
Donald L. Rumelhart
Boyd L. Bronson

## 5.0 SYSTEM BASICS

### 5.0.1 Introduction

MICRO is an interactive information retrieval system, designed to be used on remote terminals ("typewriter" consoles). The system is interactive in that the user issues a command request to the system and waits for the system to respond before issuing another command. This is particularly convenient since succeeding commands are frequently dependent upon the results of the previous queries.

The MICRO Information Retrieval System has general applicability to a number of problems. It could be used for job matching, medical research, inventory control or retrieval of management information. The system is very powerful in terms of the complexity of the requests which can be made. Also the structure of the commands is English-like which makes the system easy to learn and easy to remember.

The system is limited, however, in that it must be run on an IBM 360/67 using MTS (Michigan Terminal System). This system is resident on computers at the University of Michigan and at Wayne State University. Both of these computer installations are accessible via regular telephone lines.

### 5.0.2 Concepts and Facilities

#### Data Sets and Dictionaries

In MICRO different data collections are called data sets and the collection of data sets is referred to as a data base. A data set is simply a collection of records. Each record is composed of fields. MICRO can have many data sets available to it. Each data set is referenced by a name and is stored on a disk or tape file (auxiliary storage device). A user only needs the data set name to reference a data set. A list of the available data sets can be obtained at any time during a MICRO session.

MICRO data sets are self-describing; that is, descriptions of the contents, formats and names of the data items are themselves a part of the data set. These descriptions are called the dictionary. Information about the names of fields and about various attributes of the data set(s) is stored in the dictionary.

Therefore, a MICRO data set consists of records of data and a dictionary. The collection of records alone is called an STDS (Set-Theoretic Data Structure) set, and is in a form to be used by the STDS system.[1]

Each field within a record is given a name, called the "field name." Fields can be thought of as the attributes of each record. A field name is a string of no more than sixteen characters. For example, a field of social security numbers could be given the field name "SOCSECNUM". Since long field names can be cumbersome, an up-to-four-character abbreviation is allowed. The abbreviation for the above example might be "SSN".

The contents of a field in MICRO are called the values of the field. For example, the value of the field "SOCSECNUM" for the first record may be 123456789. However, certain kinds of data can best be thought of as falling into categories. Therefore, some values of fields may be referenced by category names. For instance, a field called SEX may have actual values of 2 or 3. However, it would be very convenient to refer to these values with the names MALE or FEMALE. Thus MICRO has a name-to-value association in the dictionary and data may be referred to by actual values or by category names, if any exist. Further, MICRO will use the category name when printing whenever possible. Category names may be up to eleven characters in length.

Data sets may or may not possess one or more of the following properties:

(1) Destroyable - certain data sets can never be destroyed while others can be permanently destroyed.

(2) Replaceable - certain data sets can never be replaced while others can.

(3) Scrambled - a confidential data set may be maintained in a specially encoded ("scrambled") form. If a data set is scrambled, a special password must be entered to

---

[1] STDS is a program product of Set Theoretic Information Systems Corporation, 117 N. First Street, Ann Arbor, Michigan 48108.

gain access to that data set.

(4) Permanent/temporary - a permanent data set is one that is available from one MICRO session to another. A temporary data set is one that is established during a MICRO session. Temporary sets are not available from one MICRO session to another. A temporary data set can be made permanent by using a SAVE command.

(5) Cross Tabulated - In general, a data set will have an instance of every record even if some records are identical to other records. Occasionally it is advantageous to combine identical records into a single record with a count of the number of such occurrences. This can be done with a CROSSTABULATE command and will produce what is referred to as a cross tabulated ("cross tabbed") set. A cross tabulated set can result in a considerable savings of space. However, it has the disadvantage of treating items in the aggregate which eliminates further use of some of the MICRO commands.

These properties of a data set are assigned when the data set is first created or SAVEd. The scrambled quality can be inherited. If a new data set is created by taking a subset of a scrambled data set, then the new set will also be scrambled.

At any time during a MICRO session there are a number of data sets available. Those data sets are called the universe of data sets. When a user renames a RESULT set (see below) that set temporarily becomes a member of the universe. A SAVE command can be used to add a set permanently to the universe.

Subsets and the Result Set

It is possible to extract from a data set a subset of records meeting certain criteria. This subset is itself a MICRO data set and is automatically given the data set name RESULT. The contents of this set are available until another command which creates a RESULT set is issued. If the user wants to keep a RESULT set, it can be given a new name. This newly named set then becomes a temporary set and is available for the remainder of the MICRO session. If desired, the set can be SAVEd and hence would become a permanent set.

Documentation Facility

A documentation facility (DOC) exists to allow the user's queries to be copied ("echoed") into certain files when the facility is enabled. This enables supervisory personnel to discover which commands are used most frequently and also allows

them to analyze the user/computer interaction when the user encounters problems with the system. At present the documentation facility defaults to OFF.

Macro Subsystem Within MICRO

A macro subsystem exists for the purpose of extending the MICRO language and to facilitate the reference of often-used sequences of MICRO commands. It can be accessed directly from MICRO. See the third part of this manual for the details of the macro facility. Descriptions of existing MACROS may be obtained by writing the authors in care of the Institute.

## 5.0.3 General Information

Prefix Characters

Whenever MICRO communicates with the user a special prefix character is printed as the first character of each line. These are:

(1) minus sign (-): request for a (first) line of a MICRO command.

(2) plus sign (+): request for additional lines of a command.

(3) asterisk (*): MICRO has generated the printed line.

(4) equal sign (=): indicates that the macro subsystem has generated the printed line.

Format of Commands

A MICRO command can be any number of lines in length. A period (.) is used to indicate the end of a command. A command consists of:

(a) the command name,

(b) a series of words such as data set names, field names, and special keywords as is indicated in the prototype of the command description. Each word must be separated by one or more blanks,

(c) a period.

## Notation Used in This Manual

The following conventions are used in this manual:

(1) Any word that appears in all capital letters must appear (as is) in the command.

(2) Words or phrases that appear in angle brackets (<>) should be replaced by an actual value or the appropriate name. For example: <field name> might be replaced with SEX or RACE.

(3) Words or phrases that appear in square brackets ([ ]) are optional and may be omitted. For example in the CALL command, [PARAMETER=<arg 1>,...,<arg n>] may be omitted if the subroutine requires no additional parameters.

(4) Words separated by a vertical bar (|) indicates that only one of the terms can validly appear in the command. For example, AVE|TOT indicates that either AVE or TOT should be used, but not both.

(5) The ellipsis (...) is used to indicate that a word-type or phrase-type may be repeated as many times as is required.

Further these conventions can be used in combination. For example, [AVE|TOT] indicates that either AVE, TOT or neither would result in a valid command.

## Abbreviations and Synonyms Acceptable in MICRO

The following symbols may be used interchangeably in a MICRO command:

(1) DATA SET[S] | DATASET[S] | SET[S]

(2) CATEGORY | CATEGORIES

(3) FIELD | FIELDS

(4) AND | &

(5) COMMAND | COMMANDS

(6) ALSO WHERE | OR WHERE | ;

(7) OR | |

In addition, most MICRO commands have synonyms. The synonyms for a command appear with that command's description.

## 5.0.4 Security Features of MICRO

There are several ways by which confidentiality is maintained. As was discussed previously, some data sets are scrambled and require a password to be accessed. Data sets are available only to users who know the proper index (password) required to access directory information for those data sets. (See the GET command.)

In addition to these measures some versions of MICRO require an additional password to use the system at all. If a protected version is being used, the MICRO system will prompt the user for information about the type of terminal being used and for the password. (See the Protection Key Facility.)

### Levels of Security

There are five ways in which confidentiality is maintained:

(1) Knowledge of a system (MTS) sign-on ID and appropriate password is required in order to access the Michigan Terminal System.

(2) The user must also know which command will initiate the proper version of the MICRO Information Retrieval System for the intended application.

(3) Certain versions of MICRO require the use of an additional password in order to further use MICRO. (See Protection Key Facility.)

(4) Data sets are available only to those users who know the proper index (password) required to access directory information for these data sets (See the GET command).

(5) Finally, for the most confidential information, a data set can be scrambled (specially encoded according to a key) which requires an additional password for access. It should be noted, however, that this is a very costly protection facility.

### Protection Key Facility

A protection key facility (PK) exists to limit access to the MICRO System to authorized users. If PK is ON when the user runs the MICRO system he will automatically be prompted for the type of terminal being used. There are two possibilities: (a) teletype compatible and (b) not teletype compatible. Authorized users will be informed of the distinction. After indicating terminal type,

he is prompted for an additional password. If he does not enter the proper password he is disconnected from the MICRO System. This additional password will be given only to those authorized to have access to MICRO and the data sets available. The Protection Key Facility prevents the use of the MICRO System from an unauthorized signon ID and its use from an authorized signon ID unless the proper protection key is used.

## 5.1 COMMAND DESCRIPTIONS

### CALL

COMMAND DESCRIPTION

PURPOSE: To execute a user or system subroutine while in the command mode of the MICRO Retrieval Language and to return to command mode upon completion of the subroutine.

PROTOTYPES AND DESCRIPTIONS:

(1) CALL <subroutine name> [USING <data set name>] [LIBRARY=<library name>] [KEEP=YES|NO] [PARAMETER=<arg 1>,...,<arg n>].

If USING <data set name> is omitted, then the RESULT data set is assumed.

If LIBRARY=<library name> is omitted, then the MICRO system library is used.

KEEP=YES|NO is used to control whether or not the subroutine is to remain loaded in core upon completion of the subroutine's execution. If it is omitted, the subroutine will not remain loaded.

PARAMETER=<arg 1>,...,<arg n> is used to pass additional parameters (arguments) to the subroutine. See below.

This command generates the following FORTRAN type call:

CALL <subroutine name> (NF,NFIELD,NC,NCAT,NOD, NDMAT,, <arg 1>,...,<arg n>).

Where

NF is the number of fields in the data set.

NFIELD is an integer matrix of two dimensions: NFIELD(6,NF). For the Ith field of the data set, NFIELD(1,I) through NFIELD(4,I) is the 16-character field name. NFIELD(5,I) is the number of categories in that field. NFIELD(6,I) is the index into NCAT where those category descriptions begin.

NC is the total number of categories, i.e., the sum of NFIELD(5,I).

NCAT is an integer matrix of two dimensions: NCAT(4,NC). For the Ith category, NCAT(1,I) through NCAT(3,I) is the category name. NCAT(4,I) is the category's value.

NOD is the number of records of data.

NDMAT is a two dimension integer matrix of data: NDMAT(NF,NOD).

<arg I> may be an integer, a double word floating point number or a string of up to 24 alphanumeric characters.

(2) CALL <subroutine name> WITHOUT DATA SET [LIBRARY=<library name>] [KEEP=YES|NO] [PARAMETER=<arg 1>,...,<arg n>].

This form of the call differs only from the above in that all arguments relating to the data set are omitted. This results in the following FORTRAN type call:

CALL <subroutine name> (<arg 1>,<arg2>,..., <arg n>)

COMMENTS:

(1) This command does <u>not</u> produce a new RESULT set.

(2) The order in which the keywords appear is arbitrary.

(3) If <arg I> is to be represented as a floating point number, it must contain a decimal point.

(4) The following is a list of acceptable abbreviations for keywords used in this command:

PARAMETER | PAR
LIBRARY | LIB
WITHOUT DATA SET | WITHOUT DATASET | W/O

(5) Fields containing certain character strings whose length is greater than four are currently truncated to four characters when passed to the called subroutine.

## CHANGE

### COMMAND DESCRIPTION

PURPOSE: To alter the data in the specified field in either all or certain records of the specified data set.

COMMAND SYNONYM: CH, ALTER, A

PROTOTYPES AND DESCRIPTIONS:

(1) CHANGE IN <data set name> [ALL RECORDS] SUCH THAT <field name> <operand> <new value> [... AND <field name> <operand > <new value>].

Where <operand> can be any of the following:

IS
ARE
IS EQUAL TO
ARE EQUAL TO
EQUAL
EQUALS
IS =
ARE =
=

and <new value> is a category name, integer value or character string.

This form changes in every record those fields specified by substituting the new value for the existing value.

(2) CHANGE IN <data set name> WHERE <phrase> SUCH THAT <field name> <operand> <new value> [... AND <field name> <operand> <new value>].

Where <operand> and <new value> are the same as defined above in (1). <phrase> refers to any phrase acceptable within the FIND command:

<field name> <verb> <category name>|<value> [...AND|OR|ALSO WHERE <field name> <verb>|<category name> <value>].

This form allows the user to find those records meeting the specified criteria and then to change the specified fields (as above in (1)).

(3) CHANGE IN <data set name> WHERE <phrase> TO <new value> [...AND <field name> <operand> <new value>].

This form is identical to (2) above except that the first field name whose value to be changed is not stated following the "TO". It is assumed to be the last field name stated in <phrase>.

COMMENTS:

(1) The RESULT set contains the same number of records as the original <data set name>. However, only those records meeting the specified search criteria will have been changed (i.e., the RESULT set may contain unchanged records).

(2) This command does produce a new RESULT set.

(3) The number of records changed is printed. The percentage of changed records out of all records searched in <data set name> is also printed.

(4) SUCH THAT and TO are synonyms and may be interchanged wherever either is used in the above prototypes.

## COMBINE

### COMMAND DESCRIPTION

PURPOSE: To combine two data sets into one result set, named RESULT.

COMMAND SYNONYMS: COMB, C

PROTOTYPE AND DESCRIPTION:

COMBINE <data set name 1> WITH <data set name 2>.

COMMENTS:

(1) This command can only be used when both data sets have identical fields. The RESULT is the union of the two sets. (Thus, duplicate records are removed from the result.)

(2) Caution: Currently, the count field of an XTAB set is treated like an ordinary field, thus combining two XTAB sets will result in the removal of duplicate records from the RESULT set. This may mean the loss of information.

(3) This command <u>does</u> produce a new RESULT set.

## COMMENT

### COMMAND DESCRIPTION

PURPOSE: To add a comment to the output of a MICRO session.

COMMAND SYNONYM: *

PROTOTYPE AND DESCRIPTION:

(1) COMMENT <phrase>.

Where <phrase> may be any character string.

COMMENTS:

(1) This command does <u>not</u> produce a new RESULT set.

## CROSSTABULATE

### COMMAND DESCRIPTION

PURPOSE: To perform a sorted (ascending, left to right) n-dimensional cross tabulation for the specified fields of a given data set.

COMMAND SYNONYMS: CROSSTAB, XTAB, X

PROTOTYPE AND DESCRIPTION:

CROSSTABULATE IN <data set name> <field name> [BY <field name> ...] BY [AVE|TOT] <field name>.

Where ... may specify additional BY <field name> phrases.

AVE (average) and TOT (total count) can only appear immediately before the final <field name>.

COMMENTS:

(1) This command does change the RESULT set.

(2) If AVE or TOT is used, then the data referred to by the last field will be treated numerically (instead of categorically).

(3) BY may be replaced by AND or , (comma).

## DESCRIBE

### COMMAND DESCRIPTION

PURPOSE: To get a description of the specified data set, categories, fields or command names.

COMMAND SYNONYM: D, DES

PROTOTYPES:

DESCRIBE DATA SET <data set name>.

DESCRIBE IN <data set name> FIELD <field name>.

DESCRIBE IN <data set name> CATEGORY <category name> OF <field name>.

DESCRIBE COMMAND <command name>.

COMMENTS:

(1) For a complete listing of all MICRO commands with their descriptions:

$COPY SBAU:COMLIST@CC.

or secure a copy of the Technical Reference Manual from the Institute.

(2) For a listing of just the basic MICRO commands with their descriptions:

$COPY SBAU:BASICLIST@CC.

(3) This command does not produce a new RESULT set.

## DESTROY

### COMMAND DESCRIPTION

PURPOSE: To permanently remove a data set from disk storage and from access by MICRO.

PROTOTYPE: DESTROY <data set name>.

COMMENTS

(1) Once a data set is DESTROYed, it can not be referenced again.

(2) After the DESTROY command is issued for a permanent data set, the user will be asked to confirm his action. To reconfirm, the user should type OK. Any other response will result in the command being cancelled.

(3) Certain data sets can not be destroyed. If this is attempted, the user will be so informed and no action will take place.

(4) This command does not produce a new RESULT set.

(5) If the data set is to be DESTROYed is a temporary data set, then the command is equivalent to a RELEASE command.

## END

### COMMAND DESCRIPTION

See STOP command description.

## FIND

### COMMAND DESCRIPTION

PURPOSE: To extract from a data set those records which match certain specified criteria and to store those records in the RESULT set.

COMMAND SYNONYM: F

PROTOTYPES AND DESCRIPTION:

FIND IN <data set name> WHERE <phrase>.

Where <phrase> may be one or more clauses separated by conjunctions:

$<clause_1>$ [... <conjunction> $<clause_n>$].

Where <clause > may be either of the following:

(a) <field name> <verb> <category name> | <value>

Where <verb> is any of those listed in SUPPLEMENTARY INFORMATION on the following page and where <conjunction> is either AND, OR or ALSO WHERE whose meanings are also discussed in SUPPLEMENTARY INFORMATION.

Where <value> may be an integer or character string. If it is a character string, it cannot be greater than 24 characters in length. Any string may be placed in primes ('), but if the string contains embedded blanks (such as 'JONES MOVING COMPANY') then primes must be used.

(b) <field name> IS BETWEEN <value 1> AND <value 2>.

Those elements equal to either <value 1> or <value 2> will also be included.

COMMENTS: (1) The number of records meeting the specified criteria is printed. The percentage of selected records (those in the RESULT set) out of all records searched in <data set name> is also printed.

(2) This command does produce a new RESULT set.

(3) When using a consecutive series of clauses which contain the same field name, the redundant field name need only be stated once. For example:

FIND IN JOBS WHERE ZIP IS 48104 OR ZIP IS 48105 OR ZIP IS 48108.

Is equivalent to:

FIND IN JOBS WHERE ZIP IS 48104 OR 48105 OR 48108.

SUPPLEMENTARY INFORMATION:

(1) FIND VERBS (IN EQUIVALENT GROUPS):

IS|ARE
IS|ARE EQUAL TO
EQUAL
EQUALS
IS|ARE =
=

IS|ARE NOT
IS|ARE NOT EQUAL TO
IS|ARE NOT =

IS|ARE GREATER THAN
IS|ARE >
>
IS|ARE NOT LESS THAN OR EQUAL TO
IS|ARE NOT EQUAL TO OR LESS THAN
IS|ARE NOT <=
IS|ARE NOT =<

IS|ARE LESS THAN
IS|ARE <
<
IS|ARE NOT GREATER THAN OR EQUAL TO
IS|ARE NOT EQUAL TO OR GREATER THAN
IS|ARE NOT >=
IS|ARE NOT =>

IS|ARE GREATER THAN OR EQUAL TO
IS|ARE EQUAL TO OR GREATER THAN
IS|ARE >=
IS|ARE =>
>=
=>
IS|ARE NOT LESS THAN

IS|ARE NOT <

IS|ARE LESS THAN OR EQUAL TO
IS|ARE EQUAL TO OR LESS THAN
IS|ARE <=
IS|ARE =<
<=
=<
IS|ARE NOT GREATER THAN
IS|ARE NOT >

(2) AND, OR and ALSO WHERE

Each <clause> of a FIND command is calculated separately in the order listed in the command statement. The operation defined by each <clause$_i$> is performed on some input set ($IN_i$) and a result set is generated. In order to avoid confusion between the result set generated by a FIND <clause> and a MICRO RESULT Set, the result generated by <clause$_i$> will be referred to as intermediate set ($I_i$). The input set for the first clause, <clause$_1$> is the originally named data set, specified in the command by <data set name>. Furthermore after each <clause$_i$> is processed a temporary set is created ($T_i$) which represents the results of the preceeding i <clause>'s.

There are three basic conjunctions used in the FIND command:

OR

AND

ALSO WHERE

The OR conjunction is somewhat analogous to the union operation in set theory. Specifically, if <clause$_i$> is preceeded by an OR conjunction then the input set $IN_i$ is the same as the input set for the preceeding <clause$_{i-1}$>, $IN_{i-1}$. Following the operation defined in <clause$_i$>, $T_i$ is formed as the result of a union performed on $T_{i-1}$ and $I_i$. ($T_i = T_{i-1}$ union $I_i$). (Note: $IN_1$ = <data set name>, the originally named set.)

The AND conjunction is somewhat analogous to the intersection operation in set theory. Specifically, if <clause$_i$> is preceeded by an AND conjunction then the input set IN is the temporary set from the previous i-1 clauses,

$T_{i-1}$. Furthermore $T_i = I_i$.

For example:

FIND IN JOBS WHERE ZIP IS 48104 AND DOT IS 802131 OR DOT IS 802198.

<clause$_1$> is "ZIP IS 48104". Hence, $IN_1$=JOBS and $T_1 = I_1$.

<clause$_2$> is "DOT IS 802131". Since this clause is preceeded by AND, $IN_2 = T_1$ and $T_2 = I_2$.

<clause$_3$> is "DOT IS 802198." Since this clause is preceeded by OR, $IN_i = IN_{i-1}$ which was equal to $T_1$ and $T_3 = T_2$ union $I_3$.

Since <clause$_3$> is the last clause, the MICRO RESULT set would be $T_3$ which would consist of those records in JOBS where ZIP was equal to 48104 and DOT was either 802131 or 802198.

The ALSO WHERE conjunction is more complicated, but it is somewhat analogous to having a series of FIND and COMBINE commands all in one command. If <clause$_i$> is preceeded by an ALSO WHERE, then <clause$_i$> behaves like <clause$_1$> in that $IN_i$ is the originally named data set. Also, $T_{i-1}$ is saved in a special set S, and $T_{i-1}$ is replaced with an empty set (the null set). When subsequent ALSO WHERE conjunctions are encountered then the temporary set say $T_{k-1}$ is unioned with S and that result is placed in S. Upon the conclusion of the last clause, $T_{LAST}$ is unioned with S to form the RESULT Set.

For example:

FIND IN JOBS WHERE ZIP IS 48103 AND DOT IS 802198 ALSO WHERE ZIP IS 48104 AND DOT IS 802131.

<clause$_1$> is "ZIP IS 48103". Hence $IN_1$=JOBS and $T_1 = I_1$.

<clause$_2$> is "DOT IS 802198". Since this clause is preceeded by AND, $IN_2 = T_1$ and $T_2 = I_1$.

Since the next clause is preceeded by ALSO WHERE, the previous temporary ($T_2$) is saved in a special set (S). $T_2$ is then replaced by an empty set.

<clause$_3$> is "ZIP IS 48104". Hence $IN_3$=JOBS (the clause following an ALSO WHERE behaves like <clause$_1$> in that $IN_1$ is the originally named data set) and $T_3=I_3$.

<clause$_4$> is "DOT IS 802131". Since this clause is preceeded by AND, $IN_4=T_3$ and $T_4=I_4$.

Upon conclusion of the last clause <clause$_4$>, $T_4$ is unioned with S to form the RESULT set.

## GET

### COMMAND DESCRIPTION

PURPOSE: To acquire the directory information for a group of data sets.

PROTOTYPE AND DESCRIPTION:

GET [DIRECTORIES FOR] <index>.

Where <index> is a character string of up to 16 characters. The index serves as a pointer to the directory information.

COMMENTS:

(1) It is not necessary to GET the directory information for data sets referenced in the user's directory.

(2) The various indices to the different data set groups are available to authorized users through the LMIS Project, Institute of Labor and Industrial Relations.

(3) This command does <u>not</u> produce a new RESULT set.

## LIST

### COMMAND DESCRIPTION

See PRINT command description.

## MTS

### COMMAND DESCRIPTION

See SYSTEM command description.

## NAME

### COMMAND DESCRIPTION

PURPOSE: To temporarily give a different name to any data set.

COMMAND SYNONYMS: N, RENAME, REN, RE

PROTOTYPE: NAME <old data set name> <new data set name>.

COMMENTS:

(1) The <old data set name> will no longer exist after it has been renamed.

(2) The "new name" may be up to 16 characters in length.

(3) Only a temporary data set may be renamed RESULT.

(4) A data set may not be renamed the name of any other data set.

## PRINT

### COMMAND DESCRIPTION

PURPOSE: To print on a terminal, file or device information about or from a data set, field, category, MICRO commands, etc.

COMMAND SYNONYMS: P, LIST, L

PROTOTYPES AND DESCRIPTIONS:

(1) PRINT [ALL] [DATA] SETS [NAMES].

The names and status of the data sets available to this user are printed.

There are five possible states for the status of a MICRO data set:

(a) DISK - permanent data set on disk; not in core.

(b) DISK* - permanent data set on disk; loaded in core.

(c) TAPE - permanent data set on tape; not in core.

(d) TAPE* - permanent data set on tape; loaded in core.

(e) TEMP* - temporary data loaded in core.

(2) PRINT IN <data set name> [ALL] FIELDS [NAMES].

The names of all fields in the indicated data set are printed.

(3) PRINT IN <data set name> [ALL] CATEGORIES OF <field name>.

The names of all categories in the indicated field in the indicated data set are printed.

(4) PRINT IN <data set name> COUNT.

The number of records in the data set is printed.

(5) PRINT [ON <file name>] [ENTIRE] <data set

name>.

The data for the entire data set is printed (on a file, if specified) record by record.

(6) PRINT IT.

This is synonomous with PRINT ENTIRE RESULT, a specific case of prototype (5).

(7) PRINT [ON <file name>] IN <data set name> <field name 1> [AND <field name 2> ...].

The data of the specified fields is printed (on a file, if specified) for the data set indicated.

(8) PRINT [ALL|BASIC] COMMANDS [NAMES].

The basic list or complete list of command names, and their synonyms, are printed. If neither ALL nor BASIC is specified, then ALL is assumed.

(9) PRINT COST.

The estimated cost of activity since the last COST interrogation is printed. If there was no previous interrogation, then the cost since entering MICRO is printed.

(10) PRINT VM|VMSIZE.

The current virtual memory (core storage) size used by this user is printed.

(11) PRINT TIME.

The current time and date are printed.

(12) PRINT STATUS.

TIME, COST and VMSIZE are printed.

COMMENTS: (1) This command does <u>not</u> produce a new RESULT set.

(2) See Section 1.2.4 for acceptable synonyms.

## READ

### COMMAND DESCRIPTION

PURPOSE: To read into MICRO a data set which does not have a dictionary and which is in STDS format.

PROTOTYPE AND DESCRIPTION:

READ FROM <file name>.

The dictionary and the data set name of the last explicitly named data set is used with the data read from <file name>.

COMMENTS:

(1) This command does produce a new RESULT set.

(2) Always employ the USE command prior to using the READ command. (See USE)

## RELEASE

### COMMAND DESCRIPTION

PURPOSE: To release the core storage associated with a data set in core.

COMMAND SYNONYM: R, REL

PROTOTYPES AND DESCRIPTIONS:

(1) RELEASE <data set name> [AND <data set name> ...].

The specified data set is released.

(2) RELEASE <data set name>.

(3) RELEASE ALL DATA SETS.

All data sets (both temporary and permanent) which are currently loaded are released.

(4) RELEASE *.

This is synonymous with (3).

(5) PURGE.

This is also synonymous with (3).

COMMENTS:

(1) If the data set is not in core (i.e., not loaded), the command has no effect.

(2) Permanent data sets which are not expected to be referenced again should be RELEASEd in order to reduce the costs associated with the core storage of data sets. This does not, however, preclude further reference to this permanent data set at a later time.

(3) If the data set to be released is a temporary data set, then the RELEASE command has the same effect as the DESTROY command and the data set can <u>not</u> be referenced again.

(4) This command does <u>not</u> produce a new RESULT set.

## REMOVE

### COMMAND DESCRIPTION

PURPOSE: To extract from a data set those records which match certain specified criteria and to leave in the RESULT set only those records not meeting the specified criteria.

COMMAND SYNONYM: REM

PROTOTYPES AND DESCRIPTIONS:

(1) REMOVE FROM <data set name> WHERE <phrase>.

Where <phrase> refers to any phrase acceptable within the FIND command.

(2) REMOVE <data set name 1> FROM <data set name 2>.

COMMENTS:

(1) This command does produce a new RESULT set.

(2) This command is equivalent to the relative complement concept in set theory.

(3) IN is a synonym for FROM; they may be used interchangeably.

(4) The number of records extracted is printed. The percentage of records extracted out of the total records in the specified set is printed. The number of elements in the result set is also printed.

## RENAME

### COMMAND DESCRIPTION

See NAME command description.

## REPLACE

### COMMAND DESCRIPTION

PURPOSE: To replace one data set with another data set.

COMMAND SYNONYM: REP

PROTOTYPE: REPLACE <old data set name> WITH <new data set name>.

COMMENTS:

(1) This command is most likely to be used after a data set has been altered by the CHANGE command.

(2) This ccmmand does not produce a new RESULT set.

(3) <old data set name> must refer to a permanent data which is stored on disk.

## RESTRICT

### COMMAND DESCRIPTION

PURPOSE: To extract those records from one data set whose values of a specified field match those values of a second field in a second data set. These extracted records are placed in the RESULT set.

COMMAND SYNONYMS: RES

PROTOTYPE: RESTRICT IN <data set name 1> WHERE <field name 1> IS <field name 2> IN <data set name 2>.

COMMENTS:

(1) This ccmmand does produce a new RESULT set.

(2) It is the extracted records of <data set name 1> that are placed in the RESULT set. If <data set name 1> and <data set name 2> are reversed, then a different RESULT set would be created.

## RESTRICTANDMERGE

### COMMAND DESCRIPTION

PURPOSE: To merge certain records from two data sets into a single expanded record in the RESULT set.

COMMAND SYNONYM: RAM

PROTOTYPE AND DESCRIPTION:

(1) RESTRICTANDMERGE <data set name 1> BY <field name A> [BY <field name M> ...] WITH <data set name 2> BY <field name B> [BY <field name N> ...].

This form equates specified field names of the first data set with the same number of specified field names of the second data set. Only if the values of the specified field names in the second data set equal those of the first are the entire two records merged (i.e., combined into an expanded record) and placed in the RESULT set.

(2) RESTRICTANDMERGE <data set name 1> BY <field name A> [BY <field name M ...] WITH <data set 2>.

In this form of the command the user only specifies field names for the first data set and MICRO assumes that the second data set has identical field names which are to be used for the comparison. However, the action resulting from this command form is the same as the first form.

(3) RESTRICTANDMERGE <data set name 1> WITH <data set name 2> BY <field name B> [BY <field name N> ...].

This form of the command is similar to the second but the field names are specified for the second data set.

COMMENTS: (1) This command <u>does</u> produce a new RESULT set.

(2) A comparison of the match fields is made of <u>every</u> record of the second data set with <u>each</u> record of the first data set. If any of the fields to be compared contain unique values

then the number of records in the RESULT set will be less than or equal to the number of records in the larger of the two sets. The number of records in the RESULT set depends of the number of matches and duplicates in the two sets. If there are no unique values in the fields compared, then the number of records in the RESULT set can be greater than the number of records in the larger of the two sets. This can result in extremely large sets due to the combinatorial effect of this situation.

(3) If both sets are cross tabulated sets then the RESULT set will not be a cross tabulated set. If only one of the data sets is cross tabulated then the RESULT set will be a cross tabulated set.

(4) Currently, after a RAM command the RESULT set will not contain the English descriptions for fields and categories of the second data set that could have been printed by the DESCRIBE command. However, all category and field names remain.

## SAVE

### COMMAND DESCRIPTION

PURPOSE: To permanently save a data set on disk storage for access through MICRO at a later date.

COMMAND SYNONYM: SA

PROTOTYPES: SAVE <data set name> [AS <new name>] [ON <file name>].

COMMENTS:

(1) If a <new name> is not specified, the SAVEd data set is identified by <data set name>.

(2) If the <file name> is not specified, then a new file is created and the new file name will be printed.

(3) If <file name> is specified but no file of that name exists, MICRO will create a file with that name.

(4) This command does not produce a new RESULT set.

(5) If a <new name> is specified, then the data set will be RENAMEd automatically.

(6) MICRO will not allow the user to SAVE a data set with the same name as an already exisitng set.

(7) <file name> may contain up to 16 characters.

(8) The dictionary information for a data set is stored on a file separate from the actual data itself. The procedure for selecting the file name for the data is described above. The name used for the dictionary file is formed by appending a "#" character on the end of the file name for the data, unless that name has 16 characters in which case a "#" character replaces the last character of the data file name.

## SELECT

### COMMAND DESCRIPTION

PURPOSE: To extract certain fields from each record of a data set and to store the extracted fields as records in the RESULT set.

COMMAND SYNONYM: S

PROTOTYPE: SELECT IN <data set name> <field name 1> [... AND <field name n>].

COMMENTS:

(1) This command does produce a new RESULT set.

(2) The SELECT command can be used to extract specified fields from a data set, thereby enabling the user to work with a reduced data set. Further, the original data set may be removed from core using a RELEASE command to avoid unnecessary costs associated with core storage.

(3) Caution: SELECT should not be used on a cross tabulated set as the result could be meaningless. Use the XTAB command instead.

(4) Caution: The values of the fields SELECTed should be such that the result is a subset that has no duplicate (identical) records, otherwise duplicates will be deleted without any record of count. This is most easily achieved if the first field name SELECTed contains a unique value for each record in the set. The RESULT set is sorted in ascending order according to the respective order of the fields specified (left-to-right) and duplicate records are eliminated.

(5) Caution: If the purpose of using a SELECT command is to prepare a RESULT set for use with the WRITE FOR ANALYSIS command, care should be taken to ensure that duplicate records are not eliminated because they lack a unique value (key). As stated in (4) above, the duplicates will be deleted and no count will be made. If a count is desired, use the XTAB command to select the desired fields. If duplicates are desired for analysis, the WRITE FOR ANALYSIS command should be used. (See WRITE command.)

(6) AND may be replaced by , (comma) or BY.

## SET

### COMMAND DESCRIPTION

PURPOSE: To change the action of one of various MICRO Retrieval System facilities.

COMMAND SYNONYMS: TURN, T

PROTOTYPE AND DESCRIPTION:

(1) SET <facility> <status>.

Where <status> is ON or OFF and <facility> is one of the following:

(a) CLOCK (default: OFF)

When the status is ON, MICRO will print after every command the number of seconds of elapsed time and CPU time since the last command. It is initially OFF and when it is first set is ON, it prints the time of day.

(b) ERROR CORRECTION (default: OFF)

When the status is ON, MICRO will attempt to interpret misspelled key words in MICRO commands.

(c) MACRO ECHO (default: OFF)

When the status is ON, any MICRO statement generated by a macro statement will be printed.

(d) ECHO (default: ON)

When the status is OFF, the printing of any remarks following the execution of a command (such as " XX RECORDS IN RESULT SET," etc.) will be suppressed. Error messages will not be suppressed, however.

COMMENTS: (1) This command does <u>not</u> produce a new RESULT set.

## SIGNOFF

### COMMAND DESCRIPTION

PURPOSE: To permanently terminate the current MICRO session and to sign-off the computer system (MTS).

COMMAND SYNONYM: SIG

PROTOTYPE: SIGNOFF [S|$].

Where S is the short form and $ is the summary form (only dollar amount used and dollar amount remaining are printed when $ is used).

## SORT

### COMMAND DESCRIPTION

PURPOSE: To perform an n-dimenseional sort for the specified fields of a given data set.

COMMAND SYNONYMS: SO

PROTOTYPE AND DESCRIPTION:

SORT IN <data set name> <field name> [... BY <field name>].

Where ... may specify additional BY <field name> phrases.

COMMENTS:

(1) This command does produce a new RESULT set.

(2) The number of records sorted is printed.

(3) The RESULT of the command is similar to the RESULT of the XTAB command except that duplicate occurrences are not eliminated and thus there is no COUNT field.

(4) The sort is applied only to those field(s) specified and the RESULT set is ordered accordingly. The unspecified fields are retained, but not used as part of the sort key.

(5) Sort reorders the data set and sorts on the specified fields followed by the unspecified fields in the order that they originally appeared in <data set name>.

## STOP

### COMMAND DESCRIPTION

PURPOSE: To permanently terminate the current MICRO session.

COMMAND SYNONYMS: ST, END

PROTOTYPE: STOP.

COMMENTS: (1) MICRO cannot be re-entered via a $RESTART command.

## SYSTEM

### COMMAND DESCRIPTION

PURPOSE: To temporarily leave the MICRO Information Retrieval System and return to the command mode of MTS.

COMMAND SYNONYMS: SYS, MTS

PROTOTYPE: SYSTEM.

COMMENTS: (1) MICRO can be re-entered via a $RESTART command.

(2) This command does not produce a new RESULT set.

## TAB

### COMMAND DESCRIPTION

PURPOSE: To perform a sorted one-dimensional frequency distribution for the specified field of the given data set.

COMMAND SYNONYMS: FREQUENCY, FREQ

PROTOTYPE AND DESCRIPTION:

TAB IN <data set name> [AVE|TOT] <field name>.

COMMENT: (1) This ccmmand does produce a new RESULT set.

(2) If AVE or TOT is used, the data referred to by the field name will be treated numerically (instead of categorically).

## USE

### COMMAND DESCRIPTION

PURPOSE: To make a data set the "last explicitly named data set" as required by the READ command.

COMMAND SYNONYM: U

PROTOTYPE: USE <data set name>.

COMMENTS: (1) This command does not produce a new RESULT set.

## WRITE

### COMMAND DESCRIPTION

PURPOSE: To write a MICRO data set for future use outside of the MICRO Information Retrieval System.

COMMAND SYNONYM: W

PROTOTYPES AND DESCRIPTION:

(1) WRITE <data set name> [ON <file name>].

This form of the command writes the specified data set in STDS form on the specified file.

(2) WRITE FOR ANALYSIS <data set name> [ON <file name>] [USING <field name 1> [AND <field name 2> ...]].

Where [USING <field name 1> [AND <field name 2> ...]] is similar to the SELECT command in selecting certain fields to be written for analysis. Unlike the SELECT command, however, duplicates are not eliminated.

This form of the command writes the specified data set on the specified file for use with MIDAS at the University of Michigan and CONSTAT at Wayne State University. (MIDAS and CONSTAT are general-purpose statistical programs. See instructions for using MIDAS with MICRO in Appendix A.)

COMMENTS: (1) If <file name> is not specified, then a new file is created and the new file name will be printed.

(2) If <file name> is specified, but no file of that name exists, MICRO will create a file with that name.

(3) This command does <u>not</u> produce a new RESULT set.

# XTAB

## COMMAND DESCRIPTION

See CROSSTABULATE command description.

## 5.2 MACRO SUBSYSTEM

### 5.2.1 Introduction

The macro subsystem is an extension of the MICRO Command Language. It provides a convenient way to generate a desired sequence of MICRO commands in one or more MICRO sessions. The macro-definition is written only once, and a single command, the macro-command command, is issued each time the user wants to generate the desired sequence of MICRO commands.

An additional facility, called conditional macro generation, allows the user to alter the sequence of commands to be generated during an interactive MICRO session.

It is suggested that this section be read through once and then that Appendix B be referred to for an example of a macro-description and its use.

### 5.2.2 Macro Libraries

The same macro-definition may be made available to more than one user by placing the macro-definition in a macro library. Once a macro-definition has been placed in a macro library, it may be used by typing its corresponding macro-command during a MICRO session. The procedures used for placing macro-definitions into a macro library will be described in a future update to this manual.

There are two different types of macro libraries. One is the system macro library, the other, a user macro library. Both have the same structure and function. All users have access automatically to the one system macro library during a MICRO session. In addition, when running MICRO the user can indicate which of possibly several user libraries is to be referenced. The user macro libraries contain private macros which may be available only to specified users.

### 5.2.3 Macro-Definitions

A macro-definition is a set of statements that provides MICRO with:

(1) The name and format of the macro-command, and

(2) The sequence of commands the subsystem generates when the macro-command appears.

Every macro-definition consists of:

(1) A macro-definition name/prototype statement (The DEFINE Statement),

(2) Zero or more delimiter list statements,

(3) Zero or more model statements or conditional macro-generation statements,

(4) A macro-definition end statement. (The % END DEFINITION statement).

A macro-definition cannot appear within a macro-definition, nor can a macro-command. A macro-definition must be available to the system before it's corresponding macro-command is used.

### 5.2.4 Macro-Command

Macro-commands are commands issued within the main MICRO system. When MICRO recognizes an input line as a macro-command, the macro subsystem is entered. The subsystem, under control of the appropriate macro-definition, generates MICRO commands. The generated commands are then processed like any other MICRO command.

### 5.2.5 Variable Symbols

A variable symbol is a symbol that is assigned different values by either the user or the macro subsystem. When the macro subsystem interprets a macro-definition, variable symbols in the model statements are replaced by values assigned to them. By changing the values assigned to a variable symbol, the user can vary the contents of the generated commands.

There are three types of variable symbols: symbolic parameters, symbolic delimiters, and system variable symbols.

Symbolic parameters are written with an at-sign (@) prefix, followed by one to two digits in the range 1-99. Symbolic parameters are assigned values by the user each time he writes a macro-command. The subsystem will accept as valid any such value; however, the value put in the generated command may be rejected as invalid by the main MICRO system.

Symbolic delimiters are written with an at-sign prefix followed by the letter D followed by one to two digits in the range 1-99. Like symbolic parameters, symbolic delimiters are assigned values by the user each time he writes a macro-command. However, the subsystem will accept as valid values only those that are listed in the macro-definition.

System variable symbols are assigned values by the subsystem each time it processes a macro-command. Currently, there is one system variable. It is written as @NULL and is assigned the value of a null string.

### 5.2.6 Writing Macro-Descriptions/Commands

A macro-command can have any number of lines. It is terminated by a period (.). A macro name/prototype statement can also have any number of lines and is also terminated by a period. Model statements are only one line long. Conditional generation and delimiter list statements are also only one line long and the first character of the line must be a percent sign (%). The macro-definition end statement consists of only one line and the first character must be a percent sign.

In this section, the same notation is used as in the main section of this publication.

## DEFINE

MACRO SUBSYSTEM STATEMENT

PURPOSE: To indicate the beginning of a macro-definition and to specify the macro name and the format of all macro-commands that refer to that macro-defintion.

SYNONYM: DEF

PROTOTYPE: DEFINE <macro name> [<delim 1>] [<param 1>] [[<delim m>] [<param n>] ... ].

Where <delim m> is a symbolic delimiter of the form: @Dn

and <param m> is a symbolic parameter of the form: @m

COMMENTS:

(1) This statement must be the first of every macro-definition.

(2) The macro is given the name <macro name>.

(3) The presence and position of symbolic delimiters and/or symbolic parameters indicate where the actual delimiters and parameters appear in the macro-command.

(4) The prefix character is changed to =.

EXAMPLES:

```
DEFINE PUT.

DEFINE LOOK AT @D1 @1 @2.
```

## DELIMETER LIST

MACRO SUBSYSTEM STATEMENT

PURPOSE: To specify the list of values that a symbolic delimiter may be assigned by a macro-command.

PROTOTYPE AND DESCRIPTION:

%D<n> (<char string 1> [,<char string m> ... ])

The nth symbolic delimiter is assigned the list of values in parentheses.

COMMENTS:

(1) Delimiter list statements must immediately follow a DEFINE statement and precede any model or conditional generation statements.

(2) Not every symbolic delimiter used in a description need be given a list since every symbolic delimiter is assumed to have a comma (,) as an allowable value.

EXAMPLES:

```
%D1  (BY,AND,WITH)
```

## END DEFINITION

MACRO SUBSYSTEM STATEMENT

PURPOSE: To indicate the end of a macro definition.

PROTOTYPE: %[<label>] END [DEFINITION]

COMMENTS:

(1) This statement must be the last of a macro-definition.

(2) The macro-subsystem is exited and the MICRO system is re-entered (and the prefix character reverts to a -).

EXAMPLES:

```
%  END DEFINITION

%OUT END
```

## GOTO

MACRO SUBSYSTEM STATEMENT

PURPOSE: To unconditionally branch to another macro statement within the current macro definition.

PROTOTYPE: %[ <label 1>] GOTO <label 2>

Macro processing will continue at the statement labeled <label 2>.

COMMENTS: (1) This is a conditional generation statement.

EXAMPLES: % GOTO OTHER

%HERE GOTO TEST2

# IF

MACRO SUBSYSTEM STATEMENT

PURPOSE: To test the value of a symbolic variable and, depending on that value, process either the next macro statement or one elsewhere in the macro-definition.

PROTOTYPE: %[<label 1>] IF <sym var 1> EQ|NE <char string>|@NULL|<sym var 2> GOTO <label 2>

where <sym var> is any symbolic delimiter of the form @Dn or symbolic parameter of the form @n.

If EQ is specified, then if the value of <sym var 1> equals @NULL or <char string> or <sym var 2> then the macro statement labled by <label 2> is processed next. Otherwise the next macro statement will be processed.

If NE is specified, then the condition tested for is inequality.

COMMENTS: (1) This is a conditional generation statement.

EXAMPLES: % IF @1 EQ PLOT GOTO THERE

%BITEST IF @D2 NE @NULL GOTO OUT

## MACRO-COMMAND

MACRO SUBSYSTEM STATEMENT

PURPOSE: To start the generation of MICRO commands according to a specified macro-definition.

PROTOTYPE: <macro name> [<value delim 1>] [<value param 1>] [[<value delim N>] [<value param M>] ...]

COMMENTS:

(1) The definition named <macro name> is invoked and starts generating MICRO commands. The values specified by the macro-command replace the corresponding symbolic variables in the definition.

(2) As each MICRO command is generated it is processed.

(3) Normally the MICRO commands generated are not printed. See the MICRO SET command described previously on how to effect printing of the generated commands.

Examples: DISPLAY JOB DESCRIPTIONS.

## MODEL

MACRO SUBSYSTEM STATEMENT

PURPOSE: To specify the text to be used in generating a MICRO command.

PROTOTYPE: Any sequence of blanks and/or characters including symbolic delimeter and/or parameter variables.

COMMENTS:

(1) The text is put, as is, into the MICRO command being generated. However, any symbolic delimiters or symbolic parameters are replaced by the actual values in the macro-command.

EXAMPLE: FIND IN @1 WHERE @2 @D5 28

## NOP

MACRO SUBSYSTEM STATEMENT

PURPOSE: To provide a point of reference for conditional generation of MICRO commands.

PROTOTYPE: %[<label>] NOP

COMMENTS: (1) The subsystem just passes over NOP statements.

EXAMPLE: %THERE NOP

## APPENDIX I-1
## MIDAS

Michigan Interactive Data Analysis System (MIDAS) is a statistical package available on MTS at the University of Michigan only.[2] The MICRO command

WRITE FOR ANALYSIS <data set name> [ON <file name>].

creates a file with the same name as <data set name> (or if ON <file name> is specified, it creates a file with that name). This information can be accessed at any time after the current MICRO session by issuing the following command

$SOURCE <file name>

where <file name> refers to the file created by the WRITE FOR MIDAS command. If <file name> is a temporary file, it can only be accessed after leaving MICRO and prior to $SIGning off the system.

Once you have $SOURCEd the file and entered MIDAS, it should be noted that:

(a) All field names in the data set are MIDAS variables;

(b) These variables are MIDAS analytical variables which are real, double precision numbers.

---

[2] See Documentation for MIDAS, Statistical Research Laboratory, University of Michigan, 1972, 126 pp.

## APPENDIX I-2
## MACRO EXAMPLE

The following is a sample macro-definition:

```
DEFINE GIVE @D1 @D2 @1 @D3 @2.

%D1 (STATISTICS, RANGE, MOMENTS)

%D2 (FOR,OF)

%D3 (IN)

SELECT IN @2 @1.

CALL TALYHO KEEP=Y

% IF @D1 EQ STATISTICS GOTO THREE

% IF @D1 EQ RANGE GOTO ONE

     PAR=2.

% GOTO END

%THREE NOP

     PAR=3.

% GOTO END

%ONE NOP

     PAR=1.

%END END DEFINITION
```

The macro GIVE can be invoked by any of the following forms of macro-instructions:

(1) GIVE RANGE OF <field name> IN <data set name>.

(2) GIVE STATISTICS FOR <field name> IN <data set name>.

(3) GIVE MOMENTS OF <field name> IN <data set name>.

(4) GIVE STATISTICS OF <field name> IN <data set name>.

The form (1) would cause the generation of the MICRO commands:

```
SELECT IN <data set name> <field name>.
```

```
CALL TALYHO KEEP=Y PAR=1.
```

While forms (2) and (3) would produce:

```
SELECT IN <data set name> <field name>.
CALL TALYHO KEEP=Y PAR=3.
```

And form (4) would produce:

```
SELECT IN <data set name> <field name>.
CALL TALYHO KEEP=Y PAR=2.
```