

MICRO
Information Management System
(Version 5.0)
Reference Manual

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

October 1, 1977

Malcolm S. Cohen, Co-Director of Research

Institute of Labor and Industrial Relations
The University of Michigan - Wayne State University
506 E. Liberty Street, Room 218
Ann Arbor, Michigan 48109
(313)-763-3116

Copyright (c) 1977 by the Institute of Labor and Industrial Relations, The University of Michigan - Wayne State University. General permission to republish, but not for profit, all or part of this manual is granted provided that the Institute of Labor and Industrial Relations' copyright notice is given and that reference is made to the authors and title of this manual, to its date of publication, and that reprinting privileges were granted by permission of the Institute of Labor and Industrial Relations.

Distributed by the Institute of Labor and Industrial Relations, P.O. Box B-1, Ann Arbor, Michigan 48106. Price:

Earlier versions of MICRO were supported in part by the Office of Research and Development, Manpower Administration, under contract 71-24-70-02 and grant number 21-16-74-21.

Michael A. Kahn is Principal Consultant and Owner of PRISM ASSOCIATES of Ann Arbor, Michigan. Donald L. Rumelhart was the Systems Project Coordinator and Boyd L. Bronson was a Senior Programmer Analyst at the Institute of Labor and Industrial Relations responsible for the current version of MICRO.

Questions and comments should be directed to:

David E. Metrick, Senior Programmer Analyst
Institute of Labor and Industrial Relations
The University of Michigan - Wayne State University
506 E. Liberty Street, Room 213
Ann Arbor, Michigan 48109
(313)-763-3116

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION	13
1.1 OPERATING ENVIRONMENT	13
1.2 HISTORY	14
1.3 ACKNOWLEDGEMENTS	15
1.4 HOW TO USE THIS MANUAL	15
1.5 COMPOSITION STYLE	16
1.6 OTHER RECOMMENDED READING	16
CHAPTER 2 - CONCEPTS AND TERMINOLOGY	19
2.1 DATA SETS, RECORDS, FIELDS, VALUES AND CATEGORIES	19
2.2 DATA FILES, DICTIONARIES AND DIRECTORIES	19
2.3 DESCRIPTIONS AND NAMES	20
2.4 PERMANENT AND TEMPORARY SETS AND THE RESULT SET	20
2.5 ORDERING OF RECORDS IN A DATA SET	21
2.6 CROSSTABULATED DATA SETS	22
2.6.1 EXAMPLE OF CROSSTABULATED DATA SET	22
2.6.2 THE COUNT FIELD	23
2.6.3 THE PERCENT FIELD	23
2.7 BUILDING AND USING MICRO DATA SETS	23
2.7.1 CONCEPTUALIZATION OF DATA MANAGEMENT NEEDS	26
2.7.2 LOGICAL DEFINITION	26
2.7.3 PHYSICAL DESIGN PROCESS	26
2.7.4 DATA SET DESIGN	26
2.7.5 MICRO.DEF PROGRAM	26
2.7.6 DICTIONARY	26
2.7.7 DIRECTORY	26
2.7.8 DATA	26
2.7.9 DATA ENTRY PROCESS	26
2.7.10 DATA FILE	27
2.7.11 DATA SET DOCUMENTATION PROCESS	27

2.7.12 DATA SET DOCUMENTATION	27
2.7.13 USER DECISION PROCESS	27
2.7.14 MICRO COMMANDS	27
2.7.15 MICRO COMMAND PROCESSOR	27
2.7.16 REPORTS AND ANSWERS	27
2.8 PROGRAMS IN MICRO	27
2.8.1 MICRO	28
(1) MASTER SIGNON-ID	28
(2) COMMAND TRACE FACILITY	28
(3) RUNFILE	28
(4) CONDITIONAL MACRO PROCESSOR	28
2.8.2 MICRO.DEF	29
2.8.3 MICRO.DIR	29
2.8.4 MICRO.DOC	29
2.8.5 MICRO.ADD	29
2.8.6 MICRO.UP	29
CHAPTER 3 - CONCEPTUALIZATION OF DATA MANAGEMENT NEEDS - THE LOGICAL DEFINITION	31
3.1 ABSTRACTION OF DATA	31
3.1.1 ABSTRACTION OF DATA COLLECTIONS	31
3.1.2 ABSTRACTION OF DATA ELEMENTS (FIELDS)	32
(1) NATURAL AND CODIFIED DATA	32
(2) NUMERICAL AND CHARACTER DATA	33
(3) REPEATING GROUPS	33
3.2 INITIALIZATION OF DATA SETS	33
3.3 RETRIEVAL OF INFORMATION FROM DATA SETS	34
3.4 MODIFICATION OF A DATA SET	34
3.5 INTEGRATION OF DATA SETS	35
3.6 OVERVIEW OF EXAMPLE	35
CHAPTER 4 - TRANSFORMING THE LOGICAL DEFINITION INTO A PHYSICAL DESIGN	39
4.1 AGGREGATE DATA SET REQUIREMENTS	39
4.1.1 DATA SET NAME	39
4.1.2 DATA SET DESCRIPTION	40
4.1.3 DATA SET DESTRUCTION STATUS	40
4.1.4 DATA SET REPLACEMENT STATUS	40

4.1.5 DATA FILE NAME	40
4.1.6 DICTIONARY FILE NAME	41
4.1.7 CONVERSION FILE NAME	41
4.1.8 CROSSTABULATED DATA SET INDICATOR	41
4.1.9 SCRAMBLED DATA SET INDICATOR	41
4.1.10 DICTIONARY USECCUNT	42
4.2 DEFINING EACH FIELD	42
4.2.1 FIELD NAMES AND ABBREVIATIONS	42
4.2.2 FIELD DESCRIPTION	43
4.2.3 SELECTION OF FIELD TYPE	43
4.3 NUMERICAL DATA	43
4.3.1 TYPE U - UNSIGNED INTEGER FIELD WITHOUT CATEGORIES	43
4.3.2 TYPE S - SIGNED INTEGER FIELD WITHOUT CATEGORIES	44
4.3.3 TYPE UC - UNSIGNED INTEGER FIELD WITH CATEGORIES	44
4.3.4 TYPE SC - SIGNED INTEGER FIELD WITH CATEGORIES	44
4.3.5 FIELD SIZE OF INTEGER FIELDS	44
4.3.6 THE DIFFERENCES BETWEEN SIGNED AND UNSIGNED INTEGERS ...	45
4.3.7 SCALING	45
4.3.8 REPRESENTATION OF DECIMAL NUMBERS - SCALING BY DIVISION	45
4.3.9 REPRESENTATION OF LARGE NUMBERS - SCALING BY MULTIPLICATION	46
4.3.10 SCALING BY ADDITION AND SUBTRACTION	46
4.3.11 SCALING OVERFLOW	46
4.4 CHARACTER DATA	46
4.4.1 FIXED-LENGTH CHARACTER FIELDS	47
4.4.2 TYPE C - CHARACTER STRING FIELDS WITHOUT CATEGORIES	47
4.4.3 TYPE CC - CHARACTER STRING FIELDS WITH CATEGORIES	47
4.4.4 FIELD SIZE OF CHARACTER STRING FIELDS	47
4.4.5 CONSTRAINTS ON FIXED-LENGTH CHARACTER STRING FIELDS AND SOLUTIONS	48
4.4.6 VARYING LENGTH CHARACTER FIELDS	48
4.4.7 TYPE E - EXTERNAL DATA FIELDS	48
4.4.8 FIELD SIZE - EXTERNAL DATA FIELDS	49
4.4.9 REPRESENTING CHARACTER STRING DATA AS CATEGORIES	49
4.4.10 CODIFICATION OF DATA VALUES	49
4.4.11 REQUIRED CATEGORIES	50
4.4.12 CATEGORICAL INFORMATION	50
4.4.13 CATEGORY NAME AND ABBREVIATION	50
4.4.14 CATEGORY TYPE	51
4.4.15 CATEGORY VALUE	51
4.4.16 CATEGORY DESCRIPTION	52
4.4.17 SHARED CATEGORIES	52
4.4.18 DEFAULT VALUES AND REQUIRED DATA	52

4.5	ADVANCED DESIGN CONCEPTS	52
4.5.1	INTEGRATION OF DATA SETS	52
4.5.2	REPEATING GRUPS	54
4.5.3	OVERLAPPING FIELDS	55
4.6	PHYSICAL DESIGN FOR THE EXAMPLE DATA SETS	55
4.6.1	WORKSHEET FOR THE PIN-AID DATA SET	57
4.6.2	WORKSHEET FOR THE STUDENTS DATA SET	58
CHAPTER 5	- DATA SET DEFINITION	59
5.1	HOW TO RUN THE DATA SET DEFINITION PROGRAM	59
5.2	GENERAL RULES	59
5.3	PROMPTING FOR DEFINITIONAL INFORMATION	61
5.4	GENERAL DATA SET INFORMATION	61
5.4.1	DICTIONARY FILE NAME	62
5.4.2	MASTER SIGNON-ID	62
5.4.3	MICRO RUNFILE NAME	63
5.4.4	LOCAL COMMAND TRACE FILE NAME	63
5.4.5	MASTER COMMAND TRACE FILE NAME	63
5.4.6	MICRO DATA SET NAME	63
5.4.7	DATA SET DESCRIPTION	64
5.4.8	DATA SET DESTRUCTION STATUS	64
5.4.9	DATA SET REPLACEMENT STATUS	64
5.4.10	DATA FILE NAME	64
5.4.11	CONVERSION FILE NAME	64
5.4.12	USECOUNT	65
5.4.13	CROSSTABULATED DATA SET STATUS	65
5.4.14	DATA SET SCRAMBLE STATUS	65
5.5	INFORMATION PERTAINING TO FIELDS	65
5.5.1	FIELD NAME	66
5.5.2	FIELD ABBREVIATION	66
5.5.3	FIELD TYPE	66
5.5.4	CHARACTER FIELD LENGTH	67
5.5.5	OVERLAP COLUMN	67
5.5.6	OVERLAP LENGTH	67
5.5.7	MAXIMUM VALUE	67
5.5.8	EXTERNAL DATA FILE NAME	67
5.5.9	SCALING FUNCTION	67
5.5.10	SCALING CONSTANT	68
5.5.11	CATEGORY REQUIRED STATUS	68
5.5.12	DATA REQUIRED STATUS	68
5.5.13	FIELD DESCRIPTION	68

5.6 INFORMATION PERTAINING TO CATEGORIES	68
5.6.1 CATEGORY NAME	69
5.6.2 CATEGORY ABBREVIATION	69
5.6.3 CATEGORY TYPE	69
5.6.4 CATEGORY VALUE	69
5.6.5 CATEGORY DESCRIPTION	70
5.6.6 AN ALTERNATIVE TECHNIQUE FOR ENTERING CATEGORY INFORMATION	70
5.7 CHANGING THE DATA SET DEFINITION - CORRECTION MODE	71
APPEND COMMAND DESCRIPTION	73
CHANGE COMMAND DESCRIPTION	74
CLOSE COMMAND DESCRIPTION	76
DELETE COMMAND DESCRIPTION	77
DOCUMENT COMMAND DESCRIPTION	78
INSERT COMMAND DESCRIPTION	79
MODIFY COMMAND DESCRIPTION	80
REDEFINE COMMAND DESCRIPTION	81
STOP COMMAND DESCRIPTION	82
5.8 CREATING DICTIONARIES FOR THE EXAMPLE SETS	83
5.8.1 DEFINITION OF THE FIN-AID DATA SET	84
5.8.2 DEFINITION OF THE STUDENTS DATA SET	96
5.9 CREATION AND MODIFICATION OF THE DIRECTORY ENTRY	105
5.10 MICRO.DOC PROCFAM	106
5.11 TAPE DATA SETS	107
CHAPTER 6 - PUTTING DATA INTO A DATA SET	109
6.1 INTERACTIVE PROMPTING METHOD	109
6.1.1 ADVANTAGES	110
6.1.2 DISADVANTAGES	110
6.1.3 TECHNIQUE	110
6.2 BATCH FREE-FORMAT METHOD	110
6.2.1 ADVANTAGES	110
6.2.2 DISADVANTAGES	111
6.2.3 TECHNIQUE	111
6.3 USER-SPECIFIED BATCH FIXED-FORMAT METHOD	111
6.3.1 ADVANTAGES	111
6.3.2 DISADVANTAGES	111
6.3.3 TECHNIQUE	111

6.4 GENERAL RULES FOR BATCH FIXED-FORMAT INPUT	112
6.4.1 PROMPTING FOR DATA FORMAT INFORMATION	113
6.5 GENERAL INFORMATION FOR BATCH FIXED-FORMAT	114
6.5.1 MICRO DATA SET NAME	114
6.5.2 MAXIMUM TOTAL NUMBER OF COLUMNS ON ALL CARDS	114
6.5.3 MAXIMUM NUMBER OF CARDS IN ONE RECORD	114
6.5.4 VARYING NUMBER OF CARDS PER RECORD	114
6.5.5 INCLUDING OR EXCLUDING CERTAIN RECORDS	115
6.5.6 SAMPLING STRING	116
6.6 INFORMATION PERTAINING TO DATA IN EACH FIELD - FIXED-FORMAT	116
6.6.1 STARTING COLUMN (OR SUBROUTINE FILE)	116
6.6.2 CARD NUMBER	117
6.6.3 INPUT TYPE	117
6.6.4 LENGTH	118
6.6.5 MISSING DATA VALUE	118
6.6.6 RECODE TYPE	118
6.7 CORRECTION MODE - FIXED-FORMAT	120
CLOSE	121
CONVERT	122
DOCUMENT	123
DELETE	124
RECODE	125
REDO	126
STOP	127
6.8 CONVERSION MODE - FIXED-FORMAT	128
6.9 CUSTOM PROGRAMMING METHOD	129
6.9.1 ADVANTAGES	130
6.9.2 DISADVANTAGES	130
6.9.3 TECHNIQUE	130
CHAPTER 7 - MODIFYING A DATA SET	131
7.1 MODIFYING VIA THE CHANGE COMMAND	131
7.2 MODIFYING VIA THE UPDATE COMMAND	131
7.3 HOW TO INCREASE THE EFFICIENCY OF THE CHANGE AND UPDATE COMMANDS	131
7.4 MODIFICATION VIA THE COMPUTE COMMAND	132

CHAPTER 8 - RUNNING MICRC ON MTS	133
8.1 HOW TO RUN MICRC	133
8.2 PREFIX CHARACTERS	133
8.3 ATTENTION INTERRUPTS	134
8.4 COMMAND TIME LIMITS	134
8.5 SECURITY AND DATA INTEGRITY FEATURES OF MICRO	134
8.5.1 LEVELS OF SECURITY	134
8.5.2 MAINTENANCE OF DATA INTEGRITY	135
8.5.3 A CHECKPOINT AND RECOVERY PROCEDURE	135
CHAPTER 9 - THE MICRC COMMAND LANGUAGE	137
9.1 SPECIFYING A COMMAND	137
9.2 NOTATION USED IN THIS MANUAL	137
9.3 SYNONYMS ACCEPTABLE IN MICRC	138
9.4 FORMAT OF COMMAND DESCRIPTIONS	139
9.4.1 PURPOSE SECTION	139
9.4.2 COMMAND SYNONYMS SECTION	139
9.4.3 PROTOTYPE SECTION	139
9.4.4 APPLICABLE COMMAND MODIFIERS SECTION	140
9.4.5 IMPLICIT EFFECT ON THE RESULT SET SECTION	140
9.4.6 WHAT HAPPENS SECTION	140
9.4.7 EXAMPLES SECTION	140
9.4.8 COMMENTS SECTION	140
9.4.9 ERROR MESSAGES SECTION	140
9.4.10 NOTEWORTHY USAGE STRATEGIES SECTION	140
9.4.11 SPECIAL DESIGN CONSIDERATIONS SECTION	140
9.5 COMMAND AND FIELD MODIFIERS	141
9.5.1 RULES FOR COMMAND AND FIELD MODIFIERS	141
9.6 COMMAND DESCRIPTIONS:	
CALL	143
CHANGE	147
COMBINE	151
COMMENT	153
COMPUTE	155
CROSSTABULATE	161
DATESTAMP	167

DESCRIBE	169
DESTROY	171
DISPLAY	173
DOCUMENT	175
END	179
ENTERDATA	181
EXPLAIN	197
FIND	199
FULLDOC	211
GET	215
JOIN	217
KEY	221
LIST	223
MESSAGE	227
MTS	229
NAME	231
PLOT	235
PRINT	237
PURGE	251
READ	253
RELEASE	255
REMOVE	257
REPLACE	259
RESET	263
RESTRICT	265
RESTRICTANDJOIN	269
SAVE	273
SELECT	275
SET	279
SIGNOFF	285
SORT	287
TECHDOC	289
UPDATE	291
USE	305
WRITE	307
 CHAPTER 10 - THE MACRC PROCESSOR	 309
10.1 INTRODUCTION	309
10.2 MACRO LIBRARIES	309
10.3 MACRO DEFINITIONS	309
10.4 MACRO COMMAND	310
10.5 VARIABLE SYMEOLS	310

10.6 COMMENTS IN THE MACRO DEFINITION	311
10.7 SYNONYMS FOR MACRO COMMANDS	311
10.8 WRITING MACRO DESCRIPTIONS/COMMANDS	311
DEFINE	313
DELIMITER LIST	314
END DEFINITION	315
GOTO	316
IF	317
MACRO COMMAND	318
MODEL	319
NOP	320
10.9 MACRO EXAMPLE	321
10.10 MACRO ECHO	322
APPENDIX A - EXECUTED AND ANNOTATED DATA ENTRY EXERCISES	A-1
APPENDIX B - EXECUTED AND ANNOTATED DATA MANAGEMENT, DATA MANIPULATION AND REPORT GENERATION EXERCISES	B-1
APPENDIX C - EXAMPLE CUSTOM DATA ENTRY ASSEMBLY LANGUAGE PROGRAM LISTING	C-1
APPENDIX D - EXAMPLE SUBROUTINE FOR CALL COMMAND	D-1
APPENDIX E - ERROR MESSAGES	E-1
INDEX	I-1

CHAPTER 1

INTRODUCTION

The MICRO Information Management System is a collection of computer programs which permits non-programmers to define, enter, interrogate, manipulate and update user-defined collections of data in a relatively unstructured and unconstrained environment. It has general applicability to a wide variety of educational, administrative and research data processing activities, among others.

MICRO is designed to be run interactively from computer terminals. The system is interactive in that the user issues a command request to the system and waits for the system to execute that request and to respond accordingly before the user issues another command. This is particularly convenient since succeeding command requests are frequently dependent upon the results of the previous requests. MICRO is very powerful in terms of the complexity of the requests which can be made by users without prior programming language experience because the English-like grammar is easy to learn and to use. "Canned" or predetermined procedures can be easily executed in MICRO to deal with complex reporting and retrieval problems.

1.1 OPERATING ENVIRONMENT

MICRO operates only on the Michigan Terminal System (MTS), a timesharing system developed by the Computing Center of The University of Michigan. MTS runs on virtual memory IBM and Amdahl computers at:

The University of Michigan, Ann Arbor; Wayne State University, Detroit; Rensselaer Polytechnic Institute, Troy, N.Y.; The University of Alberta, Canada; The University of British Columbia, Vancouver, Canada; and The University of Newcastle Upon Tyne in England.

All of the computer installations are accessible via regular (voice grade) telephone lines from a variety of computer terminals. At the time of publication of this manual, MICRO is available only at The University of Michigan and Wayne State University. This manual describes MICRO as it operates under MTS at The University of Michigan. Users at other installations should consult their respective computer centers for the details of any deviations from the information in this document.

1.2 HISTORY

Development of MICRO began in 1970 as part of the Labor Market Information System (LMIS) Project at The University of Michigan under Contract 71-24-70-02 from the Office of Research and Development, Manpower Administration, U.S. Department of Labor. Dr. Malcolm S. Cohen was Director of the LMIS Project and was the principal innovator and designer of the original MICRO Retrieval System. Carol Easthope and Jack Guskin were the principal programmers.

In order to reduce developmental costs for the system, MICRO was built upon an experimental data management support package founded in set theory called Set Theoretic Data Structures (STDS). STDS is a proprietary software product of the Set Theoretic Information Systems Corporation, 117 N. First Street, Ann Arbor, Michigan 48108, which has granted permission for its unrestricted use on MTS at The University of Michigan. David Childs, Vice President of STIS Corporation, provided continuing guidance in the advanced use of STDS by MICROC. For further information on STDS, see STDS/I Reference Guide (October 1975) by Set Theoretic Information Systems Corporation or contact STIS Corporation directly.

A "User's Guide to MICRO" (Version 1) by Jack Guskin appeared in the Proceedings of the 1971 Labor Market Information Conference at The University of Michigan. There was no formal documentation of Version 2 of the MICRO Retrieval System. Version 3.0 of MICRO was described in Chapter 5, "The MICRO Data Retrieval Language," of the October 1971 Progress Report On The Feasibility Of A Labor Market Information System. Chapter 5 was written by William Hershey. At this point in time, numerous labor market information experiments were underway using MICRO as an analytical tool.

In 1972, the LMIS Project became part of the Institute of Labor and Industrial Relations (ILIR), a joint institute of The University of Michigan and Wayne State University. Michael A. Kahn assumed a major role in coordinating development, documentation and training for MICRO. Early in 1973, principal programming responsibilities for MICRO were passed to Donald Rumelhart. Version 3.8 of MICRO with numerous user-oriented enhancements was released as was the accompanying MICROC Information Retrieval System (Version 3.8) Reference Manual of February 1973 by Easthope, Kahn and Rumelhart. The usage of the MICRO System in the LMIS Project experiments is described in On the Feasibility of a Labor Market Information System (3 volumes) by Malcolm S. Cohen, Project Director, which is available from ILIR.

In late 1973, additional MICROC development and labor market

MICRO Informatic Management System (Version 5.0) Reference Manual

information system experiments were authorized under Grant 21-16-74-21 from The Office of Research and Development, Manpower Administration, U.S. Department of Labor. Boyd Bronson joined the MICRO programming staff and later reorganized the dictionary creation facility and developed the data entry facility. Two MICRO documents were published: the MICRO Information Retrieval System (Version 3.9) Reference Manual of April 1974 by Kahn, Rumelhart and Bronson and the MICRO Information Retrieval System (Version 3.9) Training Manual by Kahn and Harbison. The use of MICRO in labor market information experiments is described in A Study of On-Line Use of Job Information in Employment Service Local Offices by Malcolm S. Cohen, Project Director.

In 1975, enhancements to MICRO funded by the School of Public Health of The University of Michigan and ILIR, especially in the area of user-oriented data entry and dictionary creation, led to the release of Version 4.1 and its accompanying documentation: MICRO Information Storage and Retrieval System (Version 4.1) Technical Reference Manual of February 1976 by Kahn, Rumelhart and Bronson.

Enhancements funded by the Dean's office of the College of Literature, Science, and the Arts (LS&A) of The University of Michigan led to Version 5.0. This document which greatly expands MICRO documentation, especially into the areas of dictionary definition and data set initialization, was funded by the ILIR, the LS&A College, the Computing and Data Processing Center at Wayne State University, and The University of Michigan.

1.3 ACKNOWLEDGEMENTS

The authors acknowledge the technical and editorial assistance provided by William Hershey, David Hetrick, Malcolm Cohen, Kurt Kovacs, Vaughn Frick, Beverly Kahn, and Sam Listiak. The authors thank them and those who prepared this manual for generation via *PCRMAT: Susan Tuin, Jackie Murray, Stephanie Morrison, and Denise Darcy.

1.4 HOW TO USE THIS MANUAL

This manual is divided into ten chapters and five appendices. Chapter 1 provides background information on MICRO. Chapter 2 defines MICRO concepts and terminology. Chapter 3 describes how to logically view data and data management. Chapter 4 describes the physical constraints that MICRO places on the logical view. Chapter 5 describes the process of data set definition. Chapter 6 describes how data is put into a MICRO data set, while Chapter 7 explains how this data may be modified. Chapter 8 explains how to run MICRO and Chapter 9 describes the MICRO Command Language. Chapter 10

describes MICRO's conditional macro processor. Appendix A is a series of executed and annotated data entry examples including the cost of execution. Appendix B is a series of executed and annotated data management, data manipulation and report generation examples, also including the cost of execution. Appendix C is an assembly program listing of the example of the custom programming method of data entry. Appendix D is an example subroutine written for use with the CALL command. Appendix E is a list and description of the error messages generated in MICRO.

Users who wish only to access existing data sets should read Chapters 1, 2, 8 and 9. If a user wants to enter data then Chapter 6 and Appendix A should be read. If a user wants to modify a data set then Chapter 7 should be read. If a user wants to define a new data set then Chapters 3, 4 and 5 should be read. All users should refer to the examples in Appendix B. Advanced users should read Chapter 10 and Appendices C and D. Appendix E should be referenced when needed.

Learning to use MICRO is a cumulative process. After following any of the just recommended reading sequences, users will most likely find it helpful to return to the concepts presented in the first several chapters.

1.5 COMPOSITION STYLE

In this manual, common English usage is followed by treating "data" as a collective noun. Purist grammarians may prefer "datum" for singular and "data" for plural but, for ease of reading, "data is" will be found herein. Additionally, "he" and "him" are used to mean "he or she" or "him or her". The manual would be difficult to read if the latter were used extensively.

1.6 OTHER RECOMMENDED READING

The following are relevant volumes published by the Computing Center of The University of Michigan:

Introduction to the Computing Center, May 1976

MTS Manual Volume 1: The Michigan Terminal System, April 1976

MTS Manual Volume 2: Public File Descriptions, January 1975

MTS Manual Volume 3: Subroutine and Macro Descriptions,
May 1973

MTS Manual Volume 4: Terminals and Tapes, August 1974

MICRO Information Management System (Version 5.0) Reference Manual

The following publications deal with MIDAS, the Michigan Interactive Data Analysis System, and are published by the Statistical Research Laboratory at The University of Michigan:

Documentation for MIDAS, September 1976

A Manual of Elementary Statistics Using MIDAS, September 1975

MICRO Information Management System (Version 5.0) Reference Manual

CHAPTER 2

CONCEPTS AND TERMINOLOGY

Because the fields of data processing and information management have not yet reached the level of maturity in which terminology is consistently used, the prospective MICRO user must become familiar with the terminology and underlying concepts used in discussing and using the MICRO Information Management System. The actual commands of the MICRO language are described in detail in Chapter 9.

2.1 DATA SETS, RECORDS, FIELDS, VALUES AND CATEGORIES

A data set is a collection of information accessible through MICRO. Consider the analogy of a file cabinet containing a copy of every driver's license issued in the state. The collection of information on driver's licenses is equivalent to a data set in MICRO. A MICRO data set consists of records which have the same format but may contain different information. A blank driver's license is the common record format in the example file cabinet. Each record has a place for the same amount and kinds of information. These common elements which appear in all records of a data set are called fields. Some of the fields on a driver's license are name, address, city, county, birthdate, license number and sex. The information associated with a field or a specific record is called its value. Fields can have a variety of different types of values. Some may be character strings (such as the driver's name). Others may be numerical (such as the driver's height and weight). Some may have predetermined values such as male or female for the sex of the driver. The predetermined values are called categories.

2.2 DATA FILES, DICTIONARIES AND DIRECTORIES

The records of a MICRO data set are stored in a data file in a special format for efficiency called an STDS set. The data file is always an MTS sequential file.

In addition to some user-oriented information about the data set, the dictionary contains information about how the fields of the data set have been stored in an STDS set. The dictionary is stored in an MTS sequential file which is called the dictionary file.

Each data set must have a dictionary file and a data file in order to be used in MICRO. Since a user may have more than one data set, information relating to the dictionary and data files of the

corresponding data sets is stored in an MTS file (which must be named DIRECTORY). Each user who has one or more data sets must have a directory on his Computing Center identification number (CCID) also called a signon-ID. A user may access other users' data sets without having a directory, if the owner permits the necessary files to allow this. (See GET command.)

2.3 DESCRIPTIONS AND NAMES

In order to make information management an easier task for the non-technical user, there are provisions in MICRO for user-supplied descriptions of all fields and categories to be stored in the dictionary. Additionally, the description of each data set may be recorded in the directory. These descriptions may be referenced interactively whenever the user is unsure of the purpose (definition) of a data set, field or category. Category descriptions may also be printed in computer generated reports.

Each data set must have a user-supplied name of up to eleven characters in length. This name is called the data set name. Each field must have a user-supplied name of up to 16 characters in length called the field name. Each field may also have a field abbreviation of up to four characters in length which is a synonym for the field name. Each category must have a user-supplied name of up to 11 characters in length called the category name and may have a synonym (up to four characters) called the category abbreviation.

2.4 PERMANENT AND TEMPORARY SETS AND THE RESULT SET

There are two different time frames in which a data set may exist: permanent and temporary. Permanent data sets are those which exist from one MICRO session until the next and which have their dictionary and data stored in permanent MTS disk files or on magnetic tape. Permanent data sets on disk can be destroyed or may have the data file replaced through MICRO commands if the data set was defined to permit these activities at the time of dictionary definition. Temporary data sets are created during a MICRO session as the result of various commands. Permanent disk storage is not associated with temporary data sets. Temporary data sets are not carried over from one MICRO session to the next. A temporary data set can be made into a permanent data set. Data sets residing on tape are a special case and are discussed in Section 5.11.

Almost all of MICRO's commands involving manipulation of data in permanent or temporary data sets do not physically change the data set during execution. Instead, the result of the command (be it a query, change, tabulation, etc.) is placed into a temporary data set named RESULT. For example, if the user issues a MICRO

command to find all of the freshmen in a data set named STUDENTS which contains records on all students, then the records on the freshman students would be in a set named RESULT and the STUDENTS data set would remain unchanged and still contain the records on all students. The placing of the records resulting from a MICRO operation in a set named RESULT is the result set concept. Reference to the "result set" or the "result data set" in this manual is the same as referring to the data set named RESULT. For the purpose of convenience, the data set named RESULT always has a synonym named IT; both names refer to the same result data set.

The result set is a temporary data set but it has one property unlike other temporary data sets: a named temporary data set exists until the end of the MICRO session (if it is not intentionally destroyed, renamed, etc.) whereas the existing result set is released when a new result set is created by a MICRO command. In other words, there may be only one result set and an existing result set is released when a new one is created. The result set can become a named temporary data set to prevent its subsequent loss. The effect of each command upon the result set is described in the section on command descriptions.

2.5 ORDERING OF RECORDS IN A DATA SET

The records of a data set are kept sorted (ordered) at all times. The sorting and the associated rules are important to the MICRO user. Each record of a MICRO data set has the same number of fields and, therefore, all records are of the same size (occupy the same amount of storage space). The data for each field is located in the same position in each record. These same-length records are sorted on the entire length (across the data values in each field of each record). After completing this sorting process, records which are logically less than others would be at the front of the data set and those which are logically greater would be sorted to the end of the data set. This ordering scheme is called ascending order. Identical records would normally be next to each other. However, in all MICRO data sets, all identical records are reduced to just one occurrence. This process of eliminating redundant records is called weeding.

All MICRO data sets are sorted and weeded. If the user wishes to maintain duplicate records, it is necessary to make each occurrence unique (and, therefore, no longer a duplicate record). A unique key may be appended to each record to insure that no two records are identical. (See KEY command.) a crosstabulated data set is another way that redundant data can be represented and is discussed in the next section.

2.6 CROSSTABULATED DATA SETS

A "crosstabulated" (or "crosstabbed," "xtabbed," or "XTAB") data set may be thought of as aggregated or summarized data, i.e., each record of an XTAB set represents one or more records of a larger collection of data. This larger collection of data, referred to as the "original data set," may itself be a MICRO data set. Each record in an XTAB set will include a field containing the number of original data set records represented (sometimes referred to as the "weight" of the record). If the XTAB set is generated by the CROSSTABULATE command then this field is called the COUNT field and its abbreviation is CNT.

The value of the COUNT field indicates the number of records in the original data set which have the same combination of values for the fields specified in the CROSSTABULATE command. Thus the sum of the values of the COUNT field is the total number of records represented by the crosstabulated set, i.e., the cardinality of the original data set. The fraction of original data set records represented by a record in the XTAB set is equal to the COUNT field divided by the cardinality of the original data set. A special field named PERCENT, which is optionally created as part of the CROSSTABULATE command, represents this fraction. Further description of the COUNT and PERCENT fields follows an example of a crosstabulated set.

2.6.1 EXAMPLE OF CROSSTABULATED DATA SET

Suppose that MICRO is being used to maintain some information on university students (as in the example data set used later in this manual). One record exists for each student which contains such demographic information as race and sex as well as a unique identification number. The CROSSTABULATE command could be used to reduce this set to show the number (and percentage) of students in each unique combination of race and sex. Thus the reduced (XTAB) data set consists of the fields RACE, SEX, COUNT and PERCENT. Whereas there may have been many thousand records (one for each student) in the original set, the resulting XTAB set may contain only a few records representing all of the unique combinations of race and sex that existed in the original data set. See the CROSSTABULATE command description and Appendix A for more examples.

It is possible to start with data already in a crosstabulated form. Much census data is provided in this manner. Rather than release confidential data at the individual family level, the level at which the data is collected (which is often called "micro data" and is where the MICRO System got its name), the Census Bureau makes data available at different levels of aggregation: Census tract,

SMSA, state, etc. The population count associated with each unique combination of demographic characteristics can be stored as the COUNT field of an XTAB set.

2.6.2 THE COUNT FIELD

The field of a crosstabulated data set which indicates the representative value or weight associated with each record is named COUNT if generated by the CROSSTABULATE command but may be renamed using the NAME command. The COUNT field is an unsigned integer that is four bytes in length. It always begins on a fullword boundary (that is, the first, fifth, ninth, etc. byte of the data set) and one to three extra bytes of zeros may be placed in front of the COUNT field to force fullword alignment. This alignment is completely transparent to the user and becomes visible to the user only in dictionary documentation.

2.6.3 THE PERCENT FIELD

A crosstabulated set may have a field named PERCENT whose abbreviation is %. Both PERCENT and % should be treated as reserved names.

The PERCENT field of each record in an XTAB data set represents the fraction derived by dividing the COUNT field of that record by the cardinality of the data set and then multiplying it by 10000. The PERCENT field, like the COUNT field, is an unsigned integer that is four bytes in length and must be fullword aligned. As previously described, alignment may be forced by adding extra bytes of zeros in front of the PERCENT field. In addition, it is scaled by division by 100 so that the result of the above calculation appears in xxx.xx format.

2.7 BUILDING AND USING MICRO DATA SETS

The roles of a data set definer (one who makes a data set available for others to use) and a data set user are interrelated. In many cases, the definer and user are one and the same. While it is mandatory for the definer to understand the process for building and using MICRO data sets, it is highly desirable for the user to be familiar with the steps of this process. In that way, the definer and user can better discuss data management problems.

Figure I on the following page graphically displays in flowchart form the MICRO data set building and usage process. This flowchart is a simplification of the process but is sufficient for general user understanding. Each box is briefly discussed so that the user can better understand what happens at each step in the

process. These steps are discussed in detail in Chapters 3 through 7.

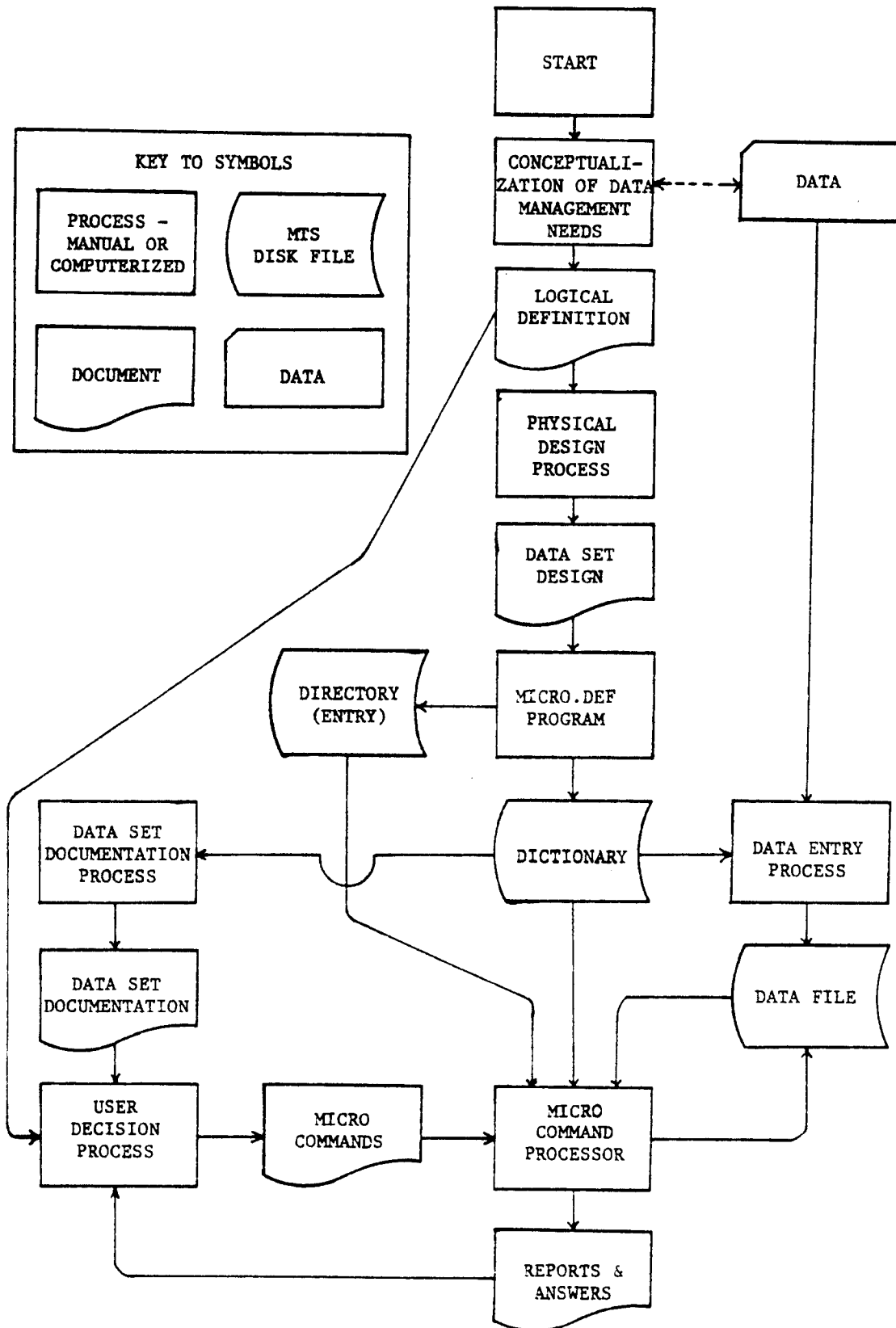


FIGURE 1: MICRO DATA SET BUILDING AND USAGE PROCESS

2.7.1 CONCEPTUALIZATION OF DATA MANAGEMENT NEEDS

The definer must decide which data management functions are needed. This analysis may influence the data to be generated or existing data may affect the analysis. Each data item needed must be studied as well as the interrelationships between the items.

2.7.2 LOGICAL DEFINITION

The results of the conceptualization of data management needs are: (1) a logical definition of the data items and their interrelationships and (2) a set of questions that must be answered or data manipulations that must be made.

2.7.3 PHYSICAL DESIGN PROCESS

In the physical design process, the facilities and constraints of the MICRO system are applied to the logical definition.

2.7.4 DATA SET DESIGN

The result of the physical design process is the data set design which is the data representation in MICRO.

2.7.5 MICRO.DEF PROGRAM

The data set design is used as input to the MICRO.DEF program which creates a dictionary and an entry in the directory for the newly defined data set.

2.7.6 DICTIONARY

The dictionary contains the description (physical properties and user-accessable names) of a data set.

2.7.7 DIRECTORY

The directory entry links the dictionary file and the data file with the corresponding data set.

2.7.8 DATA

The data may have been generated as a result of the conceptualization process or may have preceeded it.

2.7.9 DATA ENTRY PROCESS

There are several methods for putting data into a MICRO data

set. The dictionary information and the data are required for all of these.

2.7.10 DATA FILE

The data file is created as a result of any of the data entry processes. The data is now in a form accessible by MICRO.

2.7.11 DATA SET DOCUMENTATION PROCESS

There are several facilities for data set documentation. All use the dictionary as input and produce an output called data set documentation.

2.7.12 DATA SET DOCUMENTATION

The data set documentation is a complete description of a data set, its fields, their categories, etc.

2.7.13 USER DECISION PROCESS

Questions and data manipulation needs determined at the time of conceptualization are transformed into MICRO commands.

2.7.14 MICRO COMMANDS

The MICRO commands must be specified according to the allowable grammar described in Chapter 9.

2.7.15 MICRO COMMAND PROCESSOR

The MICRO command processor scans the user's MICRO commands to determine which data set is being referenced and (via the directory) acquires the needed dictionary and data files. The commands are processed followed by the generation of answers and/or reports along with the data file being updated as specified.

2.7.16 REPORTS AND ANSWERS

The reports and answers are processed by the user. Further questioning and/or data manipulation may then be required.

2.8 PROGRAMS IN MICRO

Six computer programs are available to the MICRO user/definer:

- (1) MICRO;
- (2) MICRC.DEF;

- (3) MICRO.DIR;
- (4) MICRO.DOC;
- (5) MICRO.ADD; and
- (6) MICRO.UP.

2.8.1 MICRO

MICRO is the command language processor which is accessed for query, data manipulation, etc. It is stored in the file ILIR:MICRO and is discussed in Chapters 8 and 9. Four features are associated with the MICRO command language processor:

- (1) Master Signon-ID;
- (2) Command Trace Facility;
- (3) Runfile; and
- (4) Conditional Macro Processor.

(1) MASTER SIGNON-ID

The master signon-ID is a user's signon-ID on which several key files are stored. If not specified, the default master signon-ID is ILIR. The master signon-ID is also discussed in Section 5.4.2.

(2) COMMAND TRACE FACILITY

The command trace facility allows MICRO users to optionally record all commands entered in MICRO. This may be for archival purposes or for recovery procedures. The command trace output is stored in files discussed in Section 5.4 and is in a format that can easily be read into a MICRO data set consisting of the following fields:

DATE
TIME
SIGNON-ID
COMMAND NAME
COMMAND

(3) RUNFILE

The runfile permits predefined commands to be executed once the user has entered MICRO command mode. This facility is further described in Section 5.4.3.

(4) CONDITIONAL MACRO PROCESSOR

The conditional macro processor is a subsystem of the MICRO

command processor and permits a series of MICRO commands to be named and conditionally executed. Chapter 10 describes this advanced-user feature.

2.8.2 MICRO.DEF

MICRO.DEF is the program by which the definer defines a data set. It is stored in the file ILIR:MICRO.DEF. The definition process is discussed in Chapters 3, 4, and 5.

2.8.3 MICRO.DIR

MICRO.DIR is the program for creating and deleting directory entries. It is stored in the file ILIR:MICRO.DIR and is discussed in Section 5.9.

2.8.4 MICRO.DOC

MICRO.DOC is a stand-alone program which generates data set (dictionary) documentation. This program is stored in the file ILIR:MICRO.DOC and is discussed in Section 5.10. It is also accessible through the DOCUMENT, TECHDOC and FULLDOC commands which are discussed in Chapter 9.

2.8.5 MICRO.ADD

MICRO.ADD is a stand-alone version of the ENTERDATA command and is discussed in the ENTERDATA command description in Chapter 9. The stand-alone version program is stored in the file ILIR:MICRO.ADD.

2.8.6 MICRO.UP

MICRO.UP is a stand-alone version of the UPDATE command and is described in the UPDATE command description in Chapter 9. The stand-alone program is stored in the file ILIR:MICRO.UP.

CHAPTER 3

CONCEPTUALIZATION OF DATA MANAGEMENT NEEDS - THE LOGICAL DEFINITION

In order for a potential user of any data management system to successfully meet his data management needs, he must first define these needs and the involved data in implementation-independent terms. This activity is called logical definition and is discussed in this chapter. In general terms, this activity consists of determining what data is available and what data manipulation and/or report generation is desired.

The mapping of the logical definition into the constraints and facilities of a specific data management system is called physical design. The physical design of MICRO data sets is discussed in Chapter 4.

In this manual a "user" is one who works with an existing (or defined) data set. A "definer" is one who creates (defines) a data set. In some cases, the user and definer are the same person.

There are five conceptual activities of logical definition:

- (1) Abstraction of data;
- (2) Initialization of a data set;
- (3) Retrieval of information from a data set;
- (4) Modification of a data set; and
- (5) Integration of data sets.

Many commands and processes in MICRO facilitate these conceptual activities.

3.1 ABSTRACTION OF DATA

Certain abstract characteristics can be associated with any collection of data (i.e., a data set) and with the data elements (fields) within that collection. The definer must logically define (i.e., describe) the data set and its fields independent of the facilities of the data management software that are to be used.

3.1.1 ABSTRACTION OF DATA COLLECTIONS

First, the definer must determine what is represented by a collection of data, that is, the definer must be able to clearly

describe it. The formality of the definition depends upon the users and the data collection.

Second, the definer must determine whether any user should be able to destroy this collection of data, that is, to remove it from the computer.

Third and independent of the "destroyability" of a collection of data, the definer must determine whether the collection can be updated, that is, whether permanent modifications can be made.

Fourth, the definer must determine whether the data is aggregated data. For example, printed tables of Census data are aggregated summaries of all of the individual Census forms.

Fifth, the definer must determine whether any or all of the data is sensitive and/or confidential.

3.1.2 ABSTRACTION OF DATA ELEMENTS (FIELDS)

Data can be classified according to several overlapping criteria and this process must be carried out as part of data set design. The definer must know what kinds of data are to be included in the data set.

(1) NATURAL AND CODIFIED DATA

At the user level, data for a field can be either codified or natural. Natural data is data which is stored in the computer in the same form in which it is represented to the user. Examples of natural data are numbers (such as one's age) and character strings (such as one's name). Codified data is stored in the computer in a form other than that seen by the user. Consider the example of a field containing information on the sex of individuals. The represented values to the user are usually "male" and "female" but the value stored for "male" in the computer may be "1" and the value for "female" may be "2". Or the values may be "M" or "A" for "male" or "F" or "B" for "female". The values stored may be arbitrary as long as they are consistent. In some cases, the value becomes what it represents as in the various months of the year ("1" for "January", etc). With codified data, the user is not as much concerned with the values stored as with what they represent (or are called).

The user must identify which data is or could be "codified" and which data is "natural." Additionally, the user should

identify the categories or codified states of each codified field. From the user's point of view, all data is stored as numbers or characters. The definer must determine if the stored numerical and character values are to be treated as if they are codified or natural.

(2) NUMERICAL AND CHARACTER DATA

In addition to being classified as natural or codified, data for a field can also be classified according to the kind of data stored, either numbers or characters. Numbers may be integers, real (floating point) or decimal and may also be negative or positive. The user must classify the numerical fields accordingly. Numbers can also be treated as character strings.

Character strings may be fixed or variable in length (the number of characters) and may have a maximum length. Character string fields must also be classified.

(3) REPEATING GROUPS

One additional abstraction that the user should consider is that of the "repeating group." A repeating group is a field which may have more than one value at the same time, that is, one value is not exclusive of all other values. For example, a student may receive financial aid from more than one source at the same time such as "Scholarship", "College Tuition Grant" and "City Bank Loan". In this example, all three sources of financial aid are simultaneously associated with a single field called "SOURCE-OF-AID".

3.2 INITIALIZATION OF DATA SETS

The process of putting the data into a new data set is called initialization. The user who wishes to design a data set must consider the form in which the data will be available for initialization. The first question about the form of the data is whether the data already exists in a fixed-format, computer processable form such as punched computer cards, magnetic tape, disk file, etc. Fixed-format implies that data for each field is always in the same column(s) in each record.

If the data doesn't exist in machine readable form then the user must have answers for three questions:

- (1) How many records are to be entered initially into each data

set?

- (2) How often and how many records are to be entered after the initial entry of data?
- (3) What staff resources are available to perform data entry? Are there skilled keypunch operators, non-technical clerical staff, systems professionals, academic reseachers, etc., available?

3.3 RETRIEVAL OF INFORMATION FROM DATA SETS

It is important for the user to think about the types of retrieval activities that will most likely be performed on the new data set. This determination will not preclude other retrieval activities but can affect the efficiency of the resulting data set design. The user needs to think about:

- (1) Whether searches will be performed for records meeting specific criteria, whether these searches are likely to be numerical or character data.
- (2) Whether data reduction or information summarizing will be performed on records in all or part of the data set.
- (3) Whether reports will be generated from the data stored in the data set.
- (4) Whether the user is likely to want to make selective changes in the data set (especially arithmetic changes) in order to study the effect of these changes. An example of this would be to change CENSUS data to reflect hypothetical changes in income, population, etc.
- (5) Whether there are any application specific data processing activities that will be required.

3.4 MODIFICATION OF A DATA SET

Modification of a data set can occur in several ways. The definer should determine which of the following are likely to occur:

- (1) Modification by the deletion of records;
- (2) Modification by the addition of new records;
- (3) Modification by the addition or deletion of fields from a data set;

- (4) Modification by replacing numerical values;
- (5) Modification by replacing character string values; and/or
- (6) Arithmetic modification of numerical values.

These modifications may be temporary or permanent and (4) through (6) may be performed on selected records of a data set.

3.5 INTEGRATION OF DATA SETS

Sometimes the user may want to take related information from two or more data sets and join them together into one integrated data set. The data need not be from the same source but must be related through one or more fields which make up the "key" to integrating two data sets. These fields are sometimes called the "key fields." Social Security numbers and student identification numbers are examples of a single field which may be the key field of a data set. The University Registrar's Office uses a typical example of a key consisting of multiple fields in describing the various classes that are offered: division (or department) number, course number and section number. Consider the following example:

If the user has a data set on students characteristics of which the student identification number (SIN) is a field and if the user has another data set on the financial aid awarded to students which also includes the SIN, then the user may create an integrated set consisting of the SIN, the characteristic data and the financial aid data.

The definer should determine if integration of data sets (inter-data set activity) is likely.

3.6 OVERVIEW OF EXAMPLE

Throughout this reference manual one example information management problem is used to demonstrate the capabilities and techniques for using MICRO. This example is based on a fictitious university called Intercollegiate University (ICU). The fictitious information management problem described is only part of the total information management requirements of a university but it is sufficient as an example.

ICU must keep track of all its students. The students will graduate in different years, are of different sexes and races, have different birthdays, and of course, different names. In addition, each student has been assigned a unique identification number. Some of the students have financial aid; some do not. For the ones who

do have financial aid, the university must maintain information on the amount of aid, the source of funds, the type of aid (whether for tuition, room and board, etc.), the class level of the recipient, the year and term in which the aid was granted, and any special remarks about the financial aid.

This collection of data can be characterized by considering the five points mentioned in Section 3.1.1. First, the data can be described by saying that it is personal information about students enrolled at Intercollegiate University and about the financial aid that some of them receive. Second, the university does not want it to be destroyed because it is important information. The information will change from year to year, so the third determination is to allow it to be updated. Fourth, the data is disaggregated; that is, we can retrieve specific information about each student. (In aggregated or summarized data, the details about each student would be lost.) Fifth, the university wants the information to remain strictly confidential to maintain the privacy of its students.

Having characterized the collection of data, it is necessary to look at the individual data elements. The class year is natural data, and it is numerical. It is simply the last two digits of the year in which the student will graduate. Sex and race are codified data elements. The university uses M and F to represent males and females, so the sex data element consists of character data. ICU has chosen to represent race, on the other hand, by numerical data. Each ethnic group is assigned a different 1-digit number. Birthday information is natural, numerical data that consists of numbers for the month, day, and year. Names of students are in their natural, character form. Student identification numbers are natural and numerical. As for the data pertaining to financial aid, the amount of aid is represented by natural, numerical data. The source of funds, type of aid, class level of the recipient, and term in which the aid was first granted are all codified, numerical data elements. The year in which aid was granted is natural and numerical data. Remarks are natural and in character form, but the length of these remarks may vary.

The Intercollegiate University Administrative Data Center is an organized place, so they keep all data pertaining to students in a computer processable form on disk files. All fields are in a fixed-format except for the remarks. Because data must be entered for new students, Data Center teletype operators have been trained to enter small numbers of records by reading information from student application forms. Since mistakes can be made and individual records must sometimes be changed, the Data Center teletype operators must also know how to delete records and replace

numerical and character string values. Since the administrators of Intercollegiate University frequently change their minds about what information is needed about students, the Data Center systems professionals know that they must be able to add or delete fields from a data set and modify numerical values arithmetically. The retrieval capabilities must be extensive in order to handle all the different information requests of the ICU administrators.

CHAPTER 4

TRANSFORMING THE LOGICAL DEFINITION INTO A PHYSICAL DESIGN

The questions answered in Chapter 3 about the logical definition are the basis for the physical design of a MICRO data set. Logical definition can and should be undertaken without regard to MICRO's physical constraints. The physical design process is discussed in this chapter and is the step preceding the actual "creation" of the MICRO data set. There are physical design issues which deal with both the aggregate collection of data (data set) and with each field.

The information which is required by MICRO to define a data set is discussed in this chapter. The global specifications pertaining to the data set as a whole are discussed first, followed by the specification of field and category definition. The physical design issues presented in this chapter will prepare the definer to perform dictionary definition and modification as discussed in Chapter 5. The Intercollegiate University example is continued.

4.1 AGGREGATE DATA SET REQUIREMENTS

There are pieces of information which pertain to the data set as a whole (as opposed to each field of the data set). These are:

- (1) Data set name;
- (2) Data set description;
- (3) Data set destruction status;
- (4) Data set replacement status;
- (5) Data file name;
- (6) Dictionary file name;
- (7) Conversion file name;
- (8) Crosstabulated data set indicator;
- (9) Scrambled data set indicator; and
- (10) Dictionary use count.

Each will be discussed separately.

4.1.1 DATA SET NAME

Each collection of data must be given a name (called the data set name) which may be up to eleven characters in length. The definer should attempt to choose a name which will describe the data set to the user audience.

4.1.2 DATA SET DESCRIPTION

For each data set, a description of up to 242 characters in length can be stored. The date of creation and the creating user's signon-ID will be appended to the user-supplied description in the form "MM-DD-YY/CCID". The definer might include in the description any of the following: (1) the version of the data; (2) the creating user's name; or (3) the proposed use or purpose of the data set.

4.1.3 DATA SET DESTRUCTION STATUS

Any user may attempt to destroy a MICRO data set (that is, its entry in directory file, its data file and its dictionary file). However, there are two things to prevent a user from doing so. First, MICRC will not attempt to destroy a data set unless the data set was defined as "destroyable" through the definition subsystem. If the data set is not so defined and the user attempts to destroy it, then MICRO will refuse.

Second, and regardless of the data set destruction status, the "permit status" of the MTS files associated with each data set (dictionary, data, directory and external data files) is not changed or monitored by MICRO. The permit status of these files may be changed by the user who "owns" the signon-ID on which these files are stored. Therefore, MICRC's attempts to destroy a file can be prevented by the MTS file system.

4.1.4 DATA SET REPLACEMENT STATUS

The data set may be defined to be "replaceable," which means that the contents of the data file associated with a data set can be replaced with other data. The new data must be identically structured as it uses the same dictionary as the old data. Replacement commonly occurs after the data has been modified. Attempts to replace a data set will go through the same two-tiered protection process as attempts to destroy a data set. (See Section 4.1.3.)

4.1.5 DATA FILE NAME

The definer needs to specify the name of a file where the data associated with the data set name is to be stored. It does not have to exist at the time of definition, but when it is created, it must be a sequential file. The name may be up to twelve characters (an MTS limitation) and any characters acceptable to MTS for a file name may be used. If the definer wishes to specify a file on another CCID then the maximum length is 17, including the signon-ID and a colon.

There are some time-honored but unenforced MICRO practices with respect to file names which are based on the behavior of MICRO's SAVE command. It has become common practice to name the MTS data file the same as the data set name. The data file name should not be longer than eleven characters because that is the longest name that can be assigned through the NAME command and that can be assigned via the SAVE command.

It is possible for a data set to reside on tape instead of on a disk file. In that case, four asterisks (****) should be used as the data set name which indicates that the data set resides on tape. (See Section 5.11.)

4.1.6 DICTIONARY FILE NAME

The dictionary file name also follows the MTS limitations for length and acceptable characters. It also must be a sequential file, and need not be created prior to definition. Time-honored practice dictates that the dictionary file name be the same as the data set name with a pound or sharp sign (#) appended. Therefore, the file name may be up to 12 characters, including the #. This practice is followed by any data set created through the MICRO SAVE command. If the definer wishes to specify a file on another CCID then the maximum length is 17, including the signon-ID and a colon.

4.1.7 CONVERSION FILE NAME

The conversion file is to be used with the user-specified batch fixed-format method of data entry described in Section 6.3. The conversion file name may be up to 12 characters in length if the optional signon-ID is not specified. If the signon-ID is specified, the maximum length is 17 characters.

4.1.8 CROSSTABULATED DATA SET INDICATOR

If aggregated data is defined and one of the fields will represent a count or weight then the data set will be crosstabulated. Otherwise it is not. See the discussion on crosstabulated data sets in Chapter 2.

4.1.9 SCRAMBLED DATA SET INDICATOR

If the data set will contain sensitive and/or confidential data then the data set may be defined to be "scrambled." That means that the data is stored in the computer in a manner which makes it unreadable and that it is "scrambled" and "unscrambled" according to a user-supplied password (string of characters). The password is

not permanently stored anywhere in the computer and must be remembered by the user.

4.1.10 DICTIONARY USECOUNT

The definer must know how many data sets will be sharing the same dictionary. The usecount is automatically decremented every time one of the data sets sharing the dictionary is destroyed by the MICRO DESTROY command (see Chapter 9). When the usecount reaches zero, an attempt will be made to destroy the dictionary file. (Its success depends on the MTS permit status as described in Section 4.1.3.)

4.2 DEFINING EACH FIELD

Each field must have certain information defined for its inclusion in a MICRO data set. This information is discussed in Sections 4.2.1 through 4.4.18. Advanced design considerations affecting the field definitions are discussed in Section 4.5.

4.2.1 FIELD NAMES AND ABBREVIATIONS

Each field must have a user-defined name and may have a user-defined abbreviation. The name must be no more than sixteen characters in length. The definer may also define an abbreviation or short form of the field name up to four characters in length. If the definer does not choose to define an abbreviation then the first four characters of the field name will be used as the default abbreviation. Thus, if the field name is less than five characters then the default abbreviation is the same as the field name. Field names and abbreviations may not duplicate any other field or category names or abbreviations in the same dictionary.

User/definers planning to use MIDAS in conjunction with their MICRO data sets should consult MIDAS documentation for the characters not permissible in a MIDAS variable name. These characters should not be used in MICRO field abbreviations because they are passed to MIDAS.

Because external data fields (introduced in Section 4.4.7) are treated specially by many of the commands, it may be useful to name them in a way in which users will immediately know that it is an external data field. In the examples in this manual, an asterisk is appended to the field name and abbreviation of the external data field.

4.2.2 FIELD DESCRIPTION

Each field must have a description. The description may be up to 256 characters in length. If the definer does not provide a description at the time of data definition (see Chapter 5) then the field name will be used as the description.

4.2.3 SELECTION OF FIELD TYPE

The user must select one of MICRO's seven field types for each field of data. Each field type is entered, stored, and printed differently. Four numerical field types are introduced in Section 4.3 and three character field types are introduced in Section 4.4.

4.3 NUMERICAL DATA

In MICRO, numbers are usually stored only as binary integers; however, they may be stored as non-numerical character strings. Real and decimal numbers are not supported. However, decimal numbers may be represented as integers. Real numbers may also be represented as integers to the extent that they may be represented as decimal numbers. The representation of decimal numbers is accomplished through "scaling" and is discussed later in Section 4.3.7.

There are four types of fields which deal with stored numbers.

- (1) Type U - unsigned integer field (without categories);
- (2) Type S - signed integer field (without categories);
- (3) Type UC - unsigned integer field with categories; and
- (4) Type SC - signed integer field with categories.

The latter two deal with codified data and are discussed later in Section 4.4.10 on codified data.

4.3.1 TYPE U - UNSIGNED INTEGER FIELD WITHOUT CATEGORIES

Type U is for those integer fields which will not have any negative numbers for values and whose values are natural as opposed to codified (categorical). Type U fields are converted automatically by MICRO.DEF into type UC after definition by the addition of categories.

4.3.2 TYPE S - SIGNED INTEGER FIELD WITHOUT CATEGORIES

Type S is like type U except that signed (negative) numbers are permissible values.

4.3.3 TYPE UC - UNSIGNED INTEGER FIELD WITH CATEGORIES

Type UC is for those integer fields which do not have any negative numbers for values and whose values are codified; i.e., there are categories representing some or all of the values of the field. After definition, type UC fields can be converted to type U with all of the associated categories being dropped or can be displayed as values without having the values transformed into their respective categories.

4.3.4 TYPE SC - SIGNED INTEGER FIELD WITH CATEGORIES

The similarities and differences between type S fields and SC fields are analogous to those between type U fields and UC fields.

4.3.5 FIELD SIZE OF INTEGER FIELDS

The size of the field determines how large a value can be stored in it. The size is also called length or width. The unit of size is a byte which represents 8 binary digits or bits. Integer fields may be one, two, three or four bytes in length. The minimum and maximum numbers that can be represented by the storage associated with integer fields are:

<u>Unsigned Integers</u>	<u>Minimum</u>	<u>Maximum</u>
1 byte	0	255
2 bytes	0	65,535
3 bytes	0	16,777,215
4 bytes(fullword)	0	2,147,483,647
<u>Signed Integers</u>	<u>Minimum</u>	<u>Maximum</u>
1 byte	-128	127
2 bytes	-32,768	32,767
3 bytes	-8,388,608	8,388,607
4 bytes(fullword)	-2,147,483,648	2,147,483,647

It is usually wise to plan to use the smallest amount of space required to represent the anticipated value range because allocated but unnecessary space decreases the efficiency of MICRO. There are exceptions:

- (1) If the range of the field is likely to increase. This can occur as a result of natural growth; i.e., newer data may

have larger numbers than older data, or it may occur when the field is changed arithmetically or as the result of scaling. (See Section 4.3.5 (3)) When the value of a field exceeds the storage allocated to the field a phenomenon called "overflow" occurs.

- (2) If the field is to serve as part of a key for integration of two data sets. Fields to be matched must be the same size. If one is oversized then so must be the other.
- (3) If the field is scaled then the size decision must be based on the internal representation of the number (i.e., without the scaling factor and function applied). For example, 700.00 is stored as 70000 and requires two bytes.

4.3.6 THE DIFFERENCES BETWEEN SIGNED AND UNSIGNED INTEGERS

There are two differences between signed and unsigned integers. First, the values that can be stored in one, two or three byte integers are two times larger than the values that can be stored in a correspondingly sized signed integer. This is because the leftmost bit is used as a sign indicator for signed integers.

Second, the input and output of signed fields are affected. Signed values are not acceptable for input (via the ENTERDATA command) into unsigned integer fields. Also, the output of signed fields includes an additional space for the negative sign.

4.3.7 SCALING

Scaling permits a field's signed or unsigned integer values to be represented as other numbers. Like coding from value to category and decoding from category to value, scaling takes place unknown to the user; that is, the user sees only the scaled value. Data for a scaled field is stored without scaling. When the user wants to access, print or change the values of a scaled field, the appropriate arithmetic operation to scale the field is performed. Scaling by multiplication or division by a power of ten and scaling by addition or subtraction of a positive or negative non-zero integer are permitted.

Scaling is commonly used in order to represent decimal and real numbers and to economically store large numbers.

4.3.8 REPRESENTATION OF DECIMAL NUMBERS - SCALING BY DIVISION

The field types just described did not include real (floating-point) or decimal numbers. The only internal numerical

format is binary (integers). However, decimal values may be represented in the existing integer field types, with some limitations. Real numbers can be handled only to the extent that they can be expressed as decimal numbers which can be expressed as integers. The scaling factor must be a multiple of 10.

If dollars and cents are to be represented in an integer field then the field should be scaled with a scaling function of division and a scaling factor of 100. Thus, the amount \$12345.67 is stored as the integer 1234567 and is automatically divided by 100 so that the user sees only the 12345.67.

4.3.9 REPRESENTATION OF LARGE NUMBERS - SCALING BY MULTIPLICATION

In many circumstances, it is common to round-off numbers to the nearest million of dollars (as in some financial reports) or to the nearest thousand of population (as in Census data). These and other rounded numbers may be expressed as scaled integers where the scaling function is multiplication. Numbers rounded to the nearest thousand could be stored without the last three zeros and would be multiplied by 1000 if the scaling function was multiplication and the scaling factor was 1000. The scaling factor must be a multiple of ten. (As an alternative method of dealing with large numbers, one could leave the field unscaled and simply store small values in the field which represent thousands of people or millions of dollars. In this way the overflow problems discussed earlier in this chapter are avoided.)

4.3.10 SCALING BY ADDITION AND SUBTRACTION

Scaling by addition and subtraction is similar to scaling by multiplication and division. Positive numbers may be stored as negative numbers and vice versa.

4.3.11 SCALING OVERFLOW

No test is made to see if the resulting value from a scaled operation has overflowed. Caution should be taken, as described in the COMPUTE command description in Chapter 9.

4.4 CHARACTER DATA

Character strings of data may be stored in three ways in MICRO:

- (1) As fixed-length character data;
- (2) As external data; and

- (3) As codified data (transformed into categories).

4.4.1 FIXED-LENGTH CHARACTER FIELDS

There are two MICRO field types that deal with fixed-length character fields:

- (1) Type C - character string field (without categories); and
- (2) Type CC - character string field with categories.

Fixed-length means that the same number of bytes (characters) of space is allocated in each record for a particular field. Character fields may be overlapped. This advanced technique is discussed in Section 4.5.3.

4.4.2 TYPE C - CHARACTER STRING FIELDS WITHOUT CATEGORIES

Type C is for any character string stored in its natural form. Only those type C fields which are less than or equal to four bytes (characters) in length may be converted to type CC at some point after definition. Type C fields may be from 1 to 255 characters in length.

4.4.3 TYPE CC - CHARACTER STRING FIELDS WITH CATEGORIES

Type CC is for those character string fields which are codified (i.e., there are categories representing some or all of the values of the field). Type CC fields can only be from one to four characters in length. After definition, type CC fields may be converted to type C with all of the associated categories being dropped, or they may be displayed as values with the character strings being transformed into their respective categories. Codified fields are discussed later in Sections 4.4.10 and 4.4.11.

4.4.4 FIELD SIZE OF CHARACTER STRING FIELDS

Type C fields may be from 1 to 255 characters (bytes) in length. Type CC fields may only be from one to four bytes in length. Character string values which are smaller than the field size will be stored left-justified and padded on the right end with blanks to that field size.

A character field may be completely overlapped by one or more character fields. This advanced technique is discussed in Section 4.5.3.

4.4.5 CONSTRAINTS ON FIXED-LENGTH CHARACTER STRING FIELDS AND SOLUTIONS

There are four constraints associated with character string fields:

- (1) The field may not be greater in length than 255. Strings longer than that must be external (type E).
- (2) The specification of a large field size in order to accommodate a small percentage of long character strings significantly increases overhead. In this case perhaps external representation (type E) may be more appropriate.
- (3) The use of character strings in the CHANGE, FIND and REMOVE commands cannot exceed 24 characters in length. In other words, searches and changes can only be performed on the first 24 characters of fields defined to be greater than 24 in length. After reading Section 4.4.6 on external data fields, the definer will find that the decision between a field containing character strings with more than 24 characters and an external data field is a close tradeoff in most cases.
- (4) Strings of digits stored as characters cannot be modified arithmetically and cannot be compared to integer fields. Additionally, numbers stored as characters usually occupy significantly more space than numbers stored as binary integers. It is better to store numbers as integers than as characters.

4.4.6 VARYING LENGTH CHARACTER FIELDS

Character strings of varying length not greater than 255 may be stored as type C (fixed-length character string) within constraints (2) and (3) listed in the previous section. The only provision in MICRO for storing and printing varying length character fields is type E, external data fields.

4.4.7 TYPE E - EXTERNAL DATA FIELDS

Type E is for format-free character string data associated with a record but on which searches cannot be performed. External data is just that; the character strings are stored in an MTS file external to the MICRO data set. Only a pointer (consisting of a line number for a line file or a notepointer for a sequential file) to each character string is stored as data in the MICRO data set. Each character string stored as external data may be up to 32,767

characters in length (the maximum length or lines in the file in which the data is stored).

The name of the associated MTS file for external data is stored in the dictionary associated with the data set. The file name cannot exceed eleven characters in length. A line file is recommended, as it provides much greater flexibility at a marginal increase in cost over a sequential file. Sequential files are a throwback to the days when a line of a line file could not exceed 255 characters in length.

4.4.8 FIELD SIZE - EXTERNAL DATA FIELDS

External data fields are always four bytes in length. If the external data file is a line file then the value stored is the line number scaled (multiplied) by 1000. Thus, if line 999 contains external data for record then the value of the field associated with that external data is 999000 and is printed as 999.000.

4.4.9 REPRESENTING CHARACTER STRING DATA AS CATEGORIES

If a limited number of unique character string values occur repetitively (such as the names of states) it might be wise to convert the data to numerical values (with the values 1 to 50, continuing the analogy) with the unique character strings as category names (the state names). This will work if the unique category names can be expressed in eleven or fewer characters. If no searching is desired, and the character string can not be shortened into an eleven character category name, the string may be entered as the category description. In the PRINT command, the category description may be printed instead of the category name. Significant storage savings (overhead reduction) can be accomplished. See Section 4.4.10.

4.4.10 CODIFICATION OF DATA VALUES

In Chapter 3, the differences between natural and codified data were discussed. The values of a field in a MICRO data set may be codified. This is accomplished by giving a name to each unique value of a field. In MICRO, this name is called a category name and the act of codification is called categorization. MICRO does not store category names instead of the corresponding data value but maintains a conversion table in the dictionary for converting the value into a category name and vice versa. Therefore, the user may refer to a codified state by its value or its category name (or abbreviation). Thus, a variety of users may use the same data set in the manner which is most comfortable and useful to each of them.

The use of categories is not restricted to MICRO commands. On data entry or update, through the ENTERDATA and UPDATE commands or their stand-alone equivalent programs, category names may be used instead of the corresponding data value. On output, data values are automatically converted to the corresponding category name but output as data values is also available.

All data values of a field do not have to have corresponding category names. If, on output, the data value is not found in the list of category values, the data value will be printed. It is not uncommon for some fields to have only one categorized value. A numerical field may have one value which represents "missing data." A category called MISSINGDATA with that value may be the only category for that field. Printing the values of that field would result in numerical output except when the missing data value is encountered.

4.4.11 REQUIRED CATEGORIES

At the time of data entry or update through the ENTERDATA and UPDATE commands or their stand-alone equivalent programs, it is possible to restrict the acceptable values of a field to those for which there are defined categories. An entered value which is not a defined category name, category abbreviation or category value will be rejected for a field which "requires categories." For example, the field SEX may be defined to require that entered data be either MALE or FEMALE. The value IT would be rejected. Each categorical field (types UC, SC and CC) may be defined during data set definition to require categories as input to that field. The fact that categories are required does not prevent non-categorical values from being introduced through the CHANGE and COMPUTE commands.

4.4.12 CATEGORICAL INFORMATION

There are five pieces of information needed to define a category of a field:

- (1) Category Name;
- (2) Category Abbreviation;
- (3) Category Type;
- (4) Category Value; and
- (5) Category Description.

4.4.13 CATEGORY NAME AND ABBREVIATION

Each category corresponding to a data value must have a user-defined name and may have a user-defined abbreviation. The name must be no more than eleven characters in length. The definer

may also define an abbreviation or short form of the category name up to four characters in length. If the definer does not choose to define an abbreviation then the first four characters of the category name will be used as the default abbreviation. Thus, if the category name is less than five characters then the default abbreviation is the same as the field name. Category names and abbreviations may not duplicate names or abbreviations of other categories of the same field. However, different fields may have the same category names and/or abbreviations. Category names and abbreviations may not be the same as any field name or abbreviation.

4.4.14 CATEGORY TYPE

There are two types of categories and the distinction pertains only to the ENTERDATA or UPDATE commands and facilities and the JOIN command. At the time of data entry or update, it is possible to specify that the category's "default" value for a field be used as the data value instead of explicitly specifying a value. A default category is type 1; all others are type 0. If more than one category is type 1 then the first type 1 category defined in the category list will be used as the default category.

Using the SEX field for an example, it is possible that the SEX of the student is not known thereby creating a need for a third category (in addition to MALE and FEMALE) called UNKNOWN. If UNKNOWN was a type 1 category and MALE and FEMALE were type 0, then SEX would always default to UNKNOWN, if no data were entered for SEX.

4.4.15 CATEGORY VALUE

Each category must have a value so that the association between category name and a data value can take place. The value must be acceptable within the specific field's definition of field type and size.

See Section 4.3.5 for a discussion of acceptable value ranges for integer fields. If the integer field is scaled then the category value must be given in its absolute integer form. (See Section 4.3.7 on scaling.)

For character fields, the category value is a string from one to four characters up to the maximum specified in the field size. If the value has fewer characters than the field size then blanks are padded to fill to the field size. The category value may contain embedded blanks (that is, blanks not at the end of the string) only if enclosed in primes ('). Leading primes which are part of the category value and primes within category values must be

represented as double primes ('); not quotation marks ") and the entire value must be enclosed in primes. For example, 'I'M' would express the four-character value I'M.

4.4.16 CATEGORY DESCRIPTION

Each category must have a description. The description may be up to 256 characters in length. If the definer does not provide a description at the time of data definition (see Chapter 5) then the category name will be used as the description.

4.4.17 SHARED CATEGORIES

Since it is possible for two or more fields to have the same categories, it is possible for two or more fields to share the same category list. This avoids maintenance of redundant category lists, saves space, etc.

4.4.18 DEFAULT VALUES AND REQUIRED DATA

All fields have default data values. For categorical fields, the first type 1 category will be used for the default value. For character strings without categories or with no type 1 category, the default value is a string of blanks equal in length to the field size. For integers without categories or with no type 1 category, the default value is zero. However, the data set definer can make it impossible for the user to default the value of a field during data entry or update by declaring that data is required for a field. Default values will be ignored for such fields.

4.5 ADVANCED DESIGN CONCEPTS

There are several advanced design concepts that are not required for most applications:

- (1) Integration of data sets;
- (2) Repeating groups; and
- (3) Overlapping fields.

Each will be discussed separately.

4.5.1 INTEGRATION OF DATA SETS

It is possible for MICRO to take different collections of data from the same or different sources and integrate them into a consolidated data set. This act of joining requires forethought on

the part of the data set definer because the data sets must have one or more fields in common through which integration will take place. The fields are the "key fields."

Consider the following example. The Internal Revenue Service (IRS) receives a computer tape from employers listing the wages paid, deductions from wages for taxes, etc., for each employee. The employee is identified by his Social Security number. The Welfare Administration receives a computer tape from all state welfare agencies listing the welfare benefits paid to each recipient. The recipient is also identified by his Social Security number. Both data collections have one element in common, the Social Security number. If these two data collections were MICRO data sets then they could be joined into one data set if and only if the key fields in each set, the Social Security numbers, were stored in the same field type and field size. If these two data sets were integrated via the JOIN command, analysis of wages received by welfare recipients could, hypothetically, be undertaken.

The link between data sets need not be just one field but may be many. For example, area code and telephone number can be used to connect long-distance charges with the name of a caller. The link need not be unique. If the area code and telephone number above were called many times, there would be many records in the data set identified by that combination.

There are two procedural possibilities when data in two sets are integrated and there is no matching key between the records of the data sets:

- (1) To drop the record for which there is no matching key; or
- (2) To create the missing data for that part of the integrated data record for which no data existed.

Both are possible in MICRC.

If both data sets contain only one occurrence of the same value combinations of the key fields, the resulting data set will have the same number of records as occurred in the larger of the two parent data sets. However, if multiple occurrences exist in one or both sets, growth (in terms of the records in the resulting data set) will occur. Growth is limited by the GROWTH FACTOR facility; see the SET, RESTRICTANDJOIN and JCIN command descriptions in Chapter 9. The GROWTH FACTOR may be changed from the default value by the SET command or the aGF command modifier.

4.5.2 REPEATING GROUPS

Neither multiple values for a single field or multiple occurrences of values for a group of fields are permitted in one record in MICRO. The effect of both may be emulated by using the integration technique discussed in Section 4.5.1.

Consider the example of a parent and his children. One parent may have many children. It is possible to create a MICRO data set with one record containing the following fields:

```
PARENT-NAME
PARENT-NUMBER
CHILD-NAME-1
CHILD-NAME-2
.
.
.
CHILD-NAME-n
```

That data set would be awkward to search and very wasteful of storage since most parents do not have more than two children. The alternate and recommended solution would be to have two data sets, PARENT and CHILDREN, where PARENT would consist of two fields: PARENT-NAME AND PARENT-NUMBER. CHILDREN would also consist of two fields: CHILD-NAME and PARENT-NUMBER. These two sets could be linked via PARENT-NUMBER resulting in a set consisting of:

```
PARENT-NAME
PARENT-NUMBER
CHILD-NAME
```

There would be one record in the result set for each parent-child relation (connection via PARENT-NUMBER). For those parents with more than one child, the PARENT-NAME and PARENT-NUMBER would redundantly appear in the result set.

Connection need not be done always on numerical fields. PARENT-NUMBER was used in the above example because PARENT-NAME might not have been a unique identifier and the result would have been a proliferation in the relationship between parents and children.

The repeating group of fields is handled in the same way as the single repeating field just described. An example of solving the problem of a repeating group of fields is presented in Section 4.6.

4.5.3 OVERLAPPING FIELDS

An overlapping field applies only to character type fields and is not widely used. It applies when individual characters have meaning in addition to the meaning of the whole string of characters. This is best discussed by an example.

Consider a character field of length 5 named ZIP-CODE in which the Zip Code digits are treated as characters (as opposed to being stored as a number). The five digits represent a specific delivery area within a local post office. The first digit alone indicates an area of the country, the first two digits identify a regional distribution center and the first three digits identify a local distribution center. Thus there are four possible meanings associated with the five characters of data.

The use of overlapping fields allows the user to query all meanings but with only the most complete actually being stored.

Caution should be taken at the time of data entry and modification because there really is only one field stored. Selection of overlapped fields via the SELECT command (as described in Chapter 9) can cause partially redundant fields to be created where before there was only one field. Use of the SORT command always destroys the overlapping relationship by creating redundant fields.

4.6 PHYSICAL DESIGN FOR THE EXAMPLE DATA SETS

The physical design of the example MICRO data sets must be determined using the information in Section 3.6 describing the logical definition and using the constraints discussed in this chapter.

The financial aid information is a repeating group of fields, i.e., each student may have many groups of financial aid information (one for each term and source from which aid was received) associated with his once-occurring demographic information. Because MICRO does not directly support repeating groups, it is necessary to divide Intercollegiate University's (ICU) information into two data sets which can be joined via the student identification number. All of the singly occurring information about the students will be put into a data set which will be called "STUDENTS". All of the financial aid information which may occur many times for one student will be put into a data set which will be called "FIN-AID".

Both data sets will be scrambled and not "destroyable" but "replaceable." Because ICU has some data on existing students

already keypunched, they will try the custom programming technique of data entry. For the data that is not already in machine readable form, ICU will try both of the ENTERDATA techniques of data entry.

The determination of field type is straightforward for all fields except for the variable lengthed financial aid remark. It must be handled in MICRO as an external data field (type E) because its length may exceed 256 characters. ICU is willing to live with the constraints imposed on external data fields (no searching, etc.).

Many definers find it useful to lay out all of the information that the MICRO.DEF program will require on a convenient worksheet. Sections 4.6.1 and 4.6.2 display worksheets for the example data sets that contain all category and field information except for the descriptions. LENGTH refers to the length in bytes but the maximum value may be used instead. SFUN and SFAC refer to the scaling function and scaling factor respectively while CREQ and DREQ refer to "categories required" and "data required" respectively.

4.6.1 WORKSHEET FOR THE FIN-AID DATA SET

NAME	ABBR	TYPE	LENGTH		SFUN	SPAC	CREQ	DREQ
			VALUE					
STUDENT-ID-NO	SIN	U	B4					YES
YEAR-AWARDED	YEAR	U	E1					YES
TERM-AWARDED	TERM	UC	B1				YES	YES
FALL	F	0		1				
WINTER	W	C		2				
SUMMER	S	0		3				
CLASS-LEVEL	CL	UC	B1				YES	YES
FRESHMAN	FR	0		1				
SOPHOMORE	SC	C		2				
JUNIOR	JR	0		3				
SENIOR	SR	C		4				
TYPE-OF-AID	TYPE	UC	B1				YES	YES
TUITION	TU	C		1				
ROOM-BOARD	RE	0		2				
CASH	CA	C		3				
LOAN	IC	0		4				
WORK-STUDY	WS	C		5				
OTHER	OTH	0		6				
SOURCE-OF-FUNDS	SCF	UC	B1				YES	YES
UNIVERSITY	UNIV	0		1				
STATE-GOV	SGOV	C		2				
FEDERAL-GOV	FGCV	0		3				
INDUSTRY	IND	C		4				
FOUNDATION	FCUN	C		5				
OTHER	OTH	C		6				
AMOUNT	AMT	S	E4		/	100		YES
REMARK*	REM*	E	B4					NO

SFUN = Scaling Function
 SFAC = Scaling Factor
 CREQ = Categories Required
 DREQ = Data Required

4.6.2 WORKSHEET FOR THE STUDENTS DATA SET

NAME	ABER	TYPE	<u>LENGTH</u> VALUE	SFUN	SPAC	CREQ	DREQ
STUDENT-ID-NO	SIN	U	B4				YES
NAME	NA	C	24				YES
ETHNICITY	RACE	UC	B1			YES	YES
UNKNCWN	UNK	1	0				
WHITE	WH	0	1				
BLACK	EL	0	2				
AMER-INDIAN	AI	C	3				
SPANISH	SP	0	4				
ORIENTAL	CR	C	5				
OTHER	CTH	0	6				
SEX	SEX	CC	B1			YES	YES
MALE	MA	C	M				
FEMALE	FE	0	F				
UNKNCWN	UNK	1	U				
BIRTHDATE	BDAY	UC	B1			NO	YES
MISSING	MISS	1	0				
CLASS-YEAR	CLYR	SC	B1			NO	NO
UNKNCWN	UNK	1	-1				

SFUN = Scaling Function SFAC = Scaling Factor CREQ = Categories Required DREQ = Data Required

CHAPTER 5

DATA SET DEFINITION

With the physical design completed according to the description in Chapter 4, the definer is ready to incorporate the design into a MICRO data set definition. Definitional information is gathered about the data set, its MTS files, its fields and their categories by a program. This program prompts the definer for all information and then stores it in the dictionary file associated with the data set.

5.1 HOW TO RUN THE DATA SET DEFINITION PROGRAM

The Data Set Definition Program is self-contained and not executable from the MICROC command language. It is executed by issuing the following MTS command:

```
$RUN ILIR:MICRO.DEF
```

The program will prompt the definer for all information required.

5.2 GENERAL RULES

- (1) The definer may return to MTS at any point in the definition process by typing "MTS" in column 1 in response to any request.
- (2) The definer may give an MTS command at any time. A reply is considered to be an MTS command whenever a "\$" is typed in Column 1. There is one exception. \$ENDFILE is treated as an end-of-file indicator and not an MTS command.
- (3) If the definer wants to send either "MTS" or any reply beginning with a dollar sign to the definition program, he should begin in column 2 or later.
- (4) With the exception of (1) through (3) above, all other replies may be made free of specific columnar format requirements.
- (5) The definer may anticipate the prompting for field and category information by typing more than one answer on the same line separated by blanks. Prompting will be suppressed for each item until the program has used all of the information provided or until an error is found.

- (6) The definer may type long replies (such as descriptions) on more than one input line by typing a minus sign (-) as the last character transmitted in the line to be extended. The minus sign will not appear in the dictionary.
- (7) A timer interrupt will occur after 5 CPU seconds have been used. When this happens the program will print

<TIME OUT.....

<USE "RESTART" TO CONTINUE IF YOU ARE MAKING PROGRESS

Then the amount of money spent since signing on will be printed and the definer will be returned to MTS command mode. If the definer restarts the definition program, the timer will be reset to stop again after five more seconds. Experience indicates that a great deal of definitional work can be done in 5 CPU seconds.

- (8) In the following discussion of the definition program, the term "no reply" means sending end-of-file (Control C or "\$ENDFILE") or pushing return without typing any characters. Default values are obtained by giving no reply. When a reply is required the program will try up to five times to get a reply and will then type "Failure to reply" after which the definer will be returned to MTS.
- (9) The definer may issue attention interrupts at any time. The behavior of the program will vary depending upon what the program was doing at the time of the interrupt:
 - (a) If interrupted before the program first says "field name?", the definer will be placed in MTS command mode. The program may be continued by issuing the \$RESTART command.
 - (b) If interrupted in the middle of a field definition but before adding any categories, the definer will be asked "Do you want to add more data?" YES will start the field definition again. NO will put the definer in CORRECTION mode. Either reply will result in the incomplete field information being deleted.
 - (c) If interrupted in the middle of a category then the definer will be asked "Do you want to add more data?" YES will start the category definition again. NO will place the definer in CORRECTION mode but all of the current field definition including categories will be lost.

- (d) To insure that there is a valid dictionary at all times attention interrupts may be ignored for a short period of time while the dictionary is being modified.

5.3 PROMPTING FOR DEFINITIONAL INFORMATION

Information required for the data set definition program is divided into three areas paralleling the discussion in Chapter 4:

- (1) Information pertaining to the data set as a whole;
- (2) Information pertaining to fields; and
- (3) Information pertaining to categories.

Each will be discussed separately. The discussion of the latter two applies to each field and each category of the data set.

5.4 GENERAL DATA SET INFORMATION

General data set information includes the following, presented in the order in which the definer will be prompted (if all options are used):

- (1) Dictionary file name;
- (2) Master signon-ID;
- (3) MICRC runfile name;
- (4) Local command trace file name;
- (5) Master command trace file name;
- (6) MICRO data set name;
- (7) Data set description;
- (8) Data set destruction status;
- (9) Data set replacement status;
- (10) Data file name;
- (11) Conversion file name;
- (12) Usecount;

(13) Crosstabulated data set status; and

(14) Data set scramble status.

Each is discussed separately.

5.4.1 DICTIONARY FILE NAME

The definer must provide the name of the dictionary file where the data set definition will be stored. If the file already contains a valid dictionary it is assumed the dictionary is being modified. The definer will be informed of the number of existing fields and will then be asked for the name of the next field. (See Section 5.5.)

If the dictionary file does not exist it will be assumed that a new dictionary is being defined. The dictionary file will be created automatically (unless the file is on another CCID).

The dictionary file name also follows the MTS limitations for length and acceptable characters. It also must be a sequential file, and need not be created prior to definition. Time-honored practice dictates that the dictionary file name be the same as the data set name with a pound or sharp sign (#) appended. Therefore, the file name may be up to 12 characters, including the #. This practice is followed by any data set created through the MICRO SAVE command. If the definer wishes to specify a file on another CCID then the maximum length is 17, including the signon-ID and a colon.

If the definer's signon-ID does not have a MICRO directory file, then a file called DIRECTORY will be created and initialized automatically and the definer will be prompted for a master signon-ID, runfile name, and local and master command trace file names. (See Sections 5.4.2 through 5.4.5.) If there is a directory, the user will then be prompted for the data set name. (See Section 5.4.6.)

5.4.2 MASTER SIGNON-ID

The master signon-ID is the CCID on which the macro library, the default master command trace file and the default subroutine library are located. Reply is optional. If no reply is given then MICRC will use ILIR for the CCID. This denotes that the Institute of Labor and Industrial Relations files are to be used. This question is asked only if the directory file is new. The master signon-ID can be changed via the MICRC.DIR program. (See Section 5.9.)

- (d) To insure that there is a valid dictionary at all times attention interrupts may be ignored for a short period of time while the dictionary is being modified.

5.3 PROMPTING FOR DEFINITIONAL INFORMATION

Information required for the data set definition program is divided into three areas paralleling the discussion in Chapter 4:

- (1) Information pertaining to the data set as a whole;
- (2) Information pertaining to fields; and
- (3) Information pertaining to categories.

Each will be discussed separately. The discussion of the latter two applies to each field and each category of the data set.

5.4 GENERAL DATA SET INFORMATION

General data set information includes the following, presented in the order in which the definer will be prompted (if all options are used):

- (1) Dictionary file name;
- (2) Master signon-ID;
- (3) MICRC runfile name;
- (4) Local command trace file name;
- (5) Master command trace file name;
- (6) MICRO data set name;
- (7) Data set description;
- (8) Data set destruction status;
- (9) Data set replacement status;
- (10) Data file name;
- (11) Conversion file name;
- (12) Usecount;

(13) Crosstabulated data set status; and

(14) Data set scramble status.

Each is discussed separately.

5.4.1 DICTIONARY FILE NAME

The definer must provide the name of the dictionary file where the data set definition will be stored. If the file already contains a valid dictionary it is assumed the dictionary is being modified. The definer will be informed of the number of existing fields and will then be asked for the name of the next field. (See Section 5.5.)

If the dictionary file does not exist it will be assumed that a new dictionary is being defined. The dictionary file will be created automatically (unless the file is on another CCID).

The dictionary file name also follows the MTS limitations for length and acceptable characters. It also must be a sequential file, and need not be created prior to definition. Time-honored practice dictates that the dictionary file name be the same as the data set name with a pound or sharp sign (#) appended. Therefore, the file name may be up to 12 characters, including the #. This practice is followed by any data set created through the MICRO SAVE command. If the definer wishes to specify a file on another CCID then the maximum length is 17, including the signon-ID and a colon.

If the definer's signon-ID does not have a MICRO directory file, then a file called DIRECTORY will be created and initialized automatically and the definer will be prompted for a master signon-ID, runfile name, and local and master command trace file names. (See Sections 5.4.2 through 5.4.5.) If there is a directory, the user will then be prompted for the data set name. (See Section 5.4.6.)

5.4.2 MASTER SIGNON-ID

The master signon-ID is the CCID on which the macro library, the default master command trace file and the default subroutine library are located. Reply is optional. If no reply is given then MICRC will use ILIR for the CCID. This denotes that the Institute of Labor and Industrial Relations files are to be used. This question is asked only if the directory file is new. The master signon-ID can be changed via the MICRC.DIR program. (See Section 5.9.)

5.4.3 MICRO RUNFILE NAME

The MICRO runfile is an MTS file containing MICRO commands and MTS commands which will be read and executed by MICRO after execution begins. The runfile is optional and there is no default. A reply is optional. This question is asked only if the directory file is new. The runfile name can be changed via the MICRO.DIR program and the MICRC SET command. (See Section 5.9.)

5.4.4 LOCAL COMMAND TRACE FILE NAME

The local command trace file is the file to which all MICRO commands are copied after they have been read by MICRO prior to their execution. This happens only when the command trace facility is enabled; see Section 2.8.1 and the SET command description in Chapter 9. No reply is required. If no reply is given, MICRO will copy commands into the file ICCAICMT on the current signon-ID. This question is asked only if the directory is new. The local command trace file name can be changed via the MICRO.DIR program.

5.4.5 MASTER COMMAND TRACE FILE NAME

The master command trace file is where the contents of the local command trace file are copied; that is, it can be a central repository for the local command trace files on numerous signon-IDs. This happens only when the command trace facility is enabled; see Section 2.8.1 and the SET command description in Chapter 9. If no reply is given then MICRO will copy the local command trace file into a file called MASTERCMT located on the master signon-ID. This question is asked only if the directory is new. The master command trace file name can be changed via the MICRO.DIR program.

5.4.6 MICRO DATA SET NAME

The special characters not permitted in any MICRO name are:

- blank ()
- comma (,)
- vertical bar (|)
- less than (<)
- greater than (>)
- ampersand (&)
- exclamation point (!)
- question mark (?)
- semi-colon (;)
- equal sign (=)
- prime (')
- at sign (@)

All other characters may be used. It does not contain a signon-ID since it is not an MTS file. It may not be the same as any other data set defined in the definer's directory file. A reply is required.

5.4.7 DATA SET DESCRIPTION

The data set description is optional and may be up to 242 characters in length. The current date and signon-ID are appended to the reply.

5.4.8 DATA SET DESTRUCTION STATUS

YES indicates that the MICRO DESTROY command can be used to destroy all parts of this data set as described in Section 4.1.3. NO indicates that destruction by this means cannot take place. A reply is required.

5.4.9 DATA SET REPLACEMENT STATUS

YES indicates that the MICRO REPLACE command can be used to replace the permanent data file associated with the data set as described in Section 4.1.4. NO indicates that replacement will not be allowed. A reply is required.

5.4.10 DATA FILE NAME

The data file is an MTS sequential file which contains the data associated with the MICRO data set. It need not exist at the time of definition. The name may be up to twelve characters (an MTS limitation) and any characters acceptable to MTS for a file name may be used. If the definer wishes to specify a file on another CCID then the maximum length is 17, including the signon-ID and a colon. A magnetic tape can be used instead of an MTS disk file. Four asterisks (****) should be used for the data file name which indicates that the data set resides on tape. (See Section 5.11.)

5.4.11 CONVERSION FILE NAME

The conversion file contains the conversion specifications for the user-specified batch fixed-format method of data entry. (See Section 6.3.) The file need not exist at the time of definition. The conversion file name may be up to 12 characters in length if the optional signon-ID is not specified. If the signon-ID is specified, the maximum length is 17 characters. A reply is optional.

5.4.12 USECOUNT

The usecount is the number of data sets which are sharing a dictionary file. The number must be less than 255 and the default is 1. A reply is optional.

5.4.13 CROSSTABULATED DATA SET STATUS

YES indicates that the data set is crosstabulated and causes the CCOUNT field to be automatically created as the last field. The default is NO, which indicates that the data set is not crosstabulated, and a reply is optional.

5.4.14 DATA SET SCRAMBLE STATUS

YES indicates that the data set is scrambled as indicated in Section 4.1.9. NO indicates that the data set is not scrambled. The default is NC and a reply is optional.

The scramble key or password used to scramble or unscramble a data set is not part of the data set definition, i.e., it is not permanently stored. When a user wants to access a data set which has been scrambled, he will then be prompted for the scramble key.

5.5 INFORMATION PERTAINING TO FIELDS

For each field of a data set, the following information, presented in the order in which it is prompted, must be defined:

- (1) Field name;
- (2) Field abbreviation;
- (3) Field type;
- (4) Field length;
- (5) Overlap column;
- (6) Overlap length;
- (7) Maximum value;
- (8) External data file name;
- (9) Scaling function;
- (10) Scaling factor;

- (11) Category required status;
- (12) Data required status; and
- (13) Field description.

Each is discussed separately.

5.5.1 FIELD NAME

The field name can be up to sixteen characters in length and may not contain blanks or any of the characters:

, | < > & ! ? ; = ' @

Within the data set, the name must be unique; that is, it cannot be the same as any existing field or category name or abbreviation.

There is no default value. If the definer gives no reply then MICRO.DEF will proceed to CORRECTION (editing) mode.

5.5.2 FIELD ABBREVIATION

The field abbreviation may be up to four characters in length and, like the field name, must also be unique. A reply is optional and the default value is the first four characters of the field name.

The abbreviation may not contain any of the characters described in Section 5.5.1. MIDAS users should review the warning in Section 4.2.1.

5.5.3 FIELD TYPE

The field type was discussed extensively in Section 4.2.3. The definer must specify one of the following field types or the associated integers:

U = 1	[unsigned integer field without categories]
C = 2	[character string field without categories]
UC = 3	[unsigned integer field with categories]
CC = 4	[character string field with categories]
S = 5	[signed integer field without categories]
E = 6	[external data field]
SC = 7	[signed integer field with categories]

There is no default and a reply is required.

5.5.4 CHARACTER FIELD LENGTH

Field length is requested only for field types C and CC. The length of the string or the name of the field to be overlapped (See Section 4.5.3) must be given. The length must be from 1 to 255 for type C and from 1 to 4 for type CC. If a field name is given (to be overlapped) then that field must be type C or CC and must already be defined. A reply is necessary; there is no default.

5.5.5 OVERLAP COLUMN

The overlap column pertains only to those character fields which overlap another character field; i.e., where a field name was given for the field length. (See Section 5.5.4) It indicates which column of the previously defined field will be used as the first column of the field being defined. A reply is necessary and must be an integer from 1 to the length of the existing field. (See Section 4.5.3.)

5.5.6 OVERLAP LENGTH

The overlap length pertains only to those character fields which overlap another character field. It indicates the length of the second field (starting with the overlap column). A reply is required. (See Section 4.5.3.)

5.5.7 MAXIMUM VALUE

The maximum absolute value to be stored in a U, UC, S, or SC field is used to calculate the field size as described in Section 4.3.5. A reply is necessary. Experienced users can express the value in the actual number of bytes required for the field by prefixing the number of bytes with a "B" (for bytes) or a "W" (for width) e.g., B3 or W3 would define a 3-byte field.

5.5.8 EXTERNAL DATA FILE NAME

The external data file name is requested for field type E. It is the name of the line or sequential MTS file which will contain the data for this field. The file need not exist at the time of definition. A reply is required. If no CCID is provided, the current signon-ID will prefix the name specified.

5.5.9 SCALING FUNCTION

The scaling function was discussed extensively in Section 4.3.7. It is requested for field types U, UC, S and SC. The following valid functions or their numerical equivalents can be

specified:

blank	= (0)	[no scaling]
+	= (1)	[addition]
-	= (2)	[subtraction]
*	= (3)	[multiplication]
/	= (4)	[division]

A reply is optional and the default is 0.

5.5.10 SCALING CONSTANT

The scaling constant is requested only if the field is scaled and then a reply is required. If it is scaled by addition or subtraction the constant can be any non-zero integer. If the scaling function is division or multiplication then the number must be a power of ten.

5.5.11 CATEGORY REQUIRED STATUS

The category required status is requested only for field types UC, CC and SC. It indicates that the ENTERDATA command (and facility) will restrict data for the field to valid categorical values. A reply is optional and the default value is NO.

5.5.12 DATA REQUIRED STATUS

The data required status is requested for all fields. YES indicates that a value must be explicitly provided for this field when entered via the ENTERDATA command or facility. A reply is optional and the default is NO.

5.5.13 FIELD DESCRIPTION

The field description is requested for all fields and the optional reply may be up to 256 characters in length. The default value is the full field name.

5.6 INFORMATION PERTAINING TO CATEGORIES

There are five pieces of information that are used to describe each category of a type UC, SC or CC field:

- (1) Category name;
- (2) Category abbreviation;
- (3) Category type;
- (4) Category value; and
- (5) Category description.

Each will be discussed separately.

5.6.1 CATEGORY NAME

A category name can be up to eleven characters in length and cannot be the same as any field name or abbreviation or other category name or abbreviation within the same field. It can be the same as a category name or abbreviation in a different field. If no categories have been defined for a field then the definer can use the categories of another field of the same type and length. This is called "sharing categories" and is accomplished by entering the name or abbreviation of the field whose categories are to be shared with the field whose categories are being defined.

If no reply is given for a category name, prompting for categories will cease.

5.6.2 CATEGORY ABBREVIATION

The category abbreviation can be up to four characters in length and cannot be the same as any field name or other category name or abbreviation within the same field. It can, however, be the same as the category name or the same as a category name or abbreviation in a different field. A reply is optional and the default is the first four characters of the category name or the category name itself, if it is less than five characters in length.

5.6.3 CATEGORY TYPE

Category type indicates the default category when using the ENTERDATA and UPDATE commands or facilities or the JOIN command. A reply of 1 indicates that this is a default category for the field. A reply of 0 indicates that it is not a default category. A reply is optional and the default value for this request is 0.

No check is made to see if there is more than one type 1 category per field. The first type 1 category will be used by ENTERDATA.

5.6.4 CATEGORY VALUE

The category value is used to codify and decodify the category name and abbreviation. It must fit within the size allotted to the field. For numerical fields, a category value must fit within the storage allotted to the field, i.e., the number of bytes needed to represent the maximum value.

For character fields, the category value is a string of the defined length. The value can contain blanks only if enclosed in primes (''). Leading primes which are part of the value and primes within values already enclosed in primes must be represented by

typing double primes ('', not ") The string must be enclosed within primes if it is not already enclosed.

5.6.5 CATEGORY DESCRIPTION

The category description is optional and can be up to 256 characters in length. The default value is the full category name.

5.6.6 AN ALTERNATIVE TECHNIQUE FOR ENTERING CATEGORY INFORMATION

If the definer has a field with many categories or fields appearing in different data sets with the same categories, it may be desirable to put the category definitions into an MTS file for later processing by the MICRO.DEF program. Thus the definer is using MTS to cause MICRO.DEF to read this information from a file by using the "\$MTS CONTINUE WITH" facility. Each of the five pieces of category definition must be included in the MTS file for each category (i.e., no defaults are allowed) and preferably on the same line (for ease of reading). These pieces must be in the order expected by MICRO.DEF:

- (1) Category name;
- (2) Category abbreviation;
- (3) Category type;
- (4) Category value; and
- (5) Category description

and each piece must be separated by one or more blanks. The description should not be put within primes as the fifth piece is expected to be the description by MICRO.DEF and thus can contain blanks.

If this technique is followed then the MTS file can be listed and corrections made prior to running MICRO.DEF. Because a new category cannot be inserted between existing categories, the order of categories can be set in the MTS file. When an insertion to the category list is required, the MTS file can be updated (edited) then MICRO.DEF can be run to delete all of the categories for that field after which all of the category information in the MTS file can be appended (via the APPEND command).

Consider the following example used in defining the categories for the SOURCE-CF-FUNDS field of the FIN-AID data set. All of the category information is placed into an MTS file named ILIR:CAT.SOF which is listed (via \$LIST) here:

```
1      UNIVERSITY UNIV 0 1 UNIVERSITY
2      STATE-GOV SGOV 0 2 STATE GOVERNMENT
```

```
3      FEDERAL-GOV FGCV 0 3 FEDERAL GOVERNMENT
4      INDUSTRY      IND 0 4 INDUSTRY
5      FOUNDATION FCU 0 5 FOUNDATION
6      OTHER OTH 0 6 OTHER SOURCE
END OF FILE
```

When the first category name for the field SOURCE-OF-FUNDS is needed in the definition process, the definer would reply with

\$CONTINUE WITH ILIR:CAT.SOF RETURN

which causes the file to be read (instead of information being entered by the definer at the terminal). When the information in the file is exhausted, the responsibility for further input will be returned to the definer at the terminal because RETURN was specified. If RETURN is accidentally omitted, the definer will find himself back in MTS command mode after being prompted five times for a category name. To recover from this phenomenon, the definer should enter:

\$RESTART SCARDS=*MSOURCE*

5.7 CHANGING THE DATA SET DEFINITION - CORRECTION MODE

In order to edit the existing definition, the definer should \$RUN ILIR:MICRO.DEF and should give a null reply (carriage return) when asked for a new field name. This will put the definer in CORRECTION mode. The following commands are available for modifying the data set definition:

```
APPEND
CHANGE
CLCSE
DELETE
DOCUMENT
INSERT
MODIFY
REDEFINE
STOP
```

Each command is described in detail on the following pages. The notation for the data set definition command description is the same as that used for the MICRO command descriptions and macros and is discussed in Section 9.2. It must be emphasized that MICRO.DEF does not ever alter the data file associated with a data set if one exists. For instance, adding a field to the definition of an existing (initialized) data set does not cause the creation of a new

field in the data file. Therefore, it is the responsibility of the MICRO.DEF user to insure that the dictionary of a data set is not made incompatible with an existing data file.

The information gathered by MICRO.DEF is not stored in a permanent disk file until a CLOSE, MODIFY, or STOP command is received.

Category sharing may be preserved or broken up as a result of corrections or changes in the data set definition. The definer will be prompted for his choice when a decision is needed.

APPEND

DATA SET DEFINITION COMMAND DESCRIPTION

- PURPOSES:
- (1) To define additional fields; and
 - (2) To define additional categories for a field.

PROTOTYPE: APPEND [<field name>]

Where the absence of <field name> indicates that additional fields are to be defined after the last field definition.

Where the use of <field name> indicates that additional categories are to be added to the specified field.

COMMENT: If no <field name> was given then the command does not return the definer to CORRECTION mode after defining one field but continues prompting the definer for additional field definitions until the user responds with a null reply.

CHANGE

DATA SET DEFINITION COMMAND DESCRIPTION

PURPOSE: To permanently redefine the specified part of the dictionary.

PROTOTYPE: CHANGE [<field name>[<category name>] <part name> [[TO]<new value>]

Where the absence of <field name> and <category name> indicate that one of the general data set parts is to be changed.

Where <field name> identifies the field in which the change is to be made.

Where <category name> identifies the category of <field name> in which the change is to be made.

Where <part name> identifies the part of the dictionary (within the field and category, if specified) which is to be changed. The following part names may be specified to modify a field's definition:

- NAME (field name)
- ABBREVIATION (field abbreviation)
- DESCRIPTION (field description)
- FUNCTION (scaling function)
- FACTOR (scaling factor)
- CATEGORYREQUIREDSWITCH (category required)
- CR SWITCH (same as previous)
- DATAREQUIREDSWITCH (data required)
- DR SWITCH (same as previous)
- EXTERNALDATAFILE (external data file)

The following part names may be specified to modify a category's definition:

- NAME (category name)
- ABBREVIATION (category abbreviation)
- TYPE (category type)
- VALUE (category value)
- DESCRIPTION (category description)

The following part names may be specified to

modify general data set parameters without specifying particular field or category names:

USECCUNT (integer)
SCRAMELESWITCH (YES|NO)
XTABSWITCH (YES|NO)

Where the absence of <new value> will cause the definer to be prompted for that new value.

EXAMPLES:

CHANGE XTABSWITCH TO NO

CHANGE AMCUNT FACTOR TO 100

CHANGE SOURCE-CF-FUNDS UNIVERSITY ABBREVIATION TO
UNIV

COMMENTS:

- (1) As in command names, the underscored letters of the <part name> indicate the minimum characters needed to specify a part. Any additional consecutive letters may be included in a part name up to the entire name.
- (2) Upon completion of the change, this command will return the definer to CORRECTION mode.
- (3) XTABSWITCH determines whether the data set is crosstabulated.

CICSE

DATA SET DEFINITION COMMAND DESCRIPTION

PURPOSE: To store an up-to-date permanent copy of the dictionary in the dictionary file, thus saving the dictionary definition work completed during the current session.

PROTOTYPE: CICSE

COMMENT: After closing the dictionary, this command will return the definer to CORRECTION mode.

DELETE

DATA SET DEFINITION COMMAND DESCRIPTION

- PURPOSES:
- (1) To delete a field from the data set definition;
 - (2) To delete a category from a field's definition;
and
 - (3) To delete all categories of a field from that field's definition.

PROTOTYPE: DELETE <field name>[<category name>|ALL]

Where DELETE <field name> will cause that field to be dropped from the data set definition.

Where DELETE <field name><category name> will cause that category to be dropped from the specified field's definition.

Where DELETE <field name> ALL will cause all categories to be dropped from the specified field's definition.

- COMMENTS:
- (1) After the deletion, this command will return the definer to CCRRECTION mode.
 - (2) Deletion of the COUNT field from an XTAB data set will cause the XTABSWITCH to be turned off; i.e., the data set will become a non-XTAB data set.

DOCUMENT

DATA SET DEFINITION COMMAND DESCRIPTION

- PURPOSES:
- (1) To print the entire data set definition;
 - (2) To print the definition of all fields excluding categories;
 - (3) To print the definition of a specified field; and
 - (4) To print the definition of a specified category of a specified field.

COMMAND SYNONYM: SHOW

PROTOTYPE: DOCUMENT [*]<field name> [<category name>]]

Where the absence of <field name> and <category name> will cause the entire data set to be documented.

Where the specification of * will cause all fields to be documented without category information.

Where the specification of <field name> without <category name> will cause the specified field to be documented.

Where the specification of <field name> and <category name> will cause the specified category of the specified field to be documented.

COMMENT: After documenting, this command will return the definer to CORRECTION mode.

INSERT

DATA SET DEFINITION COMMAND DESCRIPTION

PURPOSE: To define a new field, either as the first field of the data set or as a field positioned logically after a specified field in the data set.

PROTOTYPE: INSERT [<field name>]

Where the absence of <field name> will cause the new field to become the first field of the data set.

Where the presence of <field name> will cause the new field to be positioned logically after the specified field.

COMMENTS:

- (1) After the insertion, this command will return the definer to CORRECTION mode.
- (2) Category names cannot be inserted but must be appended to the end of the category list.

MODIFY

DATA SET DEFINITION COMMAND DESCRIPTION

PURPOSE: To CLOSE the current dictionary in order to modify another dictionary.

PROTOTYPE: MODIFY [<dictionary file name>]

Where the absence of <dictionary file name> will cause the definer to be prompted for the new dictionary file name.

Where the presence of <dictionary file name> will cause that dictionary to be read in for further definition.

COMMENT: This command is identical to STOPping and then rerunning ILIF:MICRO.DFF.

REDEFINE

DATA SET DEFINITION COMMAND DESCRIPTION

- PURPOSES:
- (1) To redefine the usecount, scramble status and XTAB status of the data set;
 - (2) To redefine for the specified field, all of the following: field name, field abbreviation, scaling function, scaling factor, field description, data required status and, if applicable, category required status and external data file name.
 - (3) To redefine all parts of the specified category of the specified field.

COMMAND SYNONYM: REDO

PROTOTYPE: REDEFINE [<field name> [<category name>]]

Where the absence of <field name> and <category name> will yield purpose (1) above.

Where the presence of <field name> and the absence of <category name> will yield purpose (2) above.

Where the specification of <field name> and <category name> will yield purpose (3) above.

COMMENT: After redefinition, this command will return the definer to CORRECTION mode.

STCP

DATA SET DEFINITION COMMAND DESCRIPTION

PURPOSE: To store in the dictionary file the data set definition as defined during the current session and to terminate the program.

PROTOTYPE: STOP

COMMENT: After closing the dictionary the program is terminated.

5.8 CREATING DICTIONARIES FOR THE EXAMPLE SETS

Based on the physical designs for the example data sets, FIN-AID and STUDENTS, outlined in Sections 4.6.1 and 4.6.2, these data sets were created. The actual definer-computer interaction appears in Sections 5.8.1 and 5.8.2. Those lines that are underscored were generated by the MICRO.DEF program. All lines that are not underscored (with the exception of annotation) were entered by the definer. "(carriage return)" indicates that the definer entered a null reply by typing only a carriage return. Annotated remarks inserted for clarity after the session appear within square brackets ([]).

5.8.1 DEFINITION OF THE FIN-AID DATA SET

<<< MICRO DATA DEFINITION SYSTEM <<<
ENTER NAME OF DICTIONARY FILE.
FIN-AID#

CONCERNING: FIN-AID#
DICTIONARY FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.
Y

ENTER MICRO DATA SET NAME.
FIN-AID

DESCRIPTION ?
DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL: DEALS WITH -
FINANCIAL AID INFORMATION. MIKE KAHN

CAN THIS DATA SET BE DESTROYED ?
NO

CAN THIS DATA SET BE REPLACED ?
Y

ENTER NAME OF DATA FILE.
FIN-AID

ENTER NAME OF CONVERSION FORMAT FILE.
(carriage return)

HOW MANY DATA SETS WILL BE USING THIS DICTIONARY ?
1

IS THE DATA SET AN "XTAB SET" ?
NO

SHOULD THE DATA FILE BE SCRAMBLED ?
YES

FIELD NAME ?
STUDENT-ID-NO

ABBREVIATION ?
SIN

FIELD TYPE ?
U

5.8 CREATING DICTIONARIES FOR THE EXAMPLE SETS

Based on the physical designs for the example data sets, FIN-AID and STUDENTS, outlined in Sections 4.6.1 and 4.6.2, these data sets were created. The actual definer-computer interaction appears in Sections 5.8.1 and 5.8.2. Those lines that are underscored were generated by the MICRO.DEF program. All lines that are not underscored (with the exception of annotation) were entered by the definer. "(carriage return)" indicates that the definer entered a null reply by typing only a carriage return. Annotated remarks inserted for clarity after the session appear within square brackets ([]).

5.8.1 DEFINITION OF THE FIN-AID DATA SET

<<< MICRO DATA DEFINITION SYSTEM <<<

ENTER NAME OF DICTIONARY FILE.

FIN-AID#

CONCERNING: FIN-AID#

DICTIONARY FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.

Y

ENTER MICRO DATA SET NAME.

FIN-AID

DESCRIPTION ?

DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL: DEALS WITH -
FINANCIAL AID INFORMATION. MIKE KAHN

CAN THIS DATA SET BE DESTROYED ?

NO

CAN THIS DATA SET BE REPLACED ?

Y

ENTER NAME OF DATA FILE.

FIN-AID

ENTER NAME OF CONVERSION FORMAT FILE.

(carriage return)

HOW MANY DATA SETS WILL BE USING THIS DICTIONARY ?

1

IS THE DATA SET AN "XIAB SET" ?

NO

SHOULD THE DATA FILE BE SCRAMBLED ?

YES

FIELD NAME ?

STUDENT-ID-NO

ABBREVIATION ?

SIN

FIELD TYPE ?

U

5.8 CREATING DICTIONARIES FOR THE EXAMPLE SETS

Based on the physical designs for the example data sets, FIN-AID and STUDENTS, outlined in Sections 4.6.1 and 4.6.2, these data sets were created. The actual definer-computer interaction appears in Sections 5.8.1 and 5.8.2. Those lines that are underscored were generated by the MICRO.DEF program. All lines that are not underscored (with the exception of annotation) were entered by the definer. "(carriage return)" indicates that the definer entered a null reply by typing only a carriage return. Annotated remarks inserted for clarity after the session appear within square brackets ([]).

5.8.1 DEFINITION OF THE FIN-AID DATA SET

<<< MICRO DATA DEFINITION SYSTEM <<<
ENTER NAME OF DICTIONARY FILE.
FIN-AID#

CONCERNING: FIN-AID#
DICTIONARY FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.
Y

ENTER MICRO DATA SET NAME.
FIN-AID

DESCRIPTION ?
DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL: DEALS WITH -
FINANCIAL AID INFORMATION. MIKE KAHN

CAN THIS DATA SET BE DESTROYED ?
NO

CAN THIS DATA SET BE REPLACED ?
Y

ENTER NAME OF DATA FILE.
FIN-AID

ENTER NAME OF CONVERSION FORMAT FILE.
(carriage return)

HOW MANY DATA SETS WILL BE USING THIS DICTIONARY ?
1

IS THE DATA SET AN "XIAB SET" ?
NO

SHOULD THE DATA FILE BE SCRAMBLED ?
YES

FIELD NAME ?
STUDENT-ID-NO

ABBREVIATION ?
SIN

FIELD TYPE ?
U

MAXIMUM VALUE ?
999999999

SCALING FUNCTION ?
(carriage return)

DATA REQUIRED ?
Y

DESCRIPTION ?
STUDENT IDENTIFICATION NUMBER

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
YEAR-AWARDED YEAR U 99

SCALING FUNCTION ?
(carriage return)

DATA REQUIRED ?
Y

DESCRIPTION ?
YEAR THAT FINANCIAL AID WAS AWARDED (IN YY FORMAT)

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
TERM-AWARDED

ABBREVIATION ?
TERM

FIELD TYPE ?
UC

MAXIMUM VALUE ?
B1

SCALING FUNCTION ?
(carriage return)

CATEGORY REQUIRED ?
Y

DATA REQUIRED ?

Y

DESCRIPTION ?

TERM IN WHICH FINANCIAL AID WAS AWARDED

CATEGORY NAME ?

FALL

ABBREVIATION ?

F

CATEGORY TYPE ?

0

CATEGORY VALUE ?

1

DESCRIPTION ?

FALL TERM

CATEGORY NAME ?

WINTER

ABBREVIATION ?

W

CATEGORY TYPE ?

0

CATEGORY VALUE ?

2

DESCRIPTION ?

WINTER TERM

CATEGORY NAME ?

SUMMER S 0 3

DESCRIPTION ?

SUMMER TERM

CATEGORY NAME ?

(carriage return)

FIELD NAME ?

CLASS-LEVEL CL

FIELD TYPE ?
UC 4

SCALING FUNCTION ?
(carriage return)

CATEGORY REQUIRED ?
Y Y

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
FRESHMAN FR 0 1 FRESHMAN

CATEGORY NAME ?
SOPHOMORE SO 0 2

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
JUNIOR JP 0 3

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
SENIOR SR 0 4

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
TYPE-CF-AID

ABBREVIATION ?
TYPE

FIELD TYPE ?
UC

MAXIMUM VALUE ?
6

SCALING FUNCTION ?
(carriage return)

CATEGORY REQUIRED ?
Y

DATA REQUIRED ?
Y

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
TUITION

ABBREVIATION ?
TU

CATEGORY TYPE ?
0

CATEGORY VALUE ?
1

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
(carriage return)

CORRECTIONS:
STOP

[The data definition session was terminated before all fields were defined. MICRO.DEF was run again to complete the definition.]

<<< MICRO DATA DEFINITION SYSTEM <<<
ENTER NAME OF DICTIONARY FILE.
FIN-AID

CONCERNING: FIN-AID
DICTIONARY FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.
NO

ENTER A NEW DICTIONARY NAME OF "CANCEL".
FIN-AID#

CURRENT NUMBER OF FIELDS = 5
FIELD NAME ?
(carriage return)

CORRECTIONS:
[More categories are to be added to the field TYPE-OF-AID.]
APPEND TYPE

CATEGORY NAME ?
ROOM-BOARD

ABBREVIATION ?
RB

CATEGORY TYPE ?
0

CATEGORY VALUE ?
2

DESCRIPTION ?
ROOM AND BOARD

CATEGORY NAME ?
CASH CA 0 3

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
LOAN LO 0 4 LOAN

CATEGORY NAME ?
WORK-STUDY WS 0 5 WORK STUDY PROGRAM

CATEGORY NAME ?
OTHER OTH 0 6

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
(carriage return)

CORRECTIONS:
SHOW TYPE

P(8)	FIELD NAME	ABBR	VALUE	DESCRIPTION	TYPE	LENGTH	DISP	SCALE	FACTOR
P(1)	TYPE-JP-AID	TYPE	REQUIRED	TYPE-JP-AID	UC	1	7		
	CATEGORIES (ONLY)								
	TUITION	TU	1	TUITION					
	ROOM-BOARD	RB	2	ROOM AND BOARD					
	CASH	CA	3	CASH					
	LOAN	LO	4	LOAN					
	WORK-STUDY	WS	5	WORK STUDY PROGRAM					
	OTHER	OTH	6	OTHER					

[NOTE: Field numbers do not refer to the order in which they appear in the dictionary. They start at P(1) for each listing specified. See Section 5.10.]

CORRECTIONS:

AP

FIELD NAME ?

SOURCE-OF-FUNDS SCF UC B1

SCALING FUNCTION ?

(carriage return)

CATEGORY REQUIRED ?

Y

DATA REQUIRED ?

Y

DESCRIPTION ?

SOURCE OF FUNDS OF AWARDED FINANCIAL AID

CATEGORY NAME ?

\$CCONTINUE WITH ILIR:CAT.SCF RETURN

[The category information entry technique (described in Section 5.6.6) was employed.]

CATEGORY NAME ?

CATEGORY NAME ?

CATEGORY NAME ?

CATEGORY NAME ?

CATEGORY NAME ?

CATEGORY NAME ?

(carriage return)

[Each time MICRO.DEF asks for a new category name, the CAT.SOURCE file provides it with all of the categorical information. When this file is exhausted the definer will be prompted for more information. Additional categories may be entered by hand or a null reply may be entered to eliminate the prompting for category information.]

FIELD NAME ?
REMARK* REM*

FIELD TYPE ?
E

EXTERNAL TEXT FILE NAME ?
REMARKS*

DATA REQUIRED ?
N

DESCRIPTION ?
ADDITIONAL COMMENTS (OPTIONAL)

FIELD NAME ?
(carriage return)

CORRECTIONS:
INSERT SOURCE-OF-FUNDS

FIELD NAME ?
AMOUNT AMT

FIELD TYPE ?
S

MAXIMUM VALUE ?
B4

SCALING FUNCTION ?
/

CONSTANT ?
100

DATA REQUIRED ?
Y

DESCRIPTION ?
AMOUNT OF FINANCIAL AID AWARD

CATEGORY NAME ?
(carriage return)

CORRECTIONS:
SHOW

P (1)	FIELD NAME	ADDR	VALUE	DESCRIPTION	STUDENT IDENTIFICATION NUMBER	TYPE LENGTH	DISP SCALE FACTOR
P (1)	STUDENT-ID-NO	SIN	REQUIRED	DESCRIPTION	U	4	0
P (2)	YEAR-AWARDED	YEAR	REQUIRED	YEAR THAT FINANCIAL AID WAS AWARDED (IN YY FORMAT)	0	1	4
P (3)	TERM-AWARDED	TERM	REQUIRED	TERM IN WHICH FINANCIAL AID WAS AWARDED	0C	1	5
CATEGORIES (ONLY)							
	FALL	P	1	FALL TERM			
	WINTER	W	2	WINTER TERM			
	SUMMER	S	3	SUMMER TERM			
P (4)	CLASS-LEVEL	CL	REQUIRED	CLASS-LEVEL	0C	1	6
CATEGORIES (ONLY)							
	FRESHMAN	FR	1	FRESHMAN			
	SOPHOMORE	SO	2	SOPHOMORE			
	JUNIOR	JR	3	JUNIOR			
	SENIOR	SR	4	SENIOR			
P (5)	TYPE-OF-AID	TYPE	REQUIRED	TYPE-OF-AID	0C	1	7
CATEGORIES (ONLY)							
	TUITION	TU	1	TUITION			
	ROOM-BOARD	RR	2	ROOM AND BOARD			
	CASH	CA	3	CASH			
	LOAN	LO	4	LOAN			
	WORK-STUDY	WS	5	WORK STUDY PROGRAM			
	OTHER	OTH	6	OTHER			
P (6)	SOURCE-OF-FUNDS	SOP	REQUIRED	SOURCE OF FUNDS OF AWARDED FINANCIAL AID	0C	1	8
CATEGORIES (ONLY)							
	UNIVERSITY	UNIV	1	UNIVERSITY			
	STATE-GOV	SGOV	2	STATE GOVERNMENT			
	FEDERAL-GOV	PGOV	3	FEDERAL GOVERNMENT			
	INDUSTRY	IND	4	INDUSTRY			
	FOUNDATION	FOUN	5	FOUNDATION			
	OTHER	OTH	6	OTHER SOURCE			
P (7)	AMOUNT	AMT	REQUIRED	AMOUNT OF FINANCIAL AID AWARDED	S	4	9 / 100
P (8)	REMARK*	REM*		ADDITIONAL COMMENTS (OPTIONAL)	E	4	13
	FILE NAME						
	ILLIP:REMARKS*						

CORRECTIONS:
STOP

5.8.2 DEFINITION OF THE STUDENTS DATA SET

<<< MICRO DATA DEFINITION SYSTEM <<<

ENTER NAME OF DICTIONARY FILE.
STUDENTS#

CONCERNING: STUDENTS#
DICTIONARY FILE DCES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.
YES YOU MAY

ENTER MICRO DATA SET NAME.
STUDENTS

DESCRIPTION ?
DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRC MANUAL: DEALS WITH
DEMOGRAPHIC DATA ON STUDENTS. MIKE KAHN

CAN THIS DATA SET BE DESTROYED ?
N

CAN THIS DATA SET BE REPIACED ?
YES

ENTER NAME OF DATA FILE.
STUDENTS

ENTER NAME OF CONVESSION PCREAT FILE.
(carriage return)

HOW MANY DATA SETS WILL BE USING THIS DICTIONARY ?
1

IS THE DATA SET A "XTAB SET" ?
N

SHOULD THE DATA FILE BE SCRAMELED ?
Y

FIELD NAME ?
STUDENT-ID-NO SIN

FIELD TYPE ?
U

MAXIMUM VALUE ?
B4

SCALING FUNCTION ?
(carriage return)

DATA REQUIRED ?
Y

DESCRIPTION ?
STUDENT IDENTIFICATION NUMBER

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
NAME

ABBREVIATION ?
NA

FIELD TYPE ?
C

LENGTH (OR OVERLAP FIELD NAME) ?
24

DATA REQUIRED ?
Y

DESCRIPTION ?
NAME OF STUDENT IN LAST FIRST MIDDLE FORMAT

FIELD NAME ?
ETHNICITY RACE UC B1 0

CATEGORY REQUIRED ?
Y

DATA REQUIRED ?
Y

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
UNKNOWN UNK 1 0

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
WHITE WH 0 1 WHITE

CATEGORY NAME ?
BLACK BL 0 2 BLACK

CATEGORY NAME ?
AMER-INDIAN

ABBREVIATION ?
AI 0 3 AMERICAN INDIAN

CATEGORY NAME ?
SPANISH SP 0 4 SPANISH SURAMED AMERICAN

CATEGORY NAME ?
ORIENTAL OR C 5

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
OTHER OTH 0 6 OTHER

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
SEX SEX

FIELD TYPE ?
CC

LENGTH (OR OVERLAP FIELD NAME) ?
1

CATEGORY REQUIRED ?
Y

DATA REQUIRED ?
Y

DESCRIPTION ?
SEX OF APPLICANT

CATEGORY NAME ?
MALE MA 0 M MALE

CATEGORY NAME ?
FEMALE FF 0 F

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
UNKNOWN UNK 1 U UNKNOWN

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
BIRTHDATE BDAY UC 1 0

CATEGORY REQUIRED ?
N

DATA REQUIRED ?
Y

DESCRIPTION ?
DATE OF STUDENT'S BIRTH IN YYYYMM FCRMAT

CATEGORY NAME ?
MISSING MISS 1 0 DATE OF BIRTH IS MISSING

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
CLASS-YEAR CLYR SC

MAXIMUM VALUE ?
99

SCALING FUNCTION ?
(carriage return)

CATEGORY REQUIRED ?
N

DATA REQUIRED ?
N

DESCRIPTION ?
CLASS YEAR IN YY FCRMAT

100
Data Set Definition

MICRO Version 5.0
October 1977

CATEGORY NAME ?
UNKNCWN UNK 1 -1

DESCRIPTION ?
(carriage return)

CATEGORY NAME ?
(carriage return)

FIELD NAME ?
(carriage return)

CORRECTIONS:
SHOW

TYPE LENGTH DISP SCALE FACTOR			
P(1)	FIELD NAME	ABRP	DESCRIPTION
		VALUE	
P(1)	STUDENT-ID-NO	SIN	STUDENT IDENTIFICATION NUMBER
P(2)	NAME	NA	NAME OF STUDENT IN LAST FIRST MIDDLE FORMAT
P(3)	ETHNICITY	RACE	ETHNICITY
	CATEGORIES (ONLY)	REQUIRED	
	UNKNOWN	0	UNKNOWN
	WHITE	1	WHITE
	BLACK	2	BLACK
	AMER-INDIAN	3	AMERICAN INDIAN
	SPANISH	4	SPANISH SURAMED AMERICAN
	ORIENTAL	5	ORIENTAL
	OTHER	6	OTHER
P(4)	SEX	SEX	SEX OF APPLICANT
	CATEGORIES (ONLY)	REQUIRED	
	MALE	M	MALE
	FEMALE	F	FEMALE
	UNKNOWN	U	UNKNOWN
P(5)	BIRTHDATE	BDAY	DATE OF STUDENT'S BIRTH IN YYMMDD FORMAT
	CATEGORIES	REQUIRED	
	MISSING	0	DATE OF BIRTH IS MISSING
P(6)	CLASS-YEAR	CLYP	CLASS YEAR IN YY FORMAT
	CATEGORIES	REQUIRED	
	UNKNOWN	-1	UNKNOWN

J 4 0
 C 24 4
 JC 1 28
 DEFAULT
 JC 1 29
 DEFAULT
 JC 1 30
 DEFAULT
 SC 1 31
 DEFAULT

CORRECTIONS:

[Description is incorrect in SEX field; entire definition is redone.]
REDO SEX

FIELD NAME ?
SEX

ABBREVIATION ?
SEX

CATEGORY REQUIRED ?
Y

DATA REQUIRED ?
Y

DESCRIPTION ?
SEX OF STUDENT

CORRECTIONS:
SHOW SEX

P(*)		FIELD NAME	APRF	VALUE	DESCRIPTION	TYPE LENGTH DISP SCALE FACTOR	
P(1)		SPX	SPY	REQUIRED	SEX OF STUDENT	CC	1 29
		CATEGORIES (ONLY)					
		MALE	MA	M	MALE		
		FEMALE	PF	P	FEMALE		
		UNKNOWN	UNK	U	UNKNOWN		
						DEFAULT	

CORRECTIONS:

STOP

5.9 CREATION AND MODIFICATION OF THE DIRECTORY ENTRY

There are certain times when a user or definer may need to change the information stored in a particular user's directory file:

- (1) To correct an error in the directory for a specific data set;
- (2) To create a new entry for a data set and dictionary restored from an archival state (i.e., *FS tape);
- (3) To allow dictionaries to be shared among data sets; and
- (4) To create a directory entry for a data set whose data resides on magnetic tape.

The directory creation and modification is executed by entering:

`$RUN ILIE:MICRO.DIR`

There are four commands available in the MICRO.DIR program: APPEND, DELETE, REDC and STOP. Each will be discussed separately.

The APPEND command will cause MICRO.DIR to prompt the user/definer for all of the information contained in the directory:

- (1) Data set name;
- (2) Description;
- (3) Destroyability status;
- (4) Replaceability status;
- (5) Dictionary file name;
- (6) Data file name;
- (7) Conversion format file name; and
- (8) Tape mount information.

Tape mount information is requested only if the data file name is four asterisks (****) which indicates a tape. The tape pseudo-device name must be *M* and any positioning necessary must be done via the POSH keyword. (See MTS Volume 4 for full details.)

The DELETE command must be followed by a data set name. This

will cause the directory entry for that set to be eliminated.

The REDO command will cause the user/definer to be prompted for replacements for the following:

- (1) Master signon-ID;
- (2) MICRO runfile;
- (3) Local command trace file; and
- (4) Master command trace file.

The STOP command terminates the program.

An attention interrupt will return the user/definer to MTS command mode.

5.10 MICRO.DOC PROGRAM

MICRO.DOC is a computer program which generates data set documentation (for the dictionary) outside the MICRO command processor. MICRO.DOC is executed by entering:

```
$RUN IIR:MICRO.DOC [SEINT=<file or device name>]
```

where SEINT=... need not be specified if the user wishes the documentation to be generated on the terminal.

The program will prompt the user for information about the data set name, the title to be used for the documentation, whether the location of the dictionary or data file should be displayed, whether today's date should be displayed and which documentation format should be used. In reply to the latter, DESC generates the documentation described in the DOCUMENT command description, TECH generates the documentation described in the TECHDOC command description and BOTH generates the documentation described in the FULLDOC command description. See the DOCUMENT, TECHDOC and FULLDOC command descriptions in Chapter 9. MTS may be entered in reply to any question after which the user will be returned to MTS command mode. Attention interrupts are handled and, when used, the user will be returned to the beginning of the cycle in which MICRO.DOC queries the user about documentation.

After completing the documentation of the data set, MICRO.DOC will prompt the user to see if an additional data set is to be documented.

In MICRO.DOC, Y for YES and N for NO are acceptable replies.

The "F(n)" notation appearing in data set documentation is sequential numbering of the fields documented and has no other meaning.

5.11 TAPE DATA SETS

Data can be stored on a magnetic tape rather than disk. Tape data sets cannot be created directly by MICRO but require that the data file connected with the desired data set be copied to a tape. The MICRO.DIR program is then run to create a new directory entry. The directory entry for a tape data set includes the \$MOUNT command information so that the user does not have to worry about mounting a tape. Data should be stored on tape only if it is modified infrequently and is very large or if it is used infrequently. MTS disk files should be used when possible. An alternative technique for infrequently used data sets is to store them on *FS (filesave) tapes as described in Volume 2 of the MTS Manual and to restore them before using MICRO.

CHAPTER 6

PUTTING DATA INTO A DATA SET

Data can be converted to MICRO internal format by one of several procedures. Some of these can be run from the MICRO command mode and others are stand-alone programs. Some accept data only in fixed-format and others accept free-format. The data entry methods are listed below.

(1) Free-Format

- (a) Interactive prompting method: the ENTERDATA command or the stand-alone program MICRO.ADD is used.
- (b) Batch free-format method: the ENTERDATA command or the stand-alone program MICRO.ADD is used to read data from a user-specified file.

(2) Fixed-Format

- (a) User-specified batch fixed-format method: the stand-alone program MICRO.FORM is run to gather information about the format of the input data which is placed in the conversion format file. The stand-alone program MICRO.CNVRT performs the conversion of data to MICRO internal form according to the dictionary and the conversion format file.
- (b) Custom programming method: a program written by the user in any language the user wishes (frequently assembly language) is used to perform the conversion of data to MICRO internal format.

General information on all data entry methods follows. The advantages and disadvantages of each method appear at the beginning of the section describing each method. Executed examples of all data entry methods appear in Appendix A. For details on all the free-format data entry techniques, the user is referred to the MICRO command ENTERDATA. The fixed-format data entry methods are described below in detail.

6.1 INTERACTIVE PROMPTING METHOD

The interactive prompting technique can be used by users of any technical level. No programming skills are required. This technique uses the definition of each field in the dictionary to

prompt the user for values for each field, one field at a time. The dictionary definition is also used to limit the values entered for each field (via the category required status, data required status, etc.; see Chapters 4 and 5).

6.1.1 ADVANTAGES

The user is prompted for all data and is prevented from entering data that has been defined as unacceptable. There are no fixed-format requirements that the user must follow. This method is ideal for data entry from a terminal. No programming skills are required and the training time is minimal.

6.1.2 DISADVANTAGES

This is the most costly of the four techniques. The MICRO.ADD program or MICRO must be loaded during data entry. The technique becomes routine after awhile and many users no longer need the repetitive structure inherent to this technique (see Section 6.2).

6.1.3 TECHNIQUE

The interactive prompting method is available either via the ENTERDATA command from MICRC or the stand-alone program in the file ILIR:MICRO.ADD. Both are described in detail in the ENTERDATA command description in Chapter 9.

6.2 BATCH FREE-FORMAT METHOD

The batch free-format technique is very similar to the interactive prompting technique. However the data is read from a file and the user is not prompted for each value. The person entering the data must know the order in which the fields appear in the data set as this is the order in which they are processed. Thus, the data values can be placed into a file in that order for later processing. All of the records stored in a file in this manner are converted in the same process (i.e., in a batch mode). As in the interactive prompting technique, there are no fixed-formats the user must follow in preparing data. Experienced users of the interactive prompting technique will immediately be able to use the batch free-format technique. Data validation is performed as in the interactive prompting technique.

6.2.1 ADVANTAGES

Data validation is performed according to dictionary specifications and there are no fixed-format requirements. Because the MICRO.ADD program or MICRC is not loaded while data is being entered into the input file, data entry is less expensive than in

the interactive prompting technique. This technique is especially efficient for entering data which is not already in a machine readable format when the skills of a keypunch operator are not available. A record of the data input remains in the file. Records containing errors are isolated.

6.2.2 DISADVANTAGES

This technique is more expensive (usually several times) than the custom programming technique (Section 6.9).

6.2.3 TECHNIQUE

The batch free-format technique is available either through the ENTERDATA command from MICRC or the stand-alone program in the file ILIR:MICRO.ADD. Both are described in detail in the ENTERDATA command description in Chapter 9.

6.3 USER-SPECIFIED BATCH FIXED-FORMAT METHOD

The fixed-format data entry programs provide the cheapest way (in terms of personnel costs and computer costs) of entering most data collections into the MICRO system. These programs are self-contained and are not executable from the MICRO command language. Since the programs require information from the user's directory and dictionary for the data set being initiated, the program ILIR:MICRO.DEF must be executed before ILIR:MICRO.FORM.

6.3.1 ADVANTAGES

Since no program must be written, data conversion costs are reduced. Complete data validation is performed. Real number input is supported. Data recoding can be done.

6.3.2 DISADVANTAGES

This method is not as flexible as a custom program and may require more computer time than the production run of a custom program. Data must be in a fixed-format. The definer must have knowledge about data processing concepts and should have knowledge about MTS.

6.3.3 TECHNIQUE

The first program is executed by issuing the following MTS command:

```
$RUN ILIR:MICRC.FORM SPUNCH=bad-records-file
```

The program will prompt the definer for all information required. If the user is merely specifying format information and is not ready to do the actual conversion from the input data to the MICRO data set, the SPUNCH parameter can be omitted. This assignment is only required for CONVERSION mode (see section 6.8).

6.4 GENERAL RULES FOR BATCH FIXED-FORMAT INPUT

The general rules for using ILIR:MICRO.FORM are:

- (1) The definer may return to MTS at any point in the definition process by typing "MTS" in column 1 in response to any request.
- (2) The definer may give an MTS command at any time. A reply is considered to be an MTS command whenever a "\$" is typed in Column 1. There is one exception. \$ENDFILE is treated as an end-of-file indicator and not an MTS command.
- (3) If the definer wants to send either "MTS" or a reply beginning with a dollar sign to the definition program, input must begin in Column 2 or later.
- (4) Excepting (1) through (3) above, all other replies may be made free of fixed-format requirements.
- (5) The definer may anticipate the prompting for field and recode information by typing several answers on the same line separated by blanks. Prompting will be suppressed for each item until the program has used all of the information provided or until an error is found.
- (6) The definer may type long replies (such as descriptions) on more than one input line by typing a minus sign (-) as the last character transmitted in the line to be extended. The minus sign will not appear in the reply.
- (7) A timer interrupt will occur after 5 CPU seconds have been used. When this happens the program will print

<TIME OUT.....

<USE "RESTART" TO CONTINUE IF YOU ARE MAKING PROGRESS

Then the amount of money spent since signing on will be printed and the definer will be returned to MTS command mode. If the definer restarts the definition program, the timer will be reset to stop again after five more seconds. Experience indicates that a great deal of definition work

can be done in 5 CPU seconds.

- (8) In the following discussion of the definition program, the term "no reply" means sending end-of-file (Control C or "\$ENDFILE") or pushing return without typing any characters. Default values are obtained by giving no reply. When a reply is required the program will try up to five times to get a reply and will then type "Failure to reply" after which the definer will be returned to MTS.
- (9) The definer may issue attention interrupts at any time. Program response will vary:
 - (a) If interrupted before the program requests data format information for the first field, the user will be returned to MTS command mode. The program can be continued by issuing the \$RESTART command.
 - (b) If interrupted during the data format specification mode, the program will ask "Do you want to add more data?" A reply of YES will start the format specification again. A reply of NO will terminate the program. Either reply will result in the incomplete field information being deleted.
 - (c) For small amounts of time, attention interrupts will be ignored to guarantee a valid conversion table at all times.

6.4.1 PROMPTING FOR DATA FORMAT INFORMATION

Information required for the fixed-format program is divided into three parts:

- (1) The MICRO data set name and general information pertaining to the input data records;
- (2) Information pertaining to the data in each field of the MICRO data set and its location in the input data records; and
- (3) Information about possible data recoding to be done for each field.

Each part will be discussed separately. The discussion of (2) applies to each field of the data set.

6.5 GENERAL INFORMATION FOR PATCH FIXED-FORMAT

General data format information includes the following, presented in the order in which the user may be prompted for it:

- (1) MICRO data set name;
- (2) Maximum total number of columns on all cards;
- (3) Maximum number of cards in one record;
- (4) Varying number of cards per record;
- (5) Inclusion or exclusion of certain records; and
- (6) Sampling string.

Each is discussed separately.

6.5.1 MICRO DATA SET NAME

The user must provide either the name of the MICRO data set into which the data will be entered (indicated by the absence of a colon [:] in the fifth column of the reply) or the name of an existing MICRO dictionary file (indicated by a four character signon-ID followed by a colon, followed by the dictionary file name). If the reply is a dictionary file name or if no conversion format file name was provided when the data set was defined, MICRO.FORM asks for the name of a conversion format file. If the conversion file does not exist, the program will ask permission to create it. The conversion file will contain the output of the MICRO.FORM program, i.e. the conversion tables necessary to perform the actual data input and conversion. The data set definition must be complete at this time. If no reply is given the program will terminate.

6.5.2 MAXIMUM TOTAL NUMBER OF COLUMNS ON ALL CARDS

The maximum total number of columns on all cards is requested. If there is only one card or line of data per record, this is the maximum record length (number of columns) for any card. If there is more than one card of data per record, this is the total number of columns on all cards. A reply is required.

6.5.3 MAXIMUM NUMBER OF CARDS IN ONE RECORD

The program starts with the first card and reads in n cards for each record, where n is the response to this question. A reply is required.

6.5.4 VARYING NUMBER OF CARDS PER RECORD

If the number of cards per record is greater than one, the

program asks the question, "Will the number of cards per record vary?" A reply is required. If the answer is "NO" the program will proceed to the next major item of information (see section 6.3.3.5). If the response is "YES", the program will prompt for the method of determining record boundaries. The user then enters one of the following three-letter abbreviations:

- (1) SUB User-supplied subroutine. The subroutine must read a record into a buffer. Arguments to the subroutine are the buffer (supplied by the program), maximum record length (supplied by the program), and real record length (returned by the user subroutine) in that order. Return codes are: 0=successful read; 4=record is invalid and should be rejected; 8=no more records, end of file. The user will be prompted for the subroutine name.
- (2) EOB End of block indicator. When the program encounters the specified string in the specified columns, it will treat the card being read as the last one in the record. The user will be prompted for the starting column and length of the EOB indicator and the identifying string.
- (3) BOB Beginning of block indicator. When the program encounters the specified string in the specified columns, it will treat the card being read as the first one in the record. The user will be prompted for the starting column and length of the BOB indicator and the identifying string.
- (4) CHK Changing key. This method can be used when every card for the record contains a key that is unique for that record. When the key on a card is different from the key on the previous card, the program treats the new card as part of the next record. A Social Security number, for example, might be used to tie together into one record all the cards containing information about one person. The user will be prompted for the starting column and length of the key field.

6.5.5 INCLUDING OR EXCLUDING CERTAIN RECORDS

By entering "E" the user can exclude all records that have a specified string in specified columns. By entering "I" the user can include only records that have a specified string in specified contiguous columns. No reply can be entered to ignore this feature, thereby including all records that are read. If the response is "I"

or "E", the program will prompt the user for starting column, card number and length of an identifying string, as well as the string itself.

6.5.6 SAMPLING STRING

The user enters a sampling string of zeros and ones. The program will include records that correspond to ones in the sampling string and exclude records that correspond to zeros. The pattern is repeated until an end-of-file is reached. For example, if the sampling string is 110, the first two records of every three records are included creating the data set with a two-thirds sample. Entering a carriage return will ignore this feature, thereby including all records after they have been selected by the "I" or "E" feature described in the previous section.

6.6 INFORMATION PERTAINING TO THE DATA IN EACH FIELD - FIXED-FORMAT

For each field of a data set, the user is given the name of the field and then is prompted for information in the order shown below:

- (1) Starting column
- (2) Card number
- (3) Input Type
- (4) Length
- (5) Missing Data Value

Responses for these prompts may be typed on one line, separated by blanks. Each is discussed separately.

6.6.1 STARTING COLUMN (OR SUBROUTINE FILE)

The starting column is the column of the input data card in which the data for the field starts. The definer may gain some of the flexibility of a custom program by writing a subroutine which returns a value for a field. The use of a subroutine is indicated by replying with a four character signon-ID followed by a colon and the name of the object file containing the subroutine. The definer is then asked for the name of the subroutine. Arguments to the subroutine are the buffer containing the entire record, the true length of the record, the number of the field for which the subroutine is called, the name of the field for which the subroutine is called (all supplied by the program), and the value for the field (returned by the subroutine). The value returned is validated and recoded as usual. If the subroutine returns with a return code of four then the record is considered to be in error and is rejected.

If a subroutine file and name are provided, the program proceeds to the next field.

A reply is required for this question.

6.6.2 CARD NUMBER

Card number information is requested only if the maximum number of cards in one record is greater than 1. The reply indicates in which card the data for the field appears. A reply is required.

6.6.3 INPUT TYPE

If character data (FIELD TYPE=C OR CC) is specified in the dictionary, no input type information is requested, and the program prints the message "CHARACTER INPUT ASSUMED". The same is true of external data fields (FIELD TYPE=E). If the data is numerical, the program will ask for input type, and the user should enter one of the following one-letter abbreviations:

- (1) B - Binary - Data for this field consists of a string of ones and zeros, not as characters, but in internal binary form. For signed fields (types S and SC) the incoming data is considered signed. Data for unsigned fields (types U and UC) is not signed.
- (2) I - Integer - Data for this field is a positive or negative integer.
- (3) R - Real - Data for this field is a positive or negative real number with a decimal point. The decimal point may occur in any column which is part of the number. Although MICRO does not handle real numbers, this program actually creates a floating point value for the incoming number. After performing any optional floating point arithmetic operations the final floating point value is rounded (not truncated) to the nearest integer which is then stored in the data set. By multiplying real values by a power of ten and using the division scale function precision may be retained in MICRO. See Section 4.3.7.
- (4) X - Hexadecimal - Data for this field is read as hexadecimal characters. The incoming data is considered signed only if the field type is also signed (types S and SC). Use of this input type will be rare.

6.6.4 LENGTH

The length is the number of columns that the data for the specific field occupies in the input record.

6.6.5 MISSING DATA VALUE

The incoming data for a field is considered missing if columns contain all blanks or all minus signs. When data is missing, the value used for the field is the missing data value specified here by the definer. Character fields are filled with the first character of this value. If no missing data value is provided here then the value of the first default category (if any) is used. If there is no default category for the field, then zeros are placed in numeric fields and blanks are placed in character fields. A reply is not required for this question. No reply indicates that no missing data value is being specified.

6.6.6 RECODE TYPE

The user may wish to change certain data values or ranges of values to some other value for use in MICRO. This can be done for each field by specifying an appropriate recode type to be changed, and the new value. It is also possible to change the value of a field to a specified value for all records, to provide increasing key values for an index field and to do simple arithmetic operations on the data in a field. The program continues to prompt the user for more recode types after each set of recode information has been entered. If the user enters a carriage return instead of a recode type, no more recoding is performed. The information for a recode may be placed on one line with each item separated by blanks. The eight recode types and other information that must be supplied with them are shown below. After any of the following recode types are entered by the user, the program prompts for OLD VALUE and NEW VALUE.

- (1) EQ If the data in the field is equal to the OLD VALUE, the data value is changed to NEW VALUE.
- (2) LE If the data in the field is less than or equal to OLD VALUE, the data value is changed to NEW VALUE.
- (3) LT If the data in the field is less than OLD VALUE, the data value is changed to NEW VALUE.
- (4) GE If the data in the field is greater than or equal to OLD VALUE, the data value is changed to NEW VALUE.

- (5) GT If the data in the field is greater than OLD VALUE, the data value is changed to NEW VALUE.

After the following recode type is entered by the user, the program prompts for a range of INPUT VALUE (two values) and then a NEW VALUE.

- (6) BTW If the data in the field is greater than or equal to the first INPUT VALUE and less than or equal to the second INPUT VALUE, the data value is changed to NEW VALUE.

After the following recode type is entered by the user, the program prompts for INITIAL VALUE and INCREMENT.

- (7) IX The program places INITIAL VALUE into the field in the first record, adds INCREMENT to that value to produce a value for the field in the second record, adds INCREMENT to the value in the field in the second record to produce a value for the field in the third record, and continues in the same manner until reaching the last record of the data set. The default value for both INITIAL VALUE and INCREMENT is one. Default values are assigned if a carriage return is entered instead of a value.

After the following recode type is entered by the user, the program prompts for NEW VALUE.

- (8) ALI The value field in all records of the data set is changed to NEW VALUE.

Whenever the program prompts for NEW VALUE for any of the recode types described above, arithmetic operations can be performed on the data by entering an arithmetic operator (+, -, *, /). The program will then ask the definer to provide a constant. For input types B, I, and X the constant must be a positive or negative integer. However, for real number input the constant may be a real number since floating point arithmetic is performed. The resulting number is then rounded to the nearest integer value. For example, the following recode sequence would add 3 to all data values in the field greater than 100.

```
RECODE TYPE?  
GT  
CIF VALUE?  
100
```

NEW VALUE?
+
CONSTANT
3

Except for arithmetic operations and the IX recode type, characters may be recoded as well as numbers. The program recodes character fields up to 24 characters in length. Alphabetical order is used in determining the success or failure of the equality tests when character strings are compared.

If the user attempts to specify a recode operation that conflicts with one given previously, the program prints an error message and ignores the latest recode specification. Conflicts occur when the user tries to specify more than one recode for the same data value.

Recode type and other recode information may be entered on one line, with the items separated by blank(s). Only one recode per line is permitted.

6.7 CORRECTION MODE - FIXED-FORMAT

After format information has been given for all fields, the program enters CORRECTION mode. The following commands are available for modifying the format information:

CLCSE
CONVERT
DOCUMENT
DELETE
RECCDE
REDO
STOP

Each command is described in detail on the following pages. The notation of the data-format-specification command descriptions is the same as that used for the MICRO command descriptions and macros and is discussed in Section 9.2.

CLOSE

DATA FORMAT SPECIFICATION COMMAND DESCRIPTION

PURPOSE: To store an up-to-date permanent copy of the conversion table in the conversion format file, thus saving the format specification work completed during the current session.

PROTOTYPE: CLOSE

COMMENT: After closing the conversion format file, this command will return the user to CORRECTION mode.

CCNVERT

DATA FORMAT SPECIFICATION COMMAND DESCRIPTION

PURPOSE: To complete the conversion format file and begin
 converting actual data into a MICRO data set.

PROTOTYPE: CCNVERT

COMMENT: The user is asked for the name of the file
 containing the raw data to be converted. See
 Section 6.8 for a description of the conversion
 process.

DOCUMENT

DATA FORMAT SPECIFICATION COMMAND DESCRIPTION

- PURPOSE:
- (1) To print the data format information for the entire data set;
 - (2) To print the data format information for a single field; and
 - (3) To print recode information for a single range of recodes for a field.

COMMAND SYNONYM: SHCW

PROTOTYPE: DOCUMENT [<field name> [<recode number>]]

Where DOCUMENT will cause format information for the entire data to be printed.

Where DOCUMENT <field name> will cause format information for the specified field to be printed.

Where DOCUMENT <field name> <recode number> will cause the nth set of recode information to be printed for the field specified. n is the recode number.

COMMENT: After documenting, this command will return the user to CCRRECTION mode.

DELETE

DATA FORMAT SPECIFICATION COMMAND DESCRIPTION

PURPOSE: To delete a particular recode specification or all recode specification information for a field, without deleting the field format information.

PROTOTYPE: DELETE <field name> [<recode number>]

Where the presence of <recode number> causes the program to delete the information for the specified recode only. If no <recode number> is specified, information about all recodes of a field is deleted.

COMMENTS: (1) After deletion, this command will return the user to CORRECTION mode.

(2) It is not possible to delete information about a field. Field format information may be changed by using the REDO command.

RECODE

DATA FORMAT SPECIFICATION COMMAND DESCRIPTION

PURPOSE: To specify additional recode specification for a given field.

PROTOTYPE: RECODE <field name>

COMMENT: The program prompts for RECODE TYPE and other recode information, according to the rules described in Section 6.6.6. The user will be continually prompted for recoding specifications until a nc reply (carriage return) is given which returns the user to CORRECTION mode.

REDO

DATA FORMAT SPECIFICATION COMMAND DESCRIPTION

PURPOSE: (1) To redo the format specification for all fields
 in the data set; and

 (2) To redo the format specification for one field.

PROTOTYPE: REDO [<field name>]

Where REDC will cause the program to prompt for
all general data format information for the data
set.

Where REDC <field name> will cause the program to
prompt for data format information for the field
specified.

COMMENT: After redoing, this command will return the user
 to CORRECTION mode.

STOP

DATA FORMAT SPECIFICATION CCMAND DESCRIPTION

PURPOSE: To store the conversion table created during the current session in the conversion format file and to terminate the program.

PROTOTYPE: STOP

COMMENT: After closing the conversion format file the program is terminated.

6.8 CONVERSION MODE - FIXED-FORMAT

Having created a conversion format table in the first part of the FORMAT program, the user can use that table to convert the input data to the MICRO internal form by entering CONVERSION mode. The program transfers from CORRECTION mode to CONVERSION mode when the user enters the command, CONVERT, in response to a prompt for CORRECTIONS. Data may also be converted by using the program MICRO.CNVRT. This program may be accessed by issuing the following MTS command:

```
$RUN IIR=MICRO.CNVRT SCARDS=inputfile SPUNCH=errorfile
```

MICRO.CNVRT prompts on SERCOM for the information which follows. Replies are read from the MTS pseudo-device GUSER.

- (1) MICRO data set name: This may be either the name of an existing data set (indicated by the absence of a colon (:)) in the fifth column of the reply) or the name of an existing MICRO dictionary file (indicated by a four-character signon-ID on which the dictionary is stored followed by a colon and the dictionary file name). If no reply is given the program will terminate.
- (2) Signon-ID of another directory or "CANCEL": Asked only if the data set name cannot be found in the directory that was searched or if that directory does not exist or is not accessible. A four-character signon-ID is all that is required. If "CANCEL" is the reply, the user will be asked for a new data set name and the search for it will resume with the user's directory. A reply is required.
- (3) Data file name: Asked only if a dictionary file name was given instead of a MICRO data set name. The file may (a) not exist, (b) exist but be empty or (c) exist and contain data. If it does not exist, it must not begin with a signon-ID other than the CCID currently being used. If it exists then it must be a sequential file permitted RW (read-write access). If it contains data then the data must be compatible with the dictionary. A reply is required.
- (4) Scramble key: Asked only if the dictionary indicates that the data set is or should be scrambled. The same key will be used to unscramble the data set and to scramble the data set with the added records. The same key is used both to scramble and to unscramble a data set. The user must remember the key because it is not permanently stored

anywhere by MICRO. Garbled data indicates that the wrong key was used to unscramble the data set. A reply is required.

- (5) Name of conversion format file: Requested if the data set was initially defined without a conversion format file or if a dictionary file name was given instead of a MICRO data set name. The file must contain format information for each field found in the dictionary.

The program reads data from the input file and converts it using the conversion format. MICRO.CNVRT enforces the "data required" condition for each field. The converted data is recoded and checked to see that the final value does not exceed the size of the field. The value is checked to see if it is a category value if categories are required for the field.

When data for a field is invalid MICRO.CNVRT identifies the invalid value and the appropriate field and prints a message on SPURNT identifying the error. The processing of the record continues with the next field so that all erroneous values are identified. Then the entire record is written on SPUNCH, one "card" per line.

Converted records are written into the end of the sequential file -RESTDS (created automatically if necessary). When there is no more data to be converted MICRO.CNVRT rereads the converted data from the file, sorts and weeds these records, combines them with any existing data in the file and then writes out the final data set. After completing the data set, MICRO.CNVRT destroys the file -RESTDS and reports the number of records in the original set, the number of records read in, the number of records added, the number of records rejected, the number which were duplicates and were weeded out, and the number of records in the resulting permanent set.

6.9 CUSTOM PROGRAMMING METHOD

Prior to the advent of the other three user-oriented techniques for populating (adding data to) a data set, all data initialization was performed by writing a customized computer program to convert the data. This technique requires moderately sophisticated programming skills.

6.9.1 ADVANTAGES

The custom programming technique provides the greatest control over the data that enters the data set. The most complete data validation can be performed in this mode. The program can perform whatever error checking or recoding is desired. This technique can be the most efficient (cheapest computer cost per record) of the four techniques, especially if the program is a moderately well written assembly language program. This cost includes only the execution cost and does not include the cost of the programmer's time or program compilation and debugging.

6.9.2 DISADVANTAGES

The custom programming technique requires a moderately sophisticated programmer and significantly more preparation time than the other techniques. The personnel cost can easily outweigh the efficiency savings for the initialization of small-to-moderate amounts of data (under 1 million bytes). Data validation must be custom written and interfacing with the dictionary (which is optional) is complicated.

6.9.3 TECHNIQUE

Each record in the MICRO data set must conform to the fixed-length record format described in the dictionary. The raw data must be converted so that each field of data is stored as described in the dictionary:

- (1) At the displacement from the beginning of the record and in the internal format (binary integer or character);
- (2) Of the type associated with the field; and
- (3) In the proper length.

Conversion to valid category values is the responsibility of the programmer as is deblocking the raw data (if that is necessary) and blocking the converted data (which is recommended). It is inefficient but acceptable for the converted records to be stored with one record per block. It is most efficient if there are as many records per block as will fit into 32K (32,767) bytes of memory. The length of blocks with multiple records must be an integral multiple of the length of a single record (as defined in the dictionary), since each record must be the same length. After the converted data is placed into a file as described above, the data can be read into a data set via the USE and READ ARRAY commands.

CHAPTER 7

MODIFYING A DATA SET

Existing fields in a data set may be modified by three MICRO commands (CHANGE, UPDATE and CCMPUTE) in addition to modification by the addition or removal of data. The UPDATE command has an equivalent stand-alone program. Different modification strategies correspond to the use of each command. Each will be discussed separately. It should be noted that except for certain cases of the UPDATE command, all changes appear in the result set, not the original set.

7.1 MODIFYING VIA THE CHANGE COMMAND

The CHANGE command modifies a data set by isolating those records that meet specified criteria (the FIND phrase) and then by applying the specified change to all of these records. The modified records are then combined with the unchanged records and placed in the result set. If the CHANGE command is only used to change a small percentage of the records in the data set, the cost per record changed can be high due to the cost of isolation and combination.

The CHANGE command is used when the UPDATE command is inappropriate.

7.2 MODIFYING VIA THE UPDATE COMMAND

The UPDATE command performs with greater efficiency an operation that can be performed via the CHANGE command. The UPDATE command is more efficient than the CHANGE command when the user desires to supply differing new values to the same set of fields of a data set and when the records to be updated can all be identified by the same (one or more) key fields.

Consider a data set which contains cost information on 10,000 inventory parts. Each record has a part number, part name and a unit cost. If the user desires to change the price on 3,500 parts all of which can be uniquely identified by the part number, then the UPDATE command is ideal. One UPDATE command replaces 3,500 CHANGE commands. Additionally, the part number and new cost can be read from a file which was generated without loading MICRO.

7.3 HOW TO INCREASE THE EFFICIENCY OF THE CHANGE AND UPDATE COMMANDS

If a series of modifications are to be made via CHANGE or UPDATE, the cost per record modified can be reduced by isolating all

of the records to be modified from those that are to remain unchanged. If these records are removed from the original set, the user then has two temporary sets, one containing the records to be changed or updated and one to remain unchanged. The task is completed at a lower cost when the changes or updates are applied to the first of these sets with the result being combined with the second.

7.4 MODIFICATION VIA THE CCMPUTE COMMAND

Modification via the CCMPUTE command is possible only when a given field of all records of a data set is to be modified via the same arithmetic expression.

CHAPTER 8

RUNNING MICRO ON MTS

As indicated earlier, MICRO is designed to be an interactive system run from a terminal. Batch processing is not recommended because error processing is not designed for batch mode. However, batch runs can successfully be made if the commands to be executed have been shown to be without error. Furthermore, MICRO commands can be read from an MTS file or a macro to aid in executing repetitive procedures. This chapter describes MICRO execution on MTS including attention interrupt handling, time limits and the security and data integrity features of MICRO. Users unfamiliar with MTS should consult Volume 1 of the MTS Manual.

8.1 HOW TO RUN MICRO

The following is the prototype run command for executing MICRO on MTS at the University of Michigan:

```
$RUN ILIR:MICRO [O=<user macro library file name>]  
[any additional logical device assignments required  
by user subroutines] [PAR=CMT=<file name>|ON|OFF]
```

The user macro library is optional and discussed in Chapter 10. The PAR phrase is also optional and the command trace facility defaults to OFF. CMT (command trace) is described in Section 2.8.1. Therefore, the most commonly used run command is:

```
$RUN ILIR:MICRO
```

8.2 PREFIX CHARACTERS

When 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 processor has generated the printed line.

- (5) percent sign (%): indicates that the ENTER DATA command processor has generated the line.
- (6) at sign (@): indicates that the UPDATE command has generated the line.

8.3 ATTENTION INTERRUPTS

Generally when the user issues an attention interrupt in MICRO, the current operation is aborted and control is returned to the user in the command mode. However, it is frequently necessary to prevent the normal processing of attention interrupts. In this case MICRO will print "CAN'T RIGHT NOW". If three attentions are serviced during the same command, MICRO prints "MICRO PANIC INTERRUPT" and control is returned to MTS command mode. If the user wishes to continue executing the current command, \$RESTART must be entered.

8.4 COMMAND TIME LIMITS

MICRO allows a maximum time limit for the execution of any MICRO command. The maximum can be changed to any integral value. When the time limit is exceeded MICRO prints "MICRO TIME OUT" and control is returned to MTS command mode. If the user wishes to continue processing for another time period, he must type "\$RESTART". The default time limit is 10 CPU seconds and can be modified by the SET command.

8.5 SECURITY AND DATA INTEGRITY FEATURES OF MICRO

There are several ways by which confidentiality and data integrity are maintained. Some data sets are scrambled and require a password in order to be accessed. Data sets may be accessed only by users who have been permitted access to the directory, dictionary and data files via the MTS \$PERMIT command, as described in Chapters 4 and 5 and in the GET command description in Chapter 9.

8.5.1 LEVELS OF SECURITY

There are three ways in which confidentiality is maintained:

- (1) Knowledge of a system (MTS) signon-ID and appropriate password is required in order to access the Michigan Terminal System.
- (2) Directory, dictionary and data files must be properly permitted for a user to access data sets belonging to another user.

- (3) Finally, for the most confidential information, a data set can be scrambled. A scrambled data set requires an additional password for access. The scramble key (password) is not stored in the dictionary but is required when a data set is to be scrambled or unscrambled (see Section 4.1.9). It should be noted, however, that this facility increases the cost of accessing and maintaining a data set.

8.5.2 MAINTENANCE OF DATA INTEGRITY

Data integrity implies that the data in a data set has not been accidentally or maliciously changed. The security provisions of MICRO and MTS described alone will prevent most malicious manipulation. Protection from accidental loss of integrity is the sole responsibility of the user. Being careful, especially when replacing data sets, is the best recommendation. The use of command trace facility along with liberal use of *FS (MTS file save program; see Volume 2 of the MTS Manual) can aid in recovery from a loss of integrity.

8.5.3 A CHECKPOINT AND RECOVERY PROCEDURE

The DATESTAMP field (from the DATESTAMP command, see Chapter 9) can be used as a checkpoint for possible later use in the restoration of a data set to its status at a certain point in time. This can be accomplished by defensive actions, including the use of the *FS (filesave) program (see Volume 2 of the MTS Manual) and MICRO's batch data entry capability (from a file).

To accomplish this, the raw data is stored in an MTS file. Two MICRO data sets are used. The data sets are identical except that the second has a DATESTAMP field. The user creates only the first data set as the second is created from MICRO.

First, data is entered into an MTS file in the manner described in the ENTERDATA command description. This data is then entered (via ENTERDATA) from the MTS file into the data set created by the user. This data set is then DATESTAMPed to record the date it was entered and then SAVED as the second set.

The raw data file and the dictionary and external data files for two data sets are then saved via *FS. The raw data file and the data file associated with the first data set can then be \$DESTROYed (via MTS).

When subsequent raw data is to be added, it is first entered into an MTS file and then entered (via ENTERDATA) into the first

data set. Then the result set is DATESTAMPed, COMBINED with the second set and finally the first data set is REPLACed by the latest result set. The *FS procedure and file destruction is then repeated as previously described.

This procedure provides a way to reconstruct (or restore) a data set to its status at a certain point in time. The DATESTAMP field contains the date that each record was entered. If the CHANGES or UPDATES are made to the second data set then the DATESTAMP field should be changed to the date that the record was modified. The command trace facility must be used to record the nature of the changes (see Section 2.8.1).

CHAPTER 9

THE MICRO COMMAND LANGUAGE

9.1 SPECIFYING A COMMAND

A MICROC command is specified by the user typing any number of valid characters from the minimum specified (underscored) in the command description through the full command name. A MICRO command can be any number of lines in length. However, no command may exceed 500 characters (including blanks) or 100 symbols. A symbol is any string of characters separated by any of the following characters: blank, comma, vertical bar, less than symbol, greater than symbol, ampersand, exclamation point, question mark, semi-colon, equal sign, and prime symbol. 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 indicated in the prototype of the command description. Each word must be separated by one or more blanks, and
- (c) a period.

9.2 NOTATION USED IN THIS MANUAL

The following conventions are used in all command descriptions 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 between 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) A vertical bar (|) between words indicates that only one of the terms can validly appear in the command. For example, AVERAGE|TOTAL indicates that either AVERAGE or TOTAL 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 required.
- (6) The underscore (_) appearing under a command name indicates the shortest form of the command name that may be used to uniquely identify the command.

Further, these conventions can be used in combination. For example, [AVERAGE|TOTAL] indicates that either AVERAGE, TOTAL or neither would be acceptable.

9.3__SYNONYMS ACCEPTABLE IN MICRO

Within each group, the following symbols may be used interchangeably in a MICRO command:

- (1) DATA SET[S]
DATASET[S]
SET[S]
- (2) CATEGORY
CATEGORIES
CAT
- (3) FIELD
FIELDS
- (4) AND
&
- (5) COMMAND
COMMANDS
- (6) ALSO WHERE
OR WHERE
;
- (7) OR
|
- (8) RESULT
IT
- (9) WHERE
W

(10) SO THAT
SUCH THAT
ST

In addition, MICRO commands may have synonyms in addition to allowing shortened forms. Both appear with that command's description.

9.4 __FORMAT OF COMMAND DESCRIPTIONS

The format of the descriptions is basically the same for all commands. Each command description may contain the following sections, each of which is discussed below:

PURPOSE
COMMAND SYNONYMS
PROTOTYPE
APPLICABLE COMMAND MODIFIERS
IMPLICIT EFFECT ON THE RESULT SET
WHAT HAPPENS
EXAMPLES
COMMENTS
ERROR MESSAGES
NOTEWORTHY USAGE STRATEGIES
SPECIAL DESIGN CONSIDERATIONS

In addition, some command descriptions contain special sections which are unique to those descriptions.

9.4.1 __PURPOSE SECTION

The PURPOSE section states the purpose of the command.

9.4.2 __COMMAND SYNONYMS SECTION

If there are any special synonyms for a command, they are listed in the COMMAND SYNONYMS section. This section contains only different command names that are acceptable and does not repeat shorter forms of the command name (as described in (6) of Section 9.2). The underscored letters of the synonym indicate the shortest form of the synonym that may be used to identify the synonym.

9.4.3 __PROTOTYPE SECTION

The PROTOTYPE section displays the language combinations acceptable for the command. The notation described in Section 9.2 is used. Explanations are given when necessary.

9.4.4 APPLICABLE COMMAND MODIFIERS SECTION

If any command modifiers are permitted, they are identified and described in the APPLICABLE COMMAND MODIFIERS section. (See Section 9.5 on command and field modifiers.)

9.4.5 IMPLICIT EFFECT ON THE RESULT SET SECTION

If the result set from a previous command will be destroyed without the direct request of the user, then this is noted in the IMPLICIT EFFECT ON THE RESULT SET section.

9.4.6 WHAT HAPPENS SECTION

If it is important for the user to know what things MICRO is doing in executing a command (for instance, in order to respond to prompts that are a part of a command), then the behavior of the command will be described in the WHAT HAPPENS section.

9.4.7 EXAMPLES SECTION

The EXAMPLES section contains example commands based on the two example data sets, STUDENTS and FIN-AID. All examples are executed and annotated in Appendix A. The example commands are clustered together into groups and the number in square brackets ([]) at the end of each command indicates the example group of Appendix A or B in which it appears.

9.4.8 COMMENTS SECTION

Any clarification or further delineation on the command will appear in the COMMENTS section.

9.4.9 ERROR MESSAGES

All error messages specifically applicable to the command are listed. A complete list of error messages appears in Appendix E.

9.4.10 NOTEWORTHY USAGE STRATEGIES SECTION

If there are any special strategies for using the command then they are described in the section on NOTEWORTHY USAGE STRATEGIES.

9.4.11 SPECIAL DESIGN CONSIDERATIONS SECTION

If there are special implications affecting the data set design caused by the command then they are described in the section on SPECIAL DESIGN CONSIDERATIONS.

9.5 COMMAND AND FIELD MODIFIERS

Some of the commands permit the command name to have modifiers appended to it. These command modifiers change the behavior of the command. All applicable modifiers to a command are listed in the section entitled APPLICABLE COMMAND MODIFIERS. The action caused by each modifier is described elsewhere in the command description.

Some of the commands allow the field names to be modified by appending a field modifier. These modifiers affect the printing of that field.

9.5.1 RULES FOR COMMAND AND FIELD MODIFIERS

(1) All modifiers must begin with an at-sign (@) followed by the modifier and optionally by an equal sign and a value. There must be no blanks between the beginning of the command name and the end of the last modifier.

(2) More than one modifier can be used simultaneously to modify a command or field by appending the modifiers together, with no separating blanks allowed. If the same facility is specified by more than one modifier or is repeated then the last reference to it will apply.

(3) Invalid modifiers will be ignored without any written message to the user.

CALL

COMMAND DESCRIPTION

PURPOSE: To execute a user or system (MTS) subroutine from MICRO command mode, to optionally pass the data in a data set to that subroutine and to optionally pass user-supplied arguments to that subroutine.

PROTOTYPE:

```
CALL <subroutine name>    [USING <data set name>|  
WITHOUT DATA SET|W/O] [LIBRARY|LIB = <file name>]  
[KEEP=YES|NO] [PARAMETER|PAR=<arg 1>,...,<arg n>]
```

Where <subroutine name> may be any valid subroutine name.

Where USING <data set name> causes the data and basic dictionary information from <data set name> to be passed to the subroutine.

Where WITHOUT DATA SET or W/O cause no data or dictionary information to be passed to the subroutine.

Where the absence of USING <data set name>, WITHOUT DATA SET or W/O cause data and basic dictionary information from the result set to be passed to the subroutine.

Where <file name> is the name of an MTS file which contains the object code of the desired subroutines or of a library of subroutines. (See the *GENLIE description in Volume 2 of the MTS Manual).

Where the absence of LIBRARY or LIB indicates that the default library named CCID:LIBRARY containing either object code or a library will be referenced, where CCID is the master signon-ID.

Where KEEP=YES or the absence of a KEEP clause indicates that the subroutine is to remain loaded in memory after completion of the subroutine's execution. If KEEP=NO is specified, then the

subroutine will be unloaded after its execution is completed.

Where `PARAMETER|PAR=<arg 1>,...,<arg n>` is used to pass additional user-supplied parameters (arguments) to the subroutine.

The CALL command will generate the following FORTRAN (IBM type S) type call:

```
CALL <subroutine name> ([NWORDS, IFINFO,  
NCATS, ICINFO, NUMREC, IDATA] <arg 1>,...,  
<arg n>).
```

Where:

- NWORDS is an integer variable containing the width in fullwords of each data record in IDATA passed to the subroutine.
- IFINFO is an integer matrix of two dimensions: IFINFO(6,NWORDS). That is, there are six words of information in IFINFO for each fullword of a data record in IDATA. The first four fullwords of information (i.e., IFINFO(1,I) through IFINFO(4,I), as I goes from 1 through NWORDS) contain the 16-character field name. The fifth fullword (IFINFO(5,I)) contains the number of categories in the associated field. The sixth fullword (IFINFO(6,I)) contains an index into the ICINFO matrix which contains category information for the associated field. When the fifth and sixth fullwords are zero, there are no categories for the associated field.
- NCATS is an integer variable containing the total number of categories and external data file names; i.e., the sum of IFINFO(5,I) where $1 \leq I \leq \text{NWORDS}$.
- ICINFO IFINFO is an integer matrix of two dimensions: ICINFO(4,NCATS). It contains category and external data file information. The first three fullwords of each category (that is,

for each integer value of I, as I goes from 1 through NCATS) ICINFO(1,I) through ICINFO(3,I), contain the category name. The twelfth character is always blank because category names are less than or equal to eleven characters in length. The fourth fullword, ICINFO(4,I), is the value of the associated category. The only exception is for external data fields, for which ICINFO(1,I) through ICINFO(4,I) contain the file name associated with the external data field. The file name is in the format CCID: filename, which can be up to eleven characters in length. The twelfth byte is not used.

NUMREC is an integer variable representing the number of records from the data set passed to the subroutine, i.e., the number of records in the specified data set.

IDATA is a two-dimensional integer matrix of data: IDATA(NWORDS,NUMREC)).

<arg i> may be a fullword integer, a doubleword floating point (REAL*8) number or string of up to 24 characters.

APPLICABLE COMMAND MODIFIERS: None

EFFECT ON THE RESULT SET: None

- COMMENTS:
- (1) All integer field values are automatically expanded to fullword integers in IDATA.
 - (2) All character fields are expanded by padding with blanks on the right so that they are a multiple of four characters long (fullword) in IDATA. For example, a character field of length 17 would be expanded to 5 fullwords or 20 characters (bytes).
 - (3) The order in which the keywords appear in the command is arbitrary.
 - (4) If <arg i> is to be represented as a floating

point number, it must contain a decimal point.

- (5) Only a portion of the dictionary information is passed to the subroutine. The subroutine itself must contain any other desired information. The INDEX subroutine shown in Appendix B allows the subroutine to locate a specified field in the IFINFO matrix and eases the programming task of associating the passed dictionary information with subroutine held information. See Appendix E, "Example Subroutine for CALL Command" and Appendix A for the execution of this example subroutine.

- (6) DATA SET and DATASET are synonyms.

EXAMPLES: CALL TIME WITHOUT DATASET PAR=5,1. [B-15]

CALL CALLEX USING GE500 LIB=ILIR:DEMOLIB KEEP=NO
PAR=1, 2.C5, STRING. [B-15]

ERROR MESSAGE: 'ERRORS DETECTED'
Indicates that an error of an unspecified nature occurred.

CHANGE

COMMAND DESCRIPTION

PURPOSE: To create a RESULT set in which the data in specified fields is altered in all or certain records of the specified data set.

COMMAND SYNONYM: ALTER

PROTOTYPE: CHANGE IN <data set name> [WHERE <find phrase> |
[ALL RECORDS]] SO THAT [<destination field name>
<assignment verb>] <replacement phrase> [AND
<destination field name> <assignment verb>
<replacement phrase> ...].

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

Where ALL RECORDS (or the omission of <find phrase>) indicates that the changes are to be made to all records of <data set name>.

Where <destination field name> must be the name of an existing field in <data set name>.

Where <assignment verb> can be any of the following:

=
IS
ARE
IS EQUAL TO
EQUAIS
EQUAL
IS =
ARE =

"match" verbs:

MATCHES
MATCH
MAT
MA

The "match" verbs can be used to change only the

first N characters or a character field where N is equal to or less than the number of characters in the field. This is analogous to the use of the "MATCH" verb in the FIND command. See the FIND command description for an explanation of the match verbs.

where <replacement clause> is the following:

A value or previously defined category name (results in a direct replacement).

If <destination field name> and the <assignment verb> following it are omitted, the last <field name> mentioned in <find phrase> is assumed to be <destination field name> and the <assignment verb> is assumed to be an equal sign.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is lost.

WHAT HAPPENS:

- (1) The data set is divided into two intermediate sets, the first containing all of the records to be changed as identified by <find phrase> and the second containing all other records.
- (2) The change(s) are then made to the first intermediate set.
- (3) The two intermediate sets containing the changed records and the unchanged records are COMBINED and the result is placed in the result set.

COMMENTS:

- (1) SUCH THAT, SI and TC are synonyms for SO THAT.
- (2) The number of records changed is printed. The percentage of changed records in <data set name> is also printed.
- (3) The result set usually contains the same number of records as the original data set. However, loss of records can occur if the change(s)

(4) The character strings in a file associated with an external data field cannot be changed via the CHANGE command. However, the value of the external data field (the line number or note pointer) can be changed. The UPDATE command will put new character strings into a file and appropriately change the external field value. The only way to delete or edit character strings is by using the MTS editor.

CHANGE IN IT SO THAT YEAR=77. [B-13]

```

*ERROR - FIELD TO FIELD CHANGE IS NOT
IMPLEMENTED.*
Indicates that a request for a field to be
replaced by another is not allowed - see COMPUTE
command.

```

'ERROR - NC CONDITION SPECIFIED!'
Indicates that the command was incomplete in that
no criteria was specified when one was required.

'FIELDS ARE NOT COMEATIBLE.'

Indicates that a field to field comparison cannot be performed because the two fields are not comparable.

'INTERVAL FOR < CR > IS LOGICALLY EMPTY.'

MICRO assumes that when an empty interval is specified, i.e. AGE < 0, that the user has made an error and that the command is cancelled.

"IS NOT BETWEEN" IS NOT AVAILABLE.
"IS NOT BETWEEN" is not a defined operation.

NOTEWORTHY USAGE STRATEGIES:

- (1) CHANGing records is one of MICRO's least efficient operations. The UPDATE command is more efficient if changes are to be made to the same field(s) in many records.
- (2) The CHANGE command processes the changes from left to right. Multiple replacement clauses (such as YR=76 AND CLASS=FRESHMAN AND...) should be grouped together if it is at all possible because they may be replaced in the same pass through the data. Field-to-field replacements and computational replacements each require a separate pass through the data.
- (3) If a series of modifications are to be made via CHANGE or UPDATE, the cost per record modified can be reduced by isolating all of the records to be modified from those that are not changed. If these records are then removed from the original set, the user can construct two temporary sets, one containing the records to be changed or updated and one to remain unchanged. The task is completed at a lower cost when the changes or updates are applied to the first of these sets and the result of the change or update being combined with the second set.

SPECIAL DESIGN CONSIDERATION:

Only the first 24 characters of character type fields may be changed by using this command. However the UPDATE command will handle character fields of up to 255 characters in length.

CCMBINE

COMMAND DESCRIPTION

PURPOSE: To merge the data from two identically structured data sets, that is, to create the union of two identically structured data sets.

PROTOTYPE: CCMBINE <data set name 1> WITH|AND
<data set name 2>.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result is created and the old result set, if one existed, is replaced.

WHAT HAPPENS:

- (1) The two data sets are compared for compatibility. A field by field comparison is made to see that the field type and size correspond in both sets. Both sets must be either XTAB or non-XTAB.
- (2) If the two data sets are compatible and neither is an XTAB set, the data associated with both sets is merged together and redundant records are weeded out.
- (3) If the two data sets are compatible and both are XTAB sets, the data associated with both sets is merged. Those records which have the same combination of values for all fields (ignoring the CCOUNT field) are reduced to one record having that combination of values with the COUNT field representing the sum of the COUNT fields of the redundant records.
- (4) The new data is placed in the result set using the dictionary from <data set name 1>.

EXAMPLES:

COMBINE FROSH AND EVENYEARS.	[B-6]
CCMBINE IT WITH STUDENTS.	[A-6]
CCMBINE NO-AID AND GOT-AID.	[B-9]

COMBINE TEMP2 AND TEMP3.

[B-8]

ERROR MESSAGES:

'INCOMPATIBLE DATA SETS - FIELD DISPLACEMENT.'

Indicates that the two specified sets have at least one field occurring in a different location.

'INCOMPATIBLE DATA SETS - FIELD LENGTHS.'

Indicates that the two specified sets have at least one field of differing length.

'INCOMPATIBLE DATA SETS - # OF FIELDS.'

Indicates that the two specified sets have a different number of fields.

'INCOMPATIBLE DATA SETS - SET WIDTH.'

Indicates that the data of the two specified sets contain a different number of bytes.

'NO DATA SET #1 SPECIFIED.'

'NO DATA SET #2 SPECIFIED.'

'TOO MANY DATA SETS SPECIFIED.'

Indicates that more than two data sets were specified.

COMMENT

COMMAND DESCRIPTION

PURPOSE: To allow a comment to be entered during a MICRO session with no resulting action being taken.

COMMAND SYNONYM: *

PROTOTYPE: COMMENT <character string>.

Where <character string> may be any character string up to 500 characters in length.

No action is taken, the input is ignored.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENTS: (1) There must be an even number of primes within the character string, if primes are used.

(2) The comment must end with a period.

NOTEWORTHY USAGE STRATEGIES:

(1) This command is useful to denote progress in procedures that are run repetitively. See the SET INPUT description in the SET command and Chapter 10.

(2) This command may be useful in recording notes to be logged via the command trace facility. See Section 2.8.8.

EXAMPLES: COMMENT - THIS RUN IS BEING MADE BY MIKE KAHN.

*DATA SET NEEDS TO BE PERMITTED FOR OTHERS TO USE.

COMPUTE

COMMAND DESCRIPTION

PURPOSE: To store the result of one or more arithmetic operations in an existing field or in a newly created field.

PRCTOTYPE: COMPUTE in <data set name> [SO THAT|SUCH THAT|ST]
<destination field name> = <field name 1> |
<constant 1> [<operand> <field name 2>
| constant 2> ...].

Where SO THAT, SUCH THAT, and ST are optional phrases which may be added for clarity.

Where <destination field name> is either the name of any integer field which exists in the specified data set or a new (non-existent) field to be created as a signed fullword integer field.

Where <operand> can be any of the following:

- ' + ' for addition,
- ' - ' for subtraction,
- ' * ' for multiplication, and
- ' / ' for division.

It is essential that the operator be prefixed and suffixed by at least one blank.

APPLICABLE COMMAND MODIFIERS:

@KC (Keep categories); if the destination field is either a signed or unsigned integer with categories, the categories are not dropped in the resulting field.

@NSC (No scaling change); applies to existing fields only: the scaling of the destination field in the result set will be unchanged even though intermediate arithmetic operations would normally change it.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the previous result set, if one existed, is released.

CREATING A NEW FIELD:

If @KC is not specified, the new field created will be a fullword (four byte) signed integer field without categories. If @KC is specified, the new field created will be a fullword signed integer with categories. No other field types and lengths can be created through the COMPUTE command. The scaling factor and function of the new field is a function of the operands, fields and constants specified in the command and is discussed below.

The abbreviation for the field name of the newly created field is the first four characters of the new name. If the new name is four or fewer characters, the abbreviation is the same as the field name. A check is made to see if the new name is the same as an existing field name or abbreviation or any category name or abbreviation; if it is, then the user will be prompted for another name. The abbreviation is not checked to see if it duplicates any of the above names or abbreviations.

COMPUTING INTO AN EXISTING FIELD:

Two special situations can occur when the destination field is an existing field in the specified data set. First, when the destination field is an unsigned integer the field type is automatically changed to signed integer without notice to the user.

Second, if the destination field is one to three bytes in length and the result of a computation is too large or too small to fit into the space associated with the signed destination field, the message "RESULT OF COMPUTATION EXCEEDS DESTINATION FIELD'S MAXIMUM VALUE. ENTER A NEW DESTINATION FIELD NAME, OR 'CANCEL'." is printed. The name may be an existing field name or a new one to be created as a fullword signed integer in the result set. In this case, the original destination field with its original values will be included in the result set, i.e., it will not be deleted. If the user enters "CANCEL" the command is cancelled and the result

set is unchanged. If the destination field is four bytes long and the result of the computation is too large or too small to fit in four bytes an erroneous result will be stored in the destination field and the user will not be informed that this has occurred. (See "Overflow" below.)

SCALING FACTOR OF THE DESTINATION FIELD:

The scaling factor and function of an existing destination field may change depending on the operands (fields and constants) which are on the right side of the equal sign in the COMPUTE command. Similarly, the scaling factor and function of a newly-created field is determined by the operands.

The scaling factor is affected due to an attempt to preserve significant digits. This occurs in a variety of ways described by the following rules:

- (1) If fields or constants are added or subtracted, the destination field will attain the largest number of significant digits attributed to any of the fields or constants. For example, if NEWFIELD was computed as $525 + 1.05 - .003$, NEWFIELD would be equal to 526.047 but would be stored as 526047 with a scaling function of division and a scaling factor of 1000 (equivalent to three decimal places).
- (2) If fields or constants are multiplied, the destination field will also attain the highest level of precision. For example, if NEWFIELD was computed as $14 * 1.00$, NEWFIELD would equal 14.00. It would be stored as 1400 with a scaling function of division and a scaling factor of 100 (equivalent to two decimal places). The multiplication of a large number by a number with several significant decimal places can result in an overflow condition. For example, $1,000,000 * 0.9999999$ is stored as the integer 9999999000000 which is too large to fit into a fullword. Overflow is discussed below.

- (3) Division does not change the scaling of the resulting field. Answers are truncated to an integer value without any change in scaling.
- (4) By performing certain multiplication operations, the number of significant digits associated with the destination field can be intentionally increased. For example, to create a new field scaled to three decimal places with the values equal to an existing unscaled integer field, the following arithmetic operation should be performed: $NEWFIELD = CLDFIELD * 1.000$. If a value of CLDFIELD was 789, then it would become 789.000 in NEWFIELD. Division by a number with insignificant trailing zeros (such as 1.000) does not affect the scaling. Thus, if the destination field of a division operation is to have one or more decimal places of precision, it is necessary to first multiply the numerator by one followed by the appropriate number of zeros and then divide by the desired number. For example, INTEGERFIELD is an integer field without scaling. To have the answer with three decimal places of precision: divide INTEGERFIELD by 7 and $NEWFIELD = INTEGERFIELD * 1.000/7$. If a value of INTEGERFIELD is 9, the resulting answer would be 1.285; if the multiplication by 1.000 is omitted, the resulting answer would be 1.
- (5) There is no way that precision (the number of decimal places) can be reduced (truncated).

CATEGORIES OF NEW DESTINATION FIELDS:

If the destination field does not exist, then the newly-created field will not have categories unless the KC (keep categories) modifier is appended. In this case, the categories of the first field on the right of the equal sign will be attached to the new field. If the first existing field is less than four bytes in width, then the category values in the dictionary will

be expanded (binary digits shifted left with trailing zeros) to fullword size. This expansion is transparent to the user.

OVERFLOW:

Overflow is a condition which occurs when the space required to represent a number is greater than the space allocated. For example, the largest unsigned integer that can be represented in a single byte is 255, which is represented in the eight-bit binary string as 11111111. If 1 is added to that number, the result is 256, or 100000000 in binary. Because only eight bits of storage are available for representation in a single byte, the rightmost eight bits, 00000000 (zero) are stored. The leftmost bit has "overflowed" and is lost, and the value represented (zero) is incorrect. Overflow can also occur when a negative number becomes too large to be stored in the allotted space. The minimum and maximum numbers that can be represented by the storage associated with integer fields in MICRC are:

<u>Unsigned Integers</u>	<u>Minimum</u>	<u>Maximum</u>
1 byte	0	255
2 bytes	0	65,535
3 bytes	0	16,777,215
4 bytes (fullword)	0	2,147,483,647

<u>Signed Integers</u>	<u>Minimum</u>	<u>Maximum</u>
1 byte	-128	127
2 bytes	-32,768	32,767
3 bytes	-8,388,608	8,388,607
4 bytes (fullword)	-2,147,483,648	2,147,483,647

The user should remember that scaled fields are stored as integers with an implied decimal point.

Overflow can occur as a result of any arithmetic operation performed by the COMPUTE command.

UNDERFLOW:

Underflow occurs only in the division operation when the quotient is too small to be represented in the space allocated. In this case, the erroneous value in the result set will be 0. In a one-byte unsigned integer field without scaling, the smallest number that can be represented is zero. If 5 is divided by 10, the

value stored will be zero.

OTHER COMMENT: IS is synonymous with the equal sign (=) in the prototype and may be used in its place.

EXAMPLES: COMPUTE IN IT MAX.AMOUNT = MAX.AMOUNT * 1.00. [B-18]
COMPUTE IN IT MIN.AMOUNT=MIN.AMOUNT * 1.00. [B-18]
COMPUTE IN IT NEWCODE=-447. [B-13]
COMPUTE IN IT ST INCREASE=NEW.AMOUNT - AMOUNT. [B-13]
COMPUTE IN IT SUCH THAT KEY = KEY - 85 * -1. [B-16]
COMPUTE IN RECIPIENTS ST NEW.AMOUNT=AMOUNT * 1.05. [B-13]

NOTEWORTHY USAGE STRATEGY:

The COMPUTE command can be used to add to the end of the data set a new four-byte signed integer field with a new name and any arbitrary signed integer value. The COMPUTE, CROSSTABULATE, DATESTAMP, JOIN, RESTRICTANDJOIN and KEY commands are the only commands which can be used to add additional fields to a data set which already contains data. (Fields can be added to dictionaries without data by using the data set definition program, MICRO.DEF.) The DATESTAMP command is the only command available to add a three-byte field.

SPECIAL DESIGN CONSIDERATION:

The size and sign of a field can affect the action of the COMPUTE command. The user should consider anticipated arithmetic computation when defining the size and sign of a field, especially with respect to the possibility of overflow or underflow.

CROSSTABULATE

COMMAND DESCRIPTION

PURPOSE: To identify and count the number of occurrences of each unique value combination for the fields specified; to optionally perform statistical computations on additionally specified fields within the unique value combinations of the crosstabulation.

COMMAND SYNONYMS: TABULATE, XTAB

PROTOTYPE: CROSSTABULATE IN <data set name> [<statistical operation>] <field name> [...BY [<statistical operation>] <field name>].

Where <statistical operation> may be any of the following which are grouped by equivalent operations:

AVERAGE	AVE	MEAN
STD.DEV	SD	STD
MINIMUM	MIN	
MAXIMUM	MAX	
TOTAL	TCT	SUM

If the XTAB PERCENT facility is ON then the PERCENT field is computed. If the XTAB PRINT facility is ON then the result set is automatically printed. Both default to ON.

APPLICABLE COMMAND MODIFIERS:

<u>@%</u>	temporarily sets the status of the XTAB PERCENT facility to ON for the execution of this command and then returns it to its previous status.
<u>@NO%</u>	temporarily sets the status of the XTAB PERCENT facility to OFF for the execution of this command and then returns it to its previous status.
<u>@PRINT</u>	temporarily sets the status of the XTAB PRINT facility to ON for the execution of this command and then returns it to

its previous status.

@NCPRINT temporarily sets the status of the XTAB PRINT facility to OFF for the execution of this command and then returns it to its previous status.

If the XTAB PRINT facility is ON, either through the SET command or the PRINT command modifier then any of the following command modifiers will affect the printing of the result set:

@ABBREVIATION
@CCUNT = integer
@FIRST = <integer>
@FCRWARD
@HEXADECIMAL
@HCRIZONTAL
@INCREMENT = <integer>
@INDIRECTION
@PAGE [=<integer>]
@PC
@PE [=<integer>]
@REVERSE
@STRATIFICATION = <integer>
@VALUE
@VERTICAL
@WINDOW = integer
@X

Only these command modifiers will affect the printing of the result set by the XTAB command. Any field modifiers specified will be ignored. However, subsequent use of the PRINT command makes available all the command and field modifiers. See the PRINT Command description for details on each of these modifiers. These modifiers will have no effect if the XTAB PRINT facility is OFF.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

COMMENTS:

- (1) The fields in the result set are ordered as they are specified in the XTAB command with exceptions. Those fields on which the crosstabulation is performed are first, followed by the statistics calculated. These are then followed by the COUNT field and the optional PERCENT field.

- (2) Fields created as a result of statistical operations will always be signed four-byte integers that are fullword aligned.

- (3) The field names for all statistically calculated fields consist of a four-character prefix which identifies the operation performed, followed by the first 12 characters of the field name involved. The prefixes used are:

AVE. for average
S.D. for standard deviation
MIN. for minimum
MAX. for maximum
TCT. for total

The field abbreviation is the same as that of the specified field; i.e., both have the same abbreviation.

- (4) The scaling of fields resulting from the average and standard deviation operations will increase by two decimal places; that is, the current scale factor will be multiplied by 100.
- (5) The scaling of the PERCENT field defaults to division by 100 but may be changed by setting the XTAB% PRECISION facility. See the SET command description.
- (6) Erroneous values will result from statistical operations on signed fields less than four bytes long containing negative values. The user will be warned about this possibility when statistical operations are performed on those fields.

- (7) The abbreviation of the COUNT field generated is CNT. The abbreviation of the PERCENT field generated is %.
- (8) There must be one or more blanks between the <statistical operation> (described in (3) above). And the <field name> to which it refers.
- (9) If the data set being crosstabulated is itself a crosstabulated set, the COUNT fields in the result set will be the sum of the COUNT fields of the records which have been "counted."

EXAMPLES:

X IN EVENYEARS CLASS-LEVEL.	[B-6]
XTAB IN BIGJCIN SOURCE-OF-FUNDS.	[B-10]
XTAB IN FIN-AID CLASS-LEVEL.	[B-6]
XTAB IN FIN-AID TYPE BY YEAR BY TOTAL AMOUNT BY MIN AMOUNT BY MEAN AMT BY MAXIMUM AMOUNT.	[B-6]
XTAB IN FROSH CLASS-LEVEL.	[B-6]
XTAB IN NCT-JR CLASS-LEVEL.	[B-6]
XTAB IN RECIPIENTS AVE AMOUNT, MIN AMOUNT, MAX AMOUNT, YEAR.	[B-18]
XTAB IN STUDENTS RACE BY SEX.	[B-6]
XTABNCPNO% IN FIN-AID TYPE BY AVE AMOUNT.	[B-6]

ERROR MESSAGES:

'CAN'T HANDLE AVE OR TOT FOR XTABBED SET -
SCRRY.'

Indicates that the code to handle statistical operations on an XTABBED set has not been written.

'ERROR: ILLEGAL TO TCT OR AVE A CHAR TYPE FIELD.'

Indicates that a statistical operation was requested on a character type field.

SPECIAL DESIGN CONSIDERATION:

The statistical operations are only supported for unsigned integer fields and four-byte signed integer fields. For one, two, and three-byte

signed integer fields, the numerical representation is treated as a logical representation of an integer. This results in an erroneous answer when any of the values are negative.

DATESTAMP

COMMAND DESCRIPTION

PURPOSE: To add a new field or alter the contents of an existing field such that this field contains the current date.

PROTOTYPES: (1) DATESTAMP <data set name> [AT BEGINNING|END].

Where AT BEGINNING indicates that the new field is to be added as the first field of the result set followed by the fields of the specified data set.

Where AT END indicates that the new field is to be added as the last field of the result set following the fields of the specified data set.

(2) DATESTAMP <data set name> [USING|WITH|BY <field name>].

Where USING|WITH|BY <field name> allows the user to specify the name of a new or existing field as the destination field.

APPLICABLE COMMAND MODIFIERS:

@NV

No verification. This modifier suppresses the verification request that would otherwise be present before altering the contents of an existing field.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result sets, if one existed, is released.

COMMENTS: (1) If the command results in the creation of a new field, then this field will be a 3 byte unsigned integer field with field name being as specified and field abbreviation being the first four characters of the field name. If the destination field is not explicitly stated and a new field is created, then this field will have the name

- (2) If the destination field already exists, then this field must be a 3 or 4 byte unscaled integer field. In this case verification is requested before the contents of the field are altered unless the %NV modifier is specified.
- (3) The existence of a DATESTAMP field overrides the AT BEGINNING|END specification. The existing DATESTAMP field will be used if it meets the above stated conditions as a destination field.
- (4) The date is in the YYYYMMDD format.

NOTEWORTHY USAGE STRATEGIES:

MICRO Information Management System (Version 5.0) Reference Manual

DESCRIBE
COMMAND DESCRIPTION

PURPOSE: To print a description of the specified data set, fields, or categories.

PROTOTYPES: (1) DESCRIBE [DATA SET] <data set name>.

Where DATA SET is an optional phrase added for clarity.

The description of <data set name> is printed.

(2) DESCRIBE ALL [DATA SETS].

Where DATA SETS is an optional phrase added for clarity.

The descriptions of all data sets belonging to the user or acquired through the GET command are printed.

(3) DESCRIBE *.

This is synonymous with (2) above.

(4) DESCRIBE IN <data set name> [FIELD] <field name>.

Where FIELD is an optional phrase added for clarity.

The description of <field name> is printed.

(5) DESCRIBE IN <data set name> [FIELD] <field name> [CATEGORY] <category name>.

Where FIELD and CATEGORY are optional phrases added for clarity.

The description of <category name> is printed.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENTS: (1) Data set descriptions are stored in the user

directory, in the negative line numbers.

- (2) More detailed documentation concerning the contents of the dictionary of each data set can be obtained by using the DOCUMENT, FULLDOC and TECHDOC commands or the stand-alone MICRO.DOC program.

EXAMPLES: DESCRIBE ALL SFTS. [B-1]
DESCRIBE IN STUDENTS BIRTHDATE. [B-1]
DESCRIBE STUDENTS. [B-1]

ERROR MESSAGES: 'DESCRIPTION NOT IN DICTIONARY.'
Indicates that MICRO expected a description but could not find it.

'DESCRIPTION NOT IN FILE.'
Indicates that MICRO expected a description but could not find it.

'*** MICRO ERROR - BAD POINTER.'
Indicates that an erroneous description pointer was used; probably the dictionary is in error.

'THAT CATEGORY HAS NO DESCRIPTION.'
Indicates that no description is defined for the specified category.

'THAT FIELD HAS NO DESCRIPTION.'
Indicates that no description is defined for the specified field.

'THERE ARE CURRENTLY NO DATA SETS AVAILABLE.'
Indicates that the user has no data sets.

'THERE IS NO DESCRIPTION OF THAT DATA SET.'
Indicates that no description is defined for the specified category.

SPECIAL DESIGN CONSIDERATION:

The description provided by MICRO is only as good as the description entered during the dictionary definition process or when a data set is SAVED. If no description was entered at that time then none can be printed.

DESTROY

COMMAND DESCRIPTION

PURPOSE: To permanently eliminate a data set.

PROTOTYPE: DESTROY <data set name>.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

WHAT HAPPENS:

- (1) A check is made by MICRO to see if the data set can be destroyed (see Section 4.1.3). All temporary sets can be destroyed. If a data set cannot be destroyed then an error message is printed and the command is cancelled.
- (2) If a data set is destroyable, the user will be asked to confirm the destruction of the specified data set. To confirm, the user should reply OK. Any other response will result in the command being cancelled.
- (3) When confirmation is given, MICRO will first attempt to destroy the dictionary associated with the data set, then the data file and finally MICRO attempts to delete reference to that data set from the user's directory. If any of these changes are prevented by the MTS permit status of the files, an error message is printed and the command is continued. Therefore, it is possible to partially destroy a data set.

- COMMENTS:
- (1) Once a data set is destroyed, it cannot be referenced again.
 - (2) If the data set to be DESTROYed is a temporary data set then the command is equivalent to a RELEASE command.
 - (3) Data sets referenced through the GET command cannot be destroyed.

- (4) External data files are not affected by the DESTROY command.

EXAMPLES: DESTROY JOINED-SET. [B-19]
DESTROY RECIPIENTS. [B-19]

ERROR MESSAGES: 'YOU ARE NOT ALLOWED TO DESTROY THAT DATA SET.'
Indicates that the user does not have MICRO destroy access to that data set.

'YOU DO NOT HAVE THE PROPER MTS ACCESS.'
'UNABLE TO DESTROY FILE _____ AT THIS TIME.'
'PLEASE DESTROY IT LATER UNDER MTS.'
Indicates that this CCID does not have MTS destroy access to the indicated file and the file must be destroyed from MTS command mode since the references to that file from MICRO will disappear.

DISPLAY

COMMAND DESCRIPTION

PURPOSE: To print the expended computer costs, virtual memory in use and/or current time and date.

PROTOTYPES: (1) DISPLAY CCST|\$.

The estimated cost of activity since the last DISPLAY COST or DISPLAY [ALL] command was issued is printed as well as the total amount spent since signing onto MTS. If there was no previous DISPLAY COST, the cost since MICRO was loaded is printed.

(2) DISPLAY VM|VMSIZE.

The number of pages of virtual memory (core storage) currently in use by this user is printed.

(3) DISPLAY TIME.

The current time and date are printed.

(4) DISPLAY [ALL].

Where ALL is an optional phrase added for clarity.

COST, VMSIZE and TIME are printed.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

EXAMPLE: DISPLAY ALL.

[B-1]

DOCUMENT

COMMAND DESCRIPTION

PURPOSE: To document the user-oriented dictionary information for one or all fields of a specified data set.

PROTOTYPES: (1) DOCUMENT [DATA SET] <data set name>.

Where DATA SET is an optional phrase added for clarity.

The user-oriented dictionary information for all of the fields and categories in <data set name> is printed.

(2) DOCUMENT IN|FOR <data set name> [FIELD] <field name>.

Where FIELD is an optional phrase added for clarity.

All user-oriented dictionary information, including information concerning categories, is printed for the specified field of the specified data set.

(3) DOCUMENT IN|FOR <data set name> FIELDS [ONLY].

Where ONLY is an optional phrase added for clarity.

Dictionary information, excluding categorical information, is printed for all fields.

(4) DOCUMENT IN|FOR <data set name> *.

This is synonymous with (3) above.

(5) DOCUMENT IN|FOR <data set name> [FIELD] <field name> [CATEGORY] <category name>.

Where FIELD and CATEGORY are optional phrases added for clarity.

The dictionary information for the specified data

set, field and category is printed.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENTS: (1) Dictionary documentation generated via the DOCUMENT command will include the following, if defined:

- (a) data set description;
- (b) field name;
- (c) field abbreviation;
- (d) field description;
- (e) data required status;
- (f) category name;
- (g) category abbreviation;
- (h) category value;
- (i) category description; and
- (j) category required status.

(2) The dictionary documentation generated via the DOCUMENT command is the same as that generated by the stand-alone MICRO.DOC program when DESC is specified. Both require 77 printing columns.

(3) All dictionary information printed as a result of DOCUMENT command is included in the complete information printed as a result of the FULLDOC command. More limited information can be obtained via the TECHDOC, LIST and DESCRIBE commands.

EXAMPLES: DOCUMENT STUDENTS. [B-1]

 DOC IN FIN-AID SOF UNIV. [B-1]

 DOC IN STUDENTS *. [B-1]

 DOC IN FIN-AID TYPE. [B-1]

ERROR MESSAGES: 'DESCRIPTION NOT IN DICTIONARY.'
Indicates that MICRO expected a field description but could not find it.

 'DESCRIPTION NOT IN FILE.'
Indicates that MICRO expected a data set description but could not find it.

'*** MICRC ERROR - BAD POINTER.'

Indicates that an erroneous description pointer was used; probably the dictionary is in error.

'THAT CATEGORY HAS NO DESCRIPTION.'

Indicates that no description is defined for the specified category.

'THAT FIELD HAS NO DESCRIPTION.'

Indicates that no description is defined for the specified field.

'THERE ARE CURRENTLY NO DATA SETS AVAILABLE.'

Indicates that the user has no data sets.

'THERE IS NO DESCRIPTION OF THAT DATA SET.'

Indicates that no description is defined for the specified data set.

END

COMMAND DESCRIPTION

PURPOSE: To permanently terminate the current MICRO session and to return to MTS command mode.

COMMAND SYNONYM: STOP.

PROTOTYPE: END.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

The result set and all other temporary data sets are released.

COMMENTS: (1) The time and date that the MICRO session is terminated are printed.

(2) MICRO cannot be reentered via a \$RESTART command following the END command.

EXAMPLE: STOP. [A-3], [A-6],[B-20]

180
END Command

MICRO Version 5.0
October 1977

ENTERDATA

COMMAND DESCRIPTION

PURPOSE: To add records to a data set.

PROTOTYPE: ENTERDATA [INTC|IN] <data set name>
[PERMANENTLY|PERM] [FROM <file name 1>
WITH ERRORS CN <file name 2>].

Where INTC and IN are optional words which may be added for clarity.

Where PERMANENTLY and PERM are optional words which indicate that the permanent disk copy of the data set is to have the new records added to it. If PERMANENTLY or PERM is not specified, then the new records, combined with the records in the data set, will appear in the temporary result set.

Where the use of "FROM <file name 1> WITH ERRORS CN <file name 2>" is an optional phrase that indicates that the data will be in FREEFORM format, that the raw data will be read from <file name 1>, and that any erroneous raw data records will be printed on <file name 2>. If this phrase is omitted, the user will be interactively prompted for the new data.

APPLICABLE COMMAND MODIFIERS: None

IMPLICIT EFFECT ON THE RESULT SET:

If PERMANENTLY or PERM is specified then there is no change to the result set. However, if neither is specified then a new result set is created and any previous result set is released.

DESTINATION - PERMANENT SET OR RESULT SET:

In addition to the previous description of the effect of the use or omission of the PERMANENT|PERM option, there are other effects:

- (1) If PERMANENTLY or PERM is specified and if FROM <file name 1> WITH ERRORS ON <file name 2> is not

specified, the user will be prompted for the number of records to be queued before they are added to the permanent set. This allows the user to choose the level of protection against losing new records if MTS crashes while ENTERing his data.

- (2) If PERMANENTLY or PERM is omitted, the new records, together with the original records, will appear in the RESULT set only after the user has returned from the data entry mode.
- (3) Data cannot be added PERMANENTLY to a temporary data set.

INTERACTIVE PROMPTING MODE:

If FROM <file name 1> WITH ERRORS ON <file name 2> is not specified, certain rules apply:

- (1) Upon entering the interactive prompting data entry mode the user will be asked if he wants abbreviated prompting. A reply of YES will result in the user being prompted for each field with the abbreviation for the field name. Any other reply results in the use of the full field name.
- (2) Periods (.) should not appear at the end of each line typed (as in MICRO command mode) once prompting for data begins.
- (3) Input for more than one field may be entered on one line if the user wants to anticipate the prompting for the next field(s). Prompting will be suppressed for those fields for which data has already been provided in this manner. If more values are provided on a line of input than there are remaining fields, a warning message will be printed and the extra data will be ignored. The input for each field must be separated or delimited by one or more blanks. If there are one or more embedded blanks in the data for a given field then the data for that field must be surrounded by primes (''). If a prime is to be included in the string, two primes must be used to represent a single prime. The default value

cannot be indicated for a field when data for more than one field is entered on a line.

- (4) Whenever an error message requests a new value for a field, the user may obtain documentation for that field and any of its categories by typing a question mark (?) in column 1.
- (5) If the user receives an error message requesting a new value for a field, only the one replacement value should be entered. The program will remember any additional values previously entered in anticipation.
- (6) After data values for all fields of a record have been entered, the user will be prompted for corrections to the record. The user may change the value assigned to any of the fields by entering:

<field name> <verb> <value or category name>

Where <verb> may be an equal sign (=), comma (,), semi-colon (;), or colon (:). Only one field may be corrected per line. If no more corrections are required then a carriage return should be typed to begin data entry on the next record. If the user has completed his data entry activities, he should type DCNE to return to MICRO. If PERMANENTLY was specified in the ENTER command, the user may also type SAVE to force the addition of records to the permanent set, after which the user will be prompted for data for the next record. The counting process for saving will then be reset.

- (7) "No reply" can be indicated by pushing the carriage return without typing any characters. Except as otherwise indicated, default values are obtained by giving no reply. When a reply is required, the program will try up to five times to get a reply before returning to MTS. If the user wishes to try to reply again, \$RESTART should be issued.
- (8) All fields of a record which have yet to receive data and for which data is not required (as defined during dictionary definition) may be

assigned their default value by transmitting an end-of-file indicator (via Control "C" on the terminal) immediately after prompting for a field occurs or by including a \$ENDFILE (beginning in column 1) in the input stream. The user will still be prompted for values for all of the remaining required fields. This indicator applies only to the current record and must be repeated for each record for which this option is desired. This indicator is identical in function to the /D in batch free-format mode.

- (9) A /S (for same) may be used as the value for external data fields when the user wants to substitute the character string (or line number) from the previous record's input as data for the same external data field. The effect is that the line number or notepointer associated with that previous character string is used as the external data field's value.

BATCH FREE-FORMAT MODE:

If "FROM <file name 1> WITH ERRORS ON <file name 2>" is specified, the rules of batch free-format mode apply:

- (1) Instead of prompting for data, data is simply read from <file name 1> in the same field order as in interactive prompting mode. <file name 1> must be a line file.
- (2) More than one value may be placed on a single line if the values are separated by blanks. Character string values with embedded blanks must be enclosed in primes ('). Single lines are processed as they are read unless a minus sign is placed in the last column transmitted before the carriage return, in which case the field is assumed to be continued in column 1 of the next input line.
- (3) Input for a record in the file may be up to 32,767 characters long in as many lines as the user desires.
- (4) An end-of-record indicator denoting the end of input for each record is required after the last

value. /E is the end-of-record indicator and must be the last characters (except for trailing blanks) in the last line for each record. Input for the first field of each record must begin on a new line. There must be sufficient data prior to the /E for all fields. If too many or too few values are provided before the end-of-record indicator, the record will be rejected and an error message will be printed.

- (5) Unlike the prompting mode, there is no way to give "no reply" for one particular field in order to give the field its default value.
- (6) All fields of a record which have yet to receive data and for which data is not required (as defined during dictionary definition) may be assigned their default value by placing a /D in the input string. The user must still provide values for all remaining required fields in that record. The /D is identical in function to the \$ENDFILE in interactive prompting mode.
- (7) A /S may be used as the value for an external data field when the user wants to refer to the same line in the external data file entered for the same external data field in the previous record. The effect is that the line number or notepointer associated with that previous character string is used as the external data field's value.
- (8) When the data provided for a field is interpreted to be incorrect, the program will identify the error (as in interactive prompting mode). Then the line number containing the error, the line number of the first input line for the aborted record and all of the input lines (with their line numbers) for the aborted record will be printed on the terminal (or elsewhere if SERCOM has been reassigned). The input lines for the aborted record are also copied into <file name 2> at the same line numbers from which they were read. Thus, when data entry is complete, <file name 2> is ready to be edited and used as proper input data in another ENTERDATA command.
- (9) In batch free-format mode, data is automatically

saved when enough records have been collected to fill 32,767 bytes of storage and/or when all data has been read from <file name 1>. After the final save, the number of records read, rejected and added to the data set will be printed in addition to the new total number of records in the data set.

ATTENTION INTERRUPTIONS:

The user may interrupt at any time during the ENTERDATA activities. If the user is in interactive prompting mode and data entry has begun, the user will be asked, "Do you want to add more data?" A reply of YES will start a new record. A reply of NO will save all completed records not already saved and return to MICRO for the next command. In either case, the current record is aborted as soon as the interrupt is received.

During certain critical operations, attention interrupts are ignored.

DATA VALUES FOR SCALED FIELDS:

Data values entered for scaled fields should be entered in the way the user views the data, not the way MICRO stores it as an integer. For example, if AMCUNT was an unsigned integer field scaled by division by 100 (two decimal places as in dollars and cents), then 100 and 100.0 and 100.00 would all be viewed as 100.00, with the internal representation being 10000. If the user wants to indicate three dollars and forty-six cents, then 3.46 should be entered, not 346.

SCRAMBLING:

A scramble key will be requested if the data set to which data is being added needs to be unscrambled. The same key will be used to re-scramble the data set, including the new data. If there is no data in the data set but the dictionary for that set indicates that the data should be scrambled, a scramble key will be requested.

The scramble key is not permanently stored anywhere by MICRO. The user is responsible for

remembering the scramble key.

STAND-ALONE PROGRAMS FOR ENTERING DATA:

It is more economical to enter data through a program that performs data entry only, without any of MICRO's retrieval capabilities. The result is the same and the user interactions are very similar. The only shortcomings are that no notation of data entry is made through the command trace facility and that the user cannot set the CPU timer. [See Comments (7) and (8) below.]

To add data independently in interactive-prompting mode, the user should issue the following MTS command:

```
$RUN IIR:MICRO.ADD
```

To add data independently in batch free-format mode, the user should issue this MTS command:

```
$RUN IIR:MICRO.ADD SCARDS=inputfile  
SPUNCH=errorfile PAR=FREEFORM
```

Where the input file and error file correspond to <file name 1> and <file name 2> in the MICRO ENTERDATA command.

In stand-alone mode, initial user replies are read from the MTS pseudo-device GUSER. MICRO.ADD will prompt (on SFCOM) for the following information:

- (1) MICRO data set name: This may be either the name of an existing data set (indicated by the absence of a colon (:)) in the fifth column of the reply) or the name of an existing MICRO dictionary file (indicated by a four-character signon-ID followed by a colon and by the dictionary file name). If no reply is given the program will terminate.
- (2) Signon-ID of another directory or "CANCEL": Requested only if the data set name cannot be found in the directory that was searched or if that directory does not exist or is not accessible. A four-character signon-ID is all

that is required. If "CANCEL" is the reply, the user will be asked for a new data set name and the search for it will resume with the user's directory. A reply is required.

- (3) Data file name: Requested only if a dictionary file name was given instead of a MICRO data set name. The file may (a) not exist, (b) exist but be empty or (c) exist and contain data. If it does not exist, it must not begin with a signon-ID other than the CCID currently being used. If it exists then it must be a sequential file permitted RW (read-write access). If it contains data then the data must be compatible with the dictionary. A reply is required.

- (4) Scramble key: Requested only if the dictionary indicates that the data set is or should be scrambled. The same key will be used by ENTERDATA to unscramble the data set and to scramble the data set with the additional records. The scramble key must be the one by which the data set was originally scrambled. The user must remember the key because it is not permanently stored anywhere by MICRO. Garbled data is the only indication that the data set has been unscrambled using a different key than when it was scrambled. A reply is required.

COMMENTS:

- (1) At any time in data entry mode, the user may go to MTS by typing MTS beginning in column 1. In batch free-format mode, MTS may be in the input file. The user may return to data entry mode by typing \$RESTART.
- (2) At any time in data entry mode, the user may execute an MTS command. Type the command beginning with a dollar sign (\$) in column 1. In batch free-format mode, the MTS command may be in the input file. Returning to data entry mode (and to MICRO) is dependent upon which MTS command has been executed.
- (3) When the user is faced with replying with YES or NO, any reply beginning with a Y will be interpreted as affirmative and all other replies will be interpreted as negative.

- (4) Long input may be continued on additional lines by placing a minus sign or dash (-) in the last column transmitted before the carriage return. The minus sign will not be included in the input string.
- (5) A category may be specified as the value of a field by entering the category name, its abbreviation or the value associated with that category.
- (6) Data typed after the ENTER command does not appear in the file assigned to the command trace facility, if the facility is enabled.
- (7) In the stand-alone program, MICRO.ADD, a timer interrupt will occur after five CPU seconds have been used. When this happens, the following will be printed:

```
<TIME OUT.....  
<USE "$RESTART" TO CCNTINUE IF YOU ARE MAKING  
PFCGRESS
```

The amount of money spent since signing on will be printed and you will be returned to MTS command mode. If you restart, the timer will be reset to stop again after five more CPU seconds.

- (8) The CPU time limit for the ENTERDATA command from MICRO (or any other MICROC command) defaults to 10 CPU seconds. This may be changed via the SET command.
- (9) External data is never written into the file until the record is completed (and corrected), so that no garbage lines are ever written into an external data file.
- (10) When determining which default value to use for a field that has categories, ENTERDATA does not check to see if more than one category has been designated as the default field value (category type=1). The first type 1 category will be used.
- (11) Values on the same line must not be separated by more than 255 blanks.

- (12) If an external data file does not exist, the program will create it if possible and if the user desires it. Files created in this manner are always line files.
- (13) Caution should be exercised when using the ENTERDATA command in an input stream assigned via the SET INPUT command. Because the ENTERDATA command reads data on SCARDS, field values for updating should be placed in the specified file or device. The ENTERDATA command reassigns SCARDS to *MSOURCE* at their completion.

EXAMPLE: ENTERDATA INTO FIN-AID PERMANENTLY. [A-1]

ERROR MESSAGES: Most of the following error messages for this command are preceded by a phrase which indicates what item or value is incorrect. This phrase begins with the word "concerning" and then indicates the item or value. If the value for a field is incorrect then the name of the field will be identified also. The next lines tell what is wrong and request the user to supply a replacement or make some other response.

'DATA FILE DOES NOT EXIST.'
The data file does not exist but can be created. (See Section 5.4.10.) An attempt is made to create the file if the reply begins with the letter "Y". Other replies cause a request for another data file.

'DATA FILE MUST BE A SEQUENTIAL FILE.'
The file provided as a MICRO data file is a line file and is probably the incorrect file since all MICRO data files must be sequential files. (See Section 5.4.10.)

'DATA FILE MUST EXIST OR BE LOCATED ON THIS SIGNON-ID.'
The data file does not exist and is preceded by a signon-ID other than the current signon-ID. Files may only be created on the current signon-ID. Hence, this file cannot be created. (See Section 5.4.10.)

'DATA FILE NOT COMPATIBLE WITH THE DICTIONARY.'

Records in the data file are shorter than the record length implied in the dictionary. Probably the wrong dictionary is associated with the data file. This condition must be changed.

'ENTER A NEW DATA FILE NAME OR "CANCEL".'

Requested when the data file indicated cannot be used. If the reply is "cancel" the user is asked for a new data set name.

'ENTER A NEW DICTIONARY NAME OR "CANCEL".'

Requested when the indicated dictionary cannot be used. If the reply is "cancel" the user is asked for a new data set name.

'ENTER A NEW TEXT FILE NAME OR "CANCEL".'

Requested when the external data file indicated cannot be used. If the reply is "cancel" data entry is terminated by returning to MICRO (ENTERDATA command) or terminating execution (ILIR:MICRO.ADD).

'ENTER A NEW VALUE OR RETURN TO SKIP THIS FIELD.'

Requested because the value previously provided for this field is invalid. Data is not required for this field. Only one value (if any) should be entered.

'ENTER SCRAMBLE KEY.'

The data set is scrambled and the user must provide (and remember) the character string which is used to scramble and unscramble the data file.

'FIELD MUST BE SET OFF BY "=" OR ":" OR ","
OR ";".'

A correction to a record has been entered incorrectly. This often occurs when the user launches into the next record without indicating there are no corrections.

'HUH?'

This message indicates that an attention interrupt is being processed and that previous activity has stopped.

'IMPROPER FORMAT FOR A DATA FILE.'

The contents of the file provided as a MICRO data file are not a MICRO data set.

'INSUFFICIENT DATA HAS BEEN PROVIDED FOR THIS
RECCRD.'

There is at least one field for which there is no
data.

'INVALID USE OF PRIMES.'

The value for the character field indicated
contains unbalanced primes or the final prime is
not followed by a blank.

'NEW VALUE?'

Data is required for the field indicated. Only
one value should be entered.

'NO SPACE AVAILABLE - DATA ENTRY TERMINATED.'

Either a file could not be created because the
current signon-ID is out of disk space or
(rarely) sufficient virtual memory space is not
available on MTS.

'READ ACCESS NOT ALLOWED.'

The file indicated cannot be read from the
current MTS signon-ID.

'READ/WRITE (RW) ACCESS NOT ALLOWED.'

The file indicated cannot be read from and
written into from the current signon-ID.

'RECORD ABORTED.'

This message indicates that the record in
progress is being discarded due to an attention
interrupt.

'SAVE COMPLETED.'

Records have been added to the original data set.

'TEXT FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT
CREATED.'

The file associated with an external data field
does not exist. A line file is created if the
reply begins with the letter "Y".

'TEXT FILE MUST EXIST OR BE LOCATED ON THIS
SIGNON-ID.'

The external data file for the field indicated
does not exist and cannot be created because the
MTS signon-ID of the file is not the current
signon-ID.

'THIS DIRECTORY DOES NOT CONTAIN THAT DATA SET.'
The directory indicated does not contain an entry for the previously specified data set.

'THIS FILE DOES NOT EXIST.'
The file indicated does not exist.

'THIS IS NOT A FIELD NAME OR ABBREVIATION.'
A field name is incorrect in a correction to a record.

'TOO MUCH DATA HAS BEEN PROVIDED FOR THIS RECORD.'
More data items were provided than there are fields. Such records are rejected.

'VALUE EXCEEDS LENGTH OF FIELD.'
The value indicated will not fit into the field. For character fields too many characters were provided. For numeric fields the number provided is either too large or too small.

'VALUE MUST BE A CATEGORY NAME OR CATEGORY VALUE.'
The field indicated is defined so that all values of the field fall into one of the field's categories. Therefore, values provided for the field must be either a category value or category name or category abbreviation.

'VALUE MUST BE AN INTEGER.'
The value indicated must be an integer number. (Values for fields which are scaled by division may be real numbers. That is, they may contain a decimal point.)

'VALUE MUST BE AN INTEGER OR CATEGORY NAME.'
The value indicated is not a number or category name or category abbreviation.

'VALUE WILL NOT APPEAR AS ENTERED.'
Values for fields scaled by multiplication are divided by the scale factor during processing so that when they are multiplied by the scale factor later, the original value is obtained. However, since integer arithmetic is performed, the remainder of the division, if any, is lost, and the original value cannot be faithfully

recreated. The message indicates that the remainder is non-zero and the original value will not be faithfully recreated.

'WARNING: DATA FILE NOT COMPATIBLE WITH DICTIONARY.'

Records in the data file are longer than the record length implied in the dictionary. Probably the wrong dictionary is associated with the data file.

'WARNING: OVERLY PRECISE VALUES ARE BEING ROUNDED OFF.'

Numerical values with digits on the right side of a decimal point are rounded off if too many digits are provided. For fields which are not scaled by division, values are rounded off to whole numbers. Values for fields scaled by division are rounded to a number of places equal to the power of ten which the scale factor is.

'WARNING - THIS DATA IS LEFT OVER.'

More data items were provided than there are fields. Such records are accepted.

'WARNING: VALUES WITH TOO MANY DIGITS ARE BEING ROUNDED OFF.'

If the field indicated is scaled by division then numbers with more than ten digits which contain decimal points are rounded off before further processing as long as the rounding can be done to the right of the decimal point.

'WRITE ACCESS NOT ALLOWED.'

The file indicated cannot be written into from the current MTS signon-ID.

'YOU MUST MODIFY THIS DICTIONARY TO REFLECT THIS CHANGE.'

This is a reminder to update the dictionary to reflect the use of a different external data file.

SPECIAL DESIGN CONSIDERATIONS:

- (1) If at the time of data set definition a field was declared to require data, the user must supply a value for that field. No default value will be

used. Otherwise, if no data is provided for a field (by pushing only the carriage return or sending an end-of-file), the default value of the field will be the value of a category declared as the default field value. If there is no such category for the field then zero and blank are the default values for numeric and character fields respectively.

- (2) If at the time of data set definition a field was declared to require a category and if the user provides data for a field then the data entered must be a category name, abbreviation, or the value associated with a category.

EXPLAIN

COMMAND DESCRIPTION

PURPOSE: To print an explanation of the specified MICRO command.

PROTOTYPE: EXPLAIN [COMMAND] <command name>.

Where COMMAND is an optional phrase added for clarity and <command name> can be any valid MICRO command, any acceptable abbreviation or any of its synonyms.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENTS: (1) The explanations are extracted directly from the MICRO Reference Manual.

(2) A complete copy of the latest version of the MICRO Reference Manual can be obtained by purchasing a printed copy for \$9.95 from:

Institute of Labor and Industrial Relations
The University of Michigan-Wayne State University
E.C. Fox E-1
Ann Arbor, Michigan 48106

(3) Pages containing revisions to the latest MICRO Reference Manual may be obtained by issuing the MTS command

\$COPY ILIR:MICRO.REVISIONS TO *PRINT*

where *PRINT* should be set for a TN (upper/lower case) printer.

EXAMPLE: EXPLAIN COMMAND EXPLAIN. [B-1]

ERROR MESSAGES: 'EXPLANATION FILE IS NOT UP TO DATE.'
Indicates that no explanation is available for the command specified.

'NO SUCH COMMAND.'

Indicates that an explanation was requested for a command which does not exist.

FIND

COMMAND DESCRIPTION

PURPOSE: To create a result set consisting of those records from a data set that satisfy the specified criteria.

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

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

<clause 1> [<conjunction> <clause 2>...]

and where a <clause> may be any of the following:

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

Where <verb> is any of those listed in the section on EQUIVALENT VERB GROUPS later in this command description and where <conjunction> is either AND, OR or ALSO WHERE, whose meanings are discussed in the section on CONJUNCTIONS which also follows.

Where <value> may be a signed or unsigned integer, category name or abbreviation 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.

If a character field is being compared to a character string, the comparison will begin with the leftmost character of each and proceed to the right. If the <verb> is MATCHES, the comparison will end with the last non-blank character of the character string. If the <verb> is IS, or an equivalent (other than MATCHES), the comparison will end with the last non-blank character of the character field. Consequently, FIND IN STUDENTS WHERE NAME MATCHES HERSH will find all records where

the field called NAME begins with "HERSH", e. g., HERSH, HERSHEY, HERSHFIELD, etc. But FIND IN STUDENTS WHERE NAME IS HERSH will only find those records where the name is exactly "HERSH", not "HERSHEY", etc.

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

This form refers to those records whose field value is between <value 1> and <value 2> but not equal to <value 1> or <value 2>.

- (c) <field name> IS FROM <value 1> TO <value 2>.

This form refers to those records whose field values are either between <value 1> and <value 2> or equal to <value 1> or <value 2>.

- (d) <field name 1> <verb> <field name 2>

In this form, only fields of the same basic type may be compared. If the field types are unsigned or external, the fields are compared logically; that is, they are compared from right to left through the width of the shorter field and then the remaining bytes of the longer field are compared with zeros. If the field types are signed then the process of comparison is altered to take into consideration the presence of the sign in the leftmost bit of each field. If the field types are character and the verb is MATCHES, then the shorter field will be compared to the leftmost characters of the longer field, up to the length of the shorter field.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

COMMENTS: (1) The number of records meeting the specified

criteria is printed. The percentage of selected records found (those in the RESULT set) out of all records searched in <data set name> is also printed.

- (2) 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 STUDENTS WHERE CLASS-LEVEL IS FRESHMAN OR
CLASS-LEVEL IS SOPHOMORE OR CLASS-LEVEL IS
JUNIOR.

Is equivalent to:

FIND IN STUDENTS WHERE CLASS-LEVEL IS FRESHMAN OR
SOPHOMORE OR JUNIOR.

In both cases, one pass is made through the data set searching for all specified values of the specified field.

EXAMPLES:

FIND IN FIN-AID WHERE SOF IS UNIVERSITY OR
FEDERAL-GCV OR STATE GOV. [B-5]

FIND IN FIN-AID WHERE CLASS-LEVEL IS FRESHMAN.
[B-6]

FIND IN FIN-AID WHERE CLASS-LEVEL IS NOT JUNIOR.
[B-6]

FIND IN FIN-AID WHERE TYPE-OF-AID IS TUITION OR
ECOM-BOARD OF WORK-STUDY. [B-8]

FIND IN FIN-AID WHERE TYPE-OF-AID IS TUITION OR
ECOM-BOARD OF WORK-STUDY AND SOURCE-OF-FUNDS IS
UNIVERSITY OR CLASS-LEVEL IS JUNIOR AND TERM IS
FALL OR TERM IS WINTER AND AMOUNT IS FROM 0 TO
150 or AMOUNT > 750 ALSO WHERE YEAR IS 73 OR 74
AND SOURCE-OF-FUNDS IS INDUSTRY. [B-8]

FIND IN FIN-AID WHERE YEAR IS 73 OR 74. [B-8]

FIND IN RECIPIENTS WHERE AMT >= 500.00. [B-15]

FIND IN SMITH WHERE NAME IS 'SMITH MARY'. [B-4]

FIND IN STUDENTS W CLYR IS 80 AND RACE IS BLACK. [B-3]
FIND IN STUDENTS W SEX IS FEMALE. [B-10]
FIND IN STUDENTS W SEX IS MALE. [B-10]
FIND IN STUDENTS WHERE NAME MATCHES SMITH. [B-4]
FIND IN TEMP1 WHERE CLASS-LEVEL IS JUNIOR. [B-8]
FIND IN TEMP1 WHERE SOURCE-OF-FUNDS IS UNIVERSITY. [B-8]
FIND IN TEMP11 WHERE SOURCE-OF-FUNDS IS INDUSTRY. [B-8]
FIND IN TEMP4 WHERE TERM IS FALL. [B-8]
FIND IN TEMP7 WHERE AMOUNT > 750. [B-8]
FIND IN TEMP7 WHERE AMOUNT IS FROM 0 TO 150. [B-8]
FIND IN 4 WHERE SIN IS 234567890. [B-13]

See also the detailed example in the section on conjunctions for this command.

ERROR MESSAGES:

'BETWEEN' CLAUSE SPECIFIES AN EMPTY INTERVAL.'
MICRO assumes that when an empty interval is specified, i.e. YEAR IS BETWEEN 1977 AND 1978, that the user has made an error and that the command is cancelled.

'ERROR - NO CONDITION SPECIFIED!'
Indicates that the command was incomplete in that no criteria was specified when one was required.

'FIELDS ARE NOT COMPATIBLE.'
Indicates that a field to field comparison cannot be performed because the two fields are not comparable.

'INTERVAL FOR < OR > IS LOGICALLY EMPTY.'
MICRO assumes that when an empty interval is

specified, i.e. AGE < 0, that the user has made an error and that the command is cancelled.

'"IS NOT BETWEEN" IS NOT AVAILABLE.'
"IS NOT BETWEEN" is not a defined operation.

EQUIVALENT GROUPS OF FIND VERBS:

The following verbs within each group are equivalent and may be used interchangeably in MICRO. Words within square brackets ([]) are optional and may be omitted.

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] AFTER

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] BEFORE

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] SINCE

IS|ARE LESS THAN OR EQUAL TO
IS|ARE EQUAL TO CP LESS THAN
[IS|ARE] <=
[IS|ARE] =<
IS|ARE NOT GREATER THAN
[IS|ARE] NOT >
[IS|ARE] UNTIL

[DO|DCES] MATCHES
[DO|DOES] MATCH
[DO|DOES] MAT
[DC|DCES] MA

[DO|DOES] NOT MATCHES
[DC|DCES] NOT MATCH
[DO|DOES] NOT MAT
[DC|DCES] NOT MA

CONJUNCTIONS:

There are three conjunctions used in the FIND command: OR, AND and ALSO WHERE. Each is used to connect clauses and has a special purpose that corresponds to its implementation in MICRO. The same meaning can be communicated by a series of sentences consisting of one clause each or by one sentence consisting of many clauses connected by conjunctions. The use of conjunctions in MICRO will first be described by definitions and rules and then by example.

Each clause of a FIND command is interpreted and the corresponding search is executed on a single separate pass through the data set. There is one exception. If the same field is referenced in consecutive clauses separated by ORs either explicitly (as in YEAR = 1974 OR YEAR = 1975 OR YEAR = 1976) or implicitly (as in YEAR = 1974 OR 1975 OR 1976), then only one pass is made in the search for all values of the referenced field.

The OR conjunction is analogous to the union operation in set theory. It implies an inclusive OR, that is, either A or B or both. When searching for records, the use of OR may add more records to the result set because records meeting a variety of conditions may be found. For example, additional months specified in the clause

MONTH = FEBRUARY OR MARCH OR ...

Would allow additional records that meet the search criteria and become part of the result set.

The AND conjunction is analogous to the intersection operation in set theory. Its use between clauses further restricts (or reduces) the records which meet the search criteria from becoming part of the result set. For example, if the clause

MONTH = FEBRUARY AND MARCH

were part of a FIND command, the result set would be empty (contain no records) because the additional restriction imposed by the second month would remove all records from consideration since MONTH cannot be equal to two values at the same time within the same record. A more useful example of clauses joined by AND is

YEAR = 1976 AND MONTH = MARCH

in this case, only those records meeting both criteria would appear in the result set.

The ALSO WHERE conjunction has no true parallel in set theory. It has been added to MICRO to lessen the user's concern with the precedence of operations in multi-clause FIND commands. Its use will become clearer after precedence rules are discussed.

In MICRO, precedence of operations is determined solely by the order in which clauses appear from left to right within the FIND command. Parentheses do not affect precedence and are not allowed. Care must be exercised when mixing ORs, ANDs and/or ALSO WHEREs within the same FIND command. That is because clauses are executed in the order in which they appear and MICRO must create intermediate result sets to store those records meeting the search criteria. The intermediate result sets from subsequent clauses must be operated upon to form a new intermediate result set according to the conjunction

specified. The following rules indicate which intermediate set is being searched:

- (1) OR causes the search to be performed on the intermediate result set generated before the previous AND, if one was encountered; otherwise the search is performed on the first intermediate result set.
- (2) AND causes the search to be performed on the intermediate result set generated after the previous OR, if one was encountered; otherwise the search is performed on the previous intermediate result set.
- (3) ALSO WHERE always returns the search to the specified data set regardless of any previously encountered ORs or ANDs.

More generally, the use of ALSO WHERE is like starting with a clean slate. The intermediate result set from clauses preceding an ALSO WHERE conjunction is combined with the intermediate result set from the clauses following that ALSO WHERE conjunction up to the next ALSO WHERE conjunction or the end of the command.

These rules will be clarified by a detailed example. A multiclause FIND command will be broken down into a series of steps consisting of MICRO commands showing how a multiclause FIND command might be executed. The decomposed steps are prefixed by an asterisk (the synonym for the COMMENT command). The execution of both the entire command and the separate steps is shown in Appendix A. The example multiclause FIND command is not necessarily meaningful but is illustrative of the conjunctions and precedence:

```
FIND IN FIN-AID WHERE TYPE-OF-AID IS TUITION OR
FCCM-FOART OR WORK-STUDY AND SOURCE-OF-FUNDS IS
UNIVERSITY OR CLASS-LEVEL IS JUNIOR AND TERM IS
FALL OR TERM IS WINTER AND AMOUNT IS FROM 0 TO
150 OR AMOUNT > 750 ALSO WHERE YEAR IS 73 OR 74
AND SOURCE-OF-FUNDS IS INDUSTRY.
```

The following series of steps is equivalent. All comments describing the equivalent steps begin

with an asterisk (*) and end with a period. The part of the multiclaue command being executed will appear in a comment within quotation marks ("). The actual equivalent MICRO commands begin in column 1.

```
STEP 1
"FIND IN FIN-AID WHERE TYPE-OF-AID IS TUITICN OR ROOM-BOARD OR
WORK-STUDY"
* *****
FIND IN FIN-AID WHERE TYPE-OF-AID IS TUITICN OR ROOM-BOARD OR
WORK-STUDY.
NAME RESULT TEMP1.
* *****

STEP 2
"AND SOURCE-OF-FUNDS IS UNIVERSITY"
Reference is made to the previous intermediate result set (from
Step (1) because there was no previous OR clause
* *****
FIND IN TEMP1 WHERE SOURCE-OF-FUNDS IS UNIVERSITY.
NAME RESULT TEMP2.
* *****

STEP 3
"OR CLASS-LEVEL IS JUNICR"
Reference is made to TEMP1 because it is the intermediate result
set before the previous AND clause (in Step 2)
* *****
FIND IN TEMP1 WHERE CLASS-LEVEL IS JUNICR.
NAME RESULT TEMP3.
COMBINE TEMP2 AND TEMP3.
NAME RESULT TEMP4.
RELEASE TEMP2.
* *****

STEP 4
"AND TERM IS FALL"
Reference is made to TEMP4 because it is the intermediate result
set generated after the previous OR clause (in Step 3)
* *****
FIND IN TEMP4 WHERE TERM IS FALL.
NAME RESULT TEMP5.
RELEASE TEMP1.
* *****

STEP 5
"OR TERM IS WINTER"
Reference is made to TEMP4 because it is the intermediate result
set generated (in Step 3) before the previous AND clause
* *****
FIND IN TEMP4 WHERE TERM IS WINTER.
```

```
NAME RESULT TEMP6.
RELEASE TEMP4.
COMBINE TEMP5 AND TEMP6.
NAME RESULT TEMP7.
RELEASE TEMP5 AND TEMP6.
* *****
STEP 6
    "AND AMOUNT IS FROM 0 TO 150"
    Reference is made to TEMP7 because it is the intermediate result
    set generated (in Step 5) after the previous OR clause
    * *****
RELEASE TEMP3.
FIND IN TEMP7 WHERE AMOUNT IS FROM 0 TO 150.
NAME RESULT TEMP8.
* *****
STEP 7
    "OR AMOUNT > 750"
    Reference is made to TEMP7 because it is the intermediate result
    set generated (in Step 7) before the previous AND clause
    * *****
RELEASE TEMP4.
FIND IN TEMP7 WHERE AMOUNT > 750.
NAME RESULT TEMP9.
RELEASE TEMP7.
COMBINE TEMP8 AND TEMP9.
NAME RESULT TEMP10.
RELEASE TEMP8.
* *****
STEP 8
    "ALSO WHERE YEAR IS 73 OR 74"
    Reference is made to FIN-AID, the originally specified data set
    * *****
RELEASE TEMP9.
FIND IN FIN-AID WHERE YEAR IS 73 OR 74.
NAME RESULT TEMP11.
* *****
STEP 9
    "AND SOURCE-OF-FUND IS INDUSTRY"
    Reference is made to TEMP11 because it is the previous
    intermediate result set (there was no previous OR because of the
    ALSO WHERE)
    * *****
FIND IN TEMP11 WHERE SOURCE-OF-FUNDS IS INDUSTRY.
NAME RESULT TEMP12.
RELEASE TEMP11.
* *****
STEP 10
    "." (PERIOD INDICATING END OF COMMAND)
```

* *****.
COMBINE TEMP10 AND TEMP12.
RELEASE TEMP10 AND TEMP12.
* Result set contains records meeting specified criteria.

NOTEWORTHY USAGE STRATEGIES:

- (1) If two character fields of varying widths are compared then they will be compared from right to left which is of very limited value if the character string data is left-justified.
- (2) Greater efficiency can be obtained on a multiclaue FIND command consisting of a series of AND clauses if the most restrictive clause is first in the series to reduce the size of the intermediate result set for subsequent searches.
- (3) The RESTRICT command can also be used to "find" certain records in a data set.

SPECIAL DESIGN CONSIDERATIONS:

- (1) A search cannot be performed on the character data stored indirectly in external data type fields. Only the line number or notepointer (pointing to the character string) may be searched.
- (2) It is very useful for dates to be stored in YYMMDD format so that the most significant digits are to the left and the least significant are on the right. Thus December 25, 1976 would be 761225 and is logically greater than all dates that precede it.

210
FINE Command

MICRO Version 5.0
October 1977

FULLDOC

COMMAND DESCRIPTION

PURPOSE: To completely document both user-oriented and technical dictionary information for one or all fields of a specified data set.

COMMAND SYNONYM: FDCC

PROTOTYPES: (1) FULLDOC [DATA SET] <data set name>.

Where DATA SET is an optional phrase added for clarity.

All dictionary information for all of the fields in <data set name> is printed.

(2) FULLDOC IN|FOR <data set name> [FIELD] <field name>.

Where FIELD is an optional phrase added for clarity.

All dictionary information, including any categorical information, is printed for <field name> in <data set name>.

(3) FULLDOC IN|FOR <data set name> FIELDS [ONLY].

Where ONLY is an optional phrase added for clarity.

Dictionary information, excluding categorical information, is printed for all fields.

(4) FULLDOC IN|FOR <data set name> *.

This is synonymous with (3) above.

(5) FULLDOC IN|FOR <data set name> [FIELD] <field name> [CATEGORY] <category name>.

Where FIELD and CATEGORY are optional phrases added for clarity.

The dictionary information for <category name> of

MICRO Information Management System (Version 5.0) Reference Manual

'DESCRIPTION NOT IN FILE.'

Indicates that MICRO expected a description but could not find it.

'*** MICRO ERROR - BAD POINTER.'

Indicates that an erroneous description pointer was used; probably the dictionary is in error.

'THAT CATEGORY HAS NO DESCRIPTION.'

Indicates that no description is defined for the specified category.

'THAT FIELD HAS NO DESCRIPTION.'

Indicates that no description is defined for the specified field.

'THERE ARE CURRENTLY NO DATA SETS AVAILABLE.'

Indicates that the user has no data sets.

'THERE IS NO DESCRIPTION OF THAT DATA SET.'

Indicates that no description is defined for the specified data set.

GET

COMMAND DESCRIPTION

PURPOSE: To acquire information from a directory on another signon-ID in order to reference the MICRO data sets listed therein.

PROTOTYPE: GET [DIRECTORY FROM] <ccid>.

Where DIRECTORY FROM is an optional phrase added for clarity and <ccid> is a four-character MTS signon-ID.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

WHAT HAPPENS: The data sets listed in the directory on <ccid> are appended to the core copy (but not the permanent disk copy) of the signed-on user's directory.

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

(2) Only the first of multiple occurrences of the identically named data sets may be accessed. There is no warning if there are multiple occurrences of the same data set name.

(3) The user must have MTS read access to use the DIRECTORY file on <ccid>. In addition, the user must have MTS read access to the desired data and dictionary files in order to reference the data sets which appear in the list of available sets. Similarly, access to a particular data set may be restricted because the data set and/or dictionary associated with that set are not permitted, even though the data set name appears in the list of available data sets.

EXAMPLE: GET K59L. [B-1]

ERROR MESSAGE: 'THE DATA SETS ON THIS CCID ARE ALREADY AVAILABLE.'

Indicates that the user requested MICRO to GET the directory on the ccid that he is currently signed on.

JOIN

COMMAND DESCRIPTION

PURPOSE: To create a result set containing the fields from the two data sets.

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

The key fields of each record in the first data set are compared with the specified fields of each record in the second data set. When the two values are the same (i.e., they match), an expanded record is created and added to the result set. Each expanded record consists of the key fields followed by the non-key fields in <data set name 1> followed by the non-key fields in <data set name 2>. If a key value(s) from either data set cannot be matched with a key value(s) from the other data set, filler values are inserted in the expanded record in the place of missing data. Hence, the result will contain a record which represents each key value from both data sets. See Comment (2) for a discussion of filler values.

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

The action is the same as in (1) above. However, the field name(s) which make up the key for <data set name 2> are assumed to have the same name(s) as the field name(s) which constitute the key of <data set name 1>.

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

This prototype is similar to (2) except that the field name(s) which make up the key are explicitly stated for <data set name 2> and implied for <data set name 1>.

(4) JOIN <data set name 1> WITH <data set name 2>.

In this prototype, no key is specified and each record of <data set 1> is joined with each record of <data set 2>. The number of records in the result set equals the product of the number of records in each set. This results in the cartesian product of the two sets. This form of the command is identical to prototype (4) of the RESTRICTANDJOIN command.

APPLICABLE COMMAND MODIFIERS:

@GF=<integer>

EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

- COMMENTS:
- (1) The key fields must be ordered in the command so that fields of the same type and length are being compared.
 - (2) In the event that a key field value(s) does not find a match in the other data set, then filler value(s) for each undefined field in the expanded record are selected according to these rules:
 - (a) If a default category is specified for a categorical field then the default category value is used; and
 - (b) If there is no default category, blanks will be used for character fields and zeros will be used for all other fields, including external fields.

The number of records in the result set depends upon the number of matches on the key between the data sets and on whether the key uniquely identifies each record of the data sets. The number of records in the result set will never be less than that of the larger set. It is easily possible to create an extremely large set as the result of this command. Unless GROWTH FACTOR is increased via the SET command or the GF command modifier, MICRO will not always successfully create a set more than two times larger than the storage required by the larger of the two data

sets. The permissible values for GROWTH FACTOR are 1 to 9.

The JOIN command does not treat a crosstabulated data set as crosstabulated. Hence, the RESULT of this command will never be a crosstabulated set.

If there are non-key fields in both data sets with the same name, the result set will contain more than one field with the same name. The second and additional fields of the same name cannot be explicitly referenced without first removing or changing the name of the first occurrence.

The result set from a JOIN command does not contain the descriptions for the fields and categories appended from the second data set. Therefore, no description will appear in dictionary documentation.

EXAMPLES: JOIN STUDENTS BY SIN WITH FIN-AID. [B-10]

JCIN WOMEN WITH MEN. [B-10]

ERROR MESSAGES: 'INCOMPATIBLE DATA SETS - FIELD LENGTHS.'
Indicates that the two specified sets have at least one key field of differing length.

'INCOMPATIBLE DATA SETS - FIELD NAMES.'
Indicates that an implicitly specified field name does not exist.

'NO DATA SET #1 SPECIFIED.'

'NO DATA SET #2 SPECIFIED.'

'RESULT SIZE LIMIT EXCEEDED.'
Indicates that the result of the command grew more than was allowed; see the growth factor command modifier discussion.

'TOO MANY DATA SETS SPECIFIED.'

'UNABLE TO PROCESS EMPTY SET.'
Indicates that a prototype (4) was requested and at least one of the sets was empty.

'UNEQUAL FIELD LISTS.'

Indicates that the number of fields specified for the two sets differs.

SPECIAL DESIGN CONSIDERATION:

Care should be taken to insure that the identical field type and length are given to fields appearing in data sets which the user might wish to JOIN. For example, if UNIT-CODE is defined as a fullword integer in one data set and as a one-byte integer in another data set, the JOIN command cannot be executed using UNIT-CODE as one of the specified fields used to match the two sets.

KEY

COMMAND DESCRIPTION

PURPOSE: To add a unique integer field (usually named KEY) to the specified data set.

COMMAND SYNONYM: NUMBER

PROTOTYPE: KEY <data set name> [AT END] [[STARTING WITH] <starting integer>].

Where AT END indicates that the new field is to be added as the last field of the data set. If AT END is omitted then the new field will be added as the first field of the data set.

Where STARTING WITH is an optional phrase added solely for clarity.

Where <starting integer> is the positive integer with which the key is to begin. If <starting integer> is omitted, the record will be numbered from 1 to n by 1, where n is the number of records in the data set.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result, if one existed, is released.

COMMENTS: (1) The new field named KEY will be a four byte (fullword) unsigned integer field. Therefore, the <starting integer>, if specified, should be chosen so that the key of the last record will not be greater than 2,147,483,647. The abbreviation of the new field will be K.

(2) If a field named KEY already exists in the specified data set, the user will be prompted for another field name.

EXAMPLES: KEY RECIPIENTS. [B-14]

KEY S STARTING WITH 1004. [B-7]

KEY S. [B-7]

KEY STUDENTS. [B-13],[B-16]

NOTEWORTHY USAGE STRATEGIES:

KEYing a data set before SELECTing fields from it will guarantee that no records will be lost when the KEY field is one of those fields selected. If the KEY field is selected as the first field then the ordering of records will be unchanged even though fields have been removed.

LIST

COMMAND DESCRIPTION

PURPOSE: To list any of the following:

To list the names of available and/or active data sets, the field names and abbreviations of a data set, the category names and abbreviations of a field of a data set, or the names of all of the MICRO commands.

PROTOTYPES: (1) LIST [ALL] [DATA] SET[S] [NAMES].

Where ALL, DATA and NAMES are optional phrases added for clarity.

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 active.
- (b) DISK* - permanent data set on disk, active copy is loaded.
- (c) TAPE - permanent data set on tape, not active.
- (d) TAPE* - permanent data set on tape, active copy is loaded.
- (e) TEMP* - temporary data set (always active).

Active implies that the dictionary is loaded into memory and the data file is open with up to five pages associated with it.

(2) LIST [ALL] ACTIVE [DATA] SET[S] [NAMES].

Where ALL, DATA and NAMES are optional phrases added for clarity.

The names of all data sets that are currently loaded will be printed.

(3) LIST ASSETS.

The names of all data sets that are currently active are printed. (Same as (2) above.)

(4) LIST IN|FOR <data set name > [ALL] FIELDS [NAMES].

Where ALL and NAMES are optional phrases added for clarity.

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

(5) LIST IN|FOR <data set name> [ALL] CATEGORIES OF <field name>.

Where ALL is an optional phrase added for clarity.

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

(6) LIST [ALL] COMMAND[S] [NAMES].

A complete list of command names and their synonyms are printed.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

- COMMENTS:
- (1) The LIST command should not be confused with the PRINT command. The LIST command lists information about data sets, while the PRINT command prints information contained in data sets.
 - (2) To obtain an explanation of a MICRO command, use the EXPLAIN command.
 - (3) To obtain information about data sets, use the DOCUMENT, FULLDCC, TECHDCC and DESCRIBE commands.

EXAMPLES: LIST ACTIVE SETS. [B-5]

LIST ALL DATA SETS.	[B-1]
LIST COMMANDS.	[B-1]
LIST SETS.	[B-5], [B-10]

MESSAGE

COMMAND DESCRIPTION

- PURPOSE:** To print a message as output from MICRO.
- PROTOTYPE:** MESSAGE '<character string>'.
Where <character string> is any character string up to 500 characters in length.
- APPLICABLE COMMAND MODIFIERS:** None.
- IMPLICIT EFFECT ON THE RESULT SET:** None.
- COMMENTS:**
- (1) The MESSAGE command differs from the COMMENT command in that the character string in MESSAGE is "echoed back" as output from MICRO as well as appearing as input to MICRO.
 - (2) Each single prime (') in <character string> must be represented by two consecutive primes.
 - (3) The first character is interpreted as a carriage control character.
 - (4) Like all other MICRO commands, the MESSAGE command must be terminated with a period.
- EXAMPLE:** MESSAGE 'DON''T FORGET THAT THE DATA IN THIS OUTPUT WILL BE OBSOLETE BY THE END OF NEXT MONTH'.
- NOTEWORTHY USAGE STRATEGIES:**
- (1) This command can be used to print remarks from a macro, input file or runfile.
 - (2) This command can be used to insert carriage control characters and titles in the output stream.

MTS

COMMAND DESCRIPTION

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

COMMAND SYNONYM: SYSTEM

PROTOTYPE: MTS.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENT: MICRO can be re-entered via a \$RESTART command unless an MTS command is executed which will result in MICRC being unloaded (\$RUN, \$LOAD, \$UNLOAD, \$EEUG or \$SIGNOFF).

EXAMPLE: MTS. [MTS]

230
MTS Command

MICRO Version 5.0
October 1977

NAME

COMMAND DESCRIPTION

PURPOSE: To temporarily assign a different name to any data set, field, or category.

COMMAND SYNONYM: RENAME

PROTOTYPES: (1) NAME <old data set name> <new data set name>.

Where <new data set name> may be up to 11 characters in length as long as it is not the name of any other data set, either temporary or permanent.

If <old data set name> is a temporary set, then there will no longer be a set called <old data set name> after it has been NAMED.

Only a temporary set may be NAMED RESULT. If <old data set name> is a permanent set then the name will be changed only in the core copy of the directory. That is, the name in the permanent (disk) directory remains unchanged.

(2) NAME IN <data set name> <old field name> [ABBR] TO <new name>.

Where <new name> may be up to 16 characters in length unless ABBR is specified in which case the length may be up to four characters in length.

If ABBR is omitted then only the field name is changed to <new name>.

If ABBR is specified then only the field abbreviation is changed to <new name>.

<new name> cannot be the name of an existing field or category name or abbreviation in <data set name>.

(3) NAME IN <data set name> <field name>
<old category name> [ABBR] TO <new name>.

Where <new name> may be up to 11 characters in

<new name> cannot be an existing field or category name or abbreviation for <field name> in the specified data set.

EXAMPLES:

NAME IT BIGJCIN.	[B-10]
NAME RESULT TEMP1.	[B-8]
N IT MEN.	[B-10]
RENAME IN IT BDAY TO BIRTHDATE.	[B-17]
RENAME IN IT BIRTHDATE TO NEWFIELD.	[B-17]
RENAME IN IT NEWFIELD ABBR TO NEW.	[B-17]

'ERROR - NEW SET NAME EXCEEDS 11 CHARS.'

Indicates that the user attempted to give a data set a new name which exceeded eleven characters.

- (1) The primary use of this command is to change the name of the result set to a new name. This has the effect of preserving the set for the duration of the MICROC session (unless the set is RELEASED, PURGED or DESTROYED) since the set named RESULT is the only set which may be released implicitly

during commands which create new result sets.

- (2) All field abbreviations that contain non-alphabetic characters should be RENAMED prior to entering WRITE FOR ANALYSIS for MIDAS.

234
NAME Command

MICRO Version 5.0
October 1977

PLOT

COMMAND DESCRIPTION

PURPOSE: To generate a scatter plot with the points connected for up to six fields plotted against one field. Printing can be on a computer graphics terminal or a CalComp plotter.

PROTOTYPE: `PLOT IN <data set name> <field name Y1> [, <field name Y2>, ... , <field name Y6>] AGAINST [MONTHLY|QUARTERLY|YEARLY] <field name X> [ON <file name>[WITH LENGTH [OF][=]n ]].`

Where <field name Y1> through <field name Y6> are plotted on the Y-axis against <field name X> on the X-axis.

Where MONTHLY, QUARTERLY and YEARLY are optional words which cause the values of <field name X> to be specially interpreted as follows:

MONTHLY assumes that values of the field are in the format of YYMM where MM is from 01 to 12. And YY is a two digit year code.

QUARTERLY assumes that the values of the field are in the format YYQQ where QQ is from 01 to 04.

YEARLY assumes that the values of the field are in the format of YYYY (such as 1976) and are printed in that format.

Where the absence of ON <file name> will cause the plot to be displayed at the terminal, if the terminal has graphics capabilities.

Where ON <file name> will cause the plot to be printed on the specified file for later queuing for the CalComp plotter via the *CCQUEUE program. The LENGTH parameter allows variation in the size of the plot. n is an integer value representing the size of the longer axis in inches.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

- COMMENTS:
- (1) The integrated graphics (IG) routines are used to generate the graphs.
 - (2) See the *CCQUEUE write-up in Volume 2 of the MTS Manual for a description on the generation of a CalComp plot from the output file.
 - (3) Each field is labeled on the graph with its MICRO field name.
 - (4) Scaling factors associated with the specified fields are ignored in the generation of the graphs.
 - (5) <file name>, when specified, should be an empty or non-existent permanent file.

EXAMPLE: PLCT IN RESULT AVE.AMOUNT, MIN.AMOUNT, MAX.AMOUNT
 AGAINST YEARLY YEAR CN PLCTOUT. [B-18]

NOTEWORTHY USAGE STRATEGY:

All values are plotted on the same scale between the smallest and largest values in any of the one to six fields. More readable plots will be generated when the fields of the Y axis should have a similar range of values.

PRINT

COMMAND DESCRIPTION

PURPOSE: To print on a terminal, file or device either the number of records in a data set or some or all of the information contained in a data set.

PROTOTYPES: (1) PPRINT IN <data set name> COUNT.

The number of records in the data set is printed. If <data set name> is an XTAB set then the number of records represented is also printed.

(2) PPRINT [ON <file name>] [ENTIRE] <data set name>.

where <file name> may be any temporary or permanent MTS file or device.

where ENTIRE is an optional phrase added for clarity.

All fields of the data set are printed for each record.

(3) PPRINT.

All fields of the RESULT set are printed for each record. This prototype is synonymous with PRINT RESULT, a special case of prototype (2).

(4) PPRINT [ON <file name>] in <data set name> <field name 1> [AND <field name 2> ...].

where <file name> is any MTS file or device.

The contents of the specified fields are printed record by record for the data set.

IMPLICIT EFFECT ON THE RESULT SET: None.

APPLICABLE COMMAND MODIFIERS:

@ABBREVIATION
@COUNT=<integer>
@DESCRIPTION
@FIRST=<integer>

@FORWARD
@HEXADECIMAL
@HCRIZONTAL
@INCREMENT=<integer>
@INDIRECTION
@PAGE[=<integer>]
@FC
@FD[=<integer>]
@REVERSE
@STRATIFICATION=<integer>
@VALUE
@VERTICAL
@WINDOW[=<integer>]
@X

APPLICABLE FIELD MODIFIERS:

@ABBREVIATION
@DATE
@DESCRIPTION
@HEXADECIMAL
@INDIRECTION
@SSN
@VALUE
@WINDOW=<integer>
@X

APPLICABLE MICRO FACILITIES SET VIA THE SET COMMAND:

GELOCK (Get Block size)
HCC (Header Carriage Control)
HIRCC (Horizontal Print Format
Interrecord Carriage Control)

HCRIZONTAL PRINT
LPP (Lines Per Page)
CRI (Output Record Length)
OUTPUT
PC (Page Control)
PCC (Page Carriage Control)
EDATE (Page Title)
PRINTDIRECTION (Forward or Reverse)
ICC (Title Carriage Control)
TITLE
VALUE
VERTICAL PRINT
VIRCC (Vertical Print Format
Interrecord Carriage Control)

WINDOW (Width in which character strings are
 printed)

In all cases a field modifier will override the
effect of a command modifier which will override
the effect of a facility.

INTERRELATIONSHIPS BETWEEN FACILITIES, COMMAND MODIFIERS AND FIELD MODIFIERS:

The table on the next page indicates how various
printing activities are controlled by facilities,
command modifiers and field modifiers. Each
printing activity is discussed separately.

PRINTING ACTIVITIES	FACILITY	COMMAND MODIFIER	FIELD MODIFIER
OUTPUT DESTINATION	CRI CUTPUT		
WHICH RECORDS ARE TO BE PRINTED?	PRINT DIRECTION	@ACCUNT @FIRST @FORWARD @INCREMENT @REVERSE	
RECORD FORMAT	HORIZONTAL VERTICAL	@HORIZONTAL @VERTICAL @STRATIFICATION	
FIELD FORMAT	WINDOW PVALUE	@ABBREVIATION @DESCRIPTION @HEXADECIMAL @INDIRECTION @VALUE @WINDOW @X	@ABBREVIATION @DATE @DESCRIPTION @HEXADECIMAL @INDIRECTION @SSN @VALUE @WINDOW @X
EFFICIENCY	GBLOCK		
HEADINGS/ PAGE CONTROL	HCC HIRCC LPP PC ECC PDATE TCC TITLE VIRCC	@PAGE @PC @PD	

OUTPUT DESTINATION:

Unless <file name> is specified in the PRINT command, all output will be directed to the file or device attributed to the OUTPUT facility. The default destination for the OUTPUT facility is *MSINK* (using SPRINT) which is the terminal in interactive mode or the printer in batch mode.

The output record length (ORL) defaults to 118. This means that MICRO assumes that the terminal has a printing capacity of 118 columns. Users with terminals having a different printing width should use the SET command to change the output record length.

If the output line is greater than the output record length, the line is broken into several smaller lines within the ORL limit. The division is made between fields and the output "wraps around" from one line to the next.

WHICH RECORDS ARE TO BE PRINTED:

If the @COUNT command modifier is not specified, then an attempt will be made to print all records in the data set. Printing can be interrupted whenever the user wishes. If the @COUNT command modifier is specified, then only that many records will be printed.

The status of the PRINTDIRECTION facility and the overriding command modifiers, @FORWARD and @REVERSE, determines whether the data set is to be read (and printed) from the first to the last record or from the last to the first record. The forward direction is the default.

The first record printed will either be the first or last record if the @FIRST command modifier is not specified. When @FIRST=<n> is specified, then the nth record from the front or back is printed first, depending on the print direction.

The normal increment in printing records is one. However, if @INCREMENT=<integer> is specified, the increment is changed to <integer>, causing records to be skipped.

RECORD FORMAT: There are three ways that the records of a data set may be printed horizontally, vertically, or a combination of both. Horizontal format is the default, but either horizontal or vertical can become the default via the SET command. If HCRIZCNTAL is SET to ON, then VERTICAL is automatically SET to OFF, and vice versa. The @VERTICAL and @HCRIZCNTAL command modifiers may be used to override the default format.

Horizontal format means that each record is written on one line, with headings placed above the first record to identify the columns of information. For example:

	FIELD-1	FIELD-2	FIELD-3	...	FIELD-m
	-----	-----	-----		-----
(record 1)	value	value	value	...	value
(record 2)	value	value	value	...	value
.	.	.	.	...	.
.	.	.	.	...	.
(record n)	value	value	value	...	value

If the print line exceeds the output record length (CRL), the printing will wrap around to subsequent lines.

Vertical format means that the value associated with each field of a record is written on a separate line, with an extra line between records. For example:

```

FIELD 1 = value (in first record)
FIELD 2 = value (in first record)
.
.
.
FIELD m = value (in first record)

FIELD 1 = value (in second record)
FIELD 2 = value (in second record)
.
.
.

```

FIELD n = value (in second record)

.
.
.

Obviously the vertical format requires more printing and more pages than the horizontal format. However the vertical format is very useful for reviewing data that would otherwise wrap around under the horizontal format.

All data sets in MICRO are sorted. It is possible that the first one or more fields of a data set may not change very often from record to record; that is, the first few fields may have the same values for many records. Consider a data set whose first field is year and whose second field is month. The value printed for year would change only after going through a whole year's data.

The stratification format was designed to make more efficient use of the printed page by not repeating the redundant information in the first few fields of a data set. Information is printed only when the value of one or more of these first few fields is different from the previous record. The <integer> associated with the @STRATIFICATION=<integer> indicates the number of fields, starting with the first, that are likely to contain redundant information. The first fields are printed in vertical format and the remaining fields are printed in horizontal format. If the level is 2, then it would appear as follows:

	FIELD 1 = value			
	FIELD 2 = value			
	FIELD-3	FIELD-4	...	FIELD-n
	-----	-----		-----
(record 1)	value	value		value
(record 2)	value	value		value

This format would continue until the value of either FIELD 1 or FIELD 2 changed, at which time it would begin again.

The output of the PRINT command examples appears in Appendix A and should be studied if further clarification is needed.

FIELD FORMAT:

If none of the command or field modifiers affecting field format are specified, the format of each field is determined solely by its field type.

If the field type is U (unsigned integer), UC (unsigned integer with categories), S (signed integer), or SC (signed integer with categories), the output processing is similar. If the field has categories, a sequential search is made of the category list, comparing the value of each category with the value of the field from the record in the data set. If the values match, the corresponding category name is printed, right justified. If the value does not match with a category or if the field does not have categories, then the scaling factor and function, if defined, are applied and the value is printed, right-justified under the heading.

If the field type is CC (character string with categories), a sequential search is made of the category list, comparing the character string of each category with the character string of the field from the record in the data set. If a match is made, the category name is printed, left-justified. If the character string is not matched, the processing continues as if it were a C type (character string without categories) and the string is printed, left-justified under the heading.

If the type is E (external data), the value of the field (a line number or notepointer) is used to read from the associated MTS file. The string of characters stored at the associated line number or notepointer is read and printed, left-justified under the heading. The string is printed only within the window defined by the SET WINDOW= <integer> command. The string is divided into sections (breaking on blanks if possible), so that no section is longer than <integer>. The sections are printed under the heading on as many lines as are required.

The command modifiers affecting field format perform straightforward variations on this general output procedure. If @ABBREVIATION is specified as a command modifier, abbreviations for all field names are used in the heading and category abbreviations are to be used instead of category names. If @DESCRIPTION is specified as a command modifier, category descriptions are printed instead of category names. If @HEXADECIMAL or @X is specified as a command modifier, the hexadecimal representation of the value stored is printed. No search will be made for categories or external data. If @INDIRECTION is specified as a command modifier, then a category or external data is printed whenever possible. (See the SET PVALUE(print value)=ON/OFF command description.) If @VALUE is specified as a command modifier, no search will be made for categories and the value of the field will be printed. If an E type field is in the data set, the value will be printed instead of the external data. If @WINDOW=<integer> is specified as a command modifier, <integer> will be the window width for all appropriate fields, overriding, for this command only, the global width set via the WINDOW facility.

Field modifiers apply only to the fields which they modify and override the status of any similar command modifier or facility. More than one field modifier may be specified for a field. The last field modifier affecting format will be used.

The @ABBREVIATION, @DESCRIPTION, @HEXADECIMAL, @INDIRECTION, @VALUE, @WINDOW and @X field modifiers are just like their command modifier counterparts, except that they apply only to the field to which they are attributed. The @DATE field modifier converts an integer value stored in YYMMDD format into MM-YY-DD format for printing. The @SSN field modifier converts an integer value into XXX-XX-XXXX format for printing.

EFFICIENCY:

The costs associated with printing large data sets can be reduced. Large data sets are those whose data file exceeds about 25 pages of 4096

bytes. This can be accomplished by increasing the size of the buffer (memory) used to pass data between STDS and MICRO. The larger the buffer, the fewer the number of input/output operations required to pass the data to MICRO for printing. Conversely, as the buffer becomes larger, the overhead costs associated with virtual memory increase. Thus the buffer should be increased prior to printing large sets and decreased as soon as these printing operations are completed.

The buffer size is changed by the SET GBLOCK=<integer> command. GBLOCK stands for "get block of data." The integer associated with GBLOCK represents the number of eight-page blocks in the buffer. Therefore, if GBLOCK=1, the default case, eight pages are transferred at a time. If GBLOCK=8, 64 pages are transferred at a time.

MICRO will not create a buffer larger than the data set it is about to print, even if GBLOCK is set so that the buffer size will be larger than the data set. If the @COUNT=<integer> command modifier is specified, the exact amount of storage needed is obtained if it is less than the size of the buffer specified by GBLOCK. Otherwise, the buffer size is 8 pages or the exact size of the data set, whichever is less.

HEADINGS/PAGE CONTROL:

All output from MICRO is single-spaced; that is, it has a blank for a carriage control character, with the following exceptions:

- (1) The spacing between records in vertical format is determined by the VIRCC (vertical interrecord carriage control) facility, and defaults to zero which is interpreted as double spacing. The spacing between records in horizontal print format is determined by the HIRCC facility (horizontal interrecord carriage control) which defaults to blank (single-spacing).
- (2) The first record of each STRATIFICATION group uses the PCC facility (page carriage

control), which defaults to 1, (skip to the top of the next page). On many terminals, this results in skipping six spaces.

- (3) The HCC facility (heading carriage control) is used as the carriage control character for the MICRO generated heading. The HCC defaults to zero (double space).
- (4) The character string TITLE is prefixed by the title carriage control character (TCC).

The HCC, HIRCC, PCC, TCC and VIRCC facilities can be changed by the SET command.

All printing from MICRO is done without regard to the end of the page or the beginning of the next, except as just stated. MICRO will keep track of the lines printed per page and will repeat the heading at the top of each subsequent page when PC (page control) facility is SET to ON or the @PC command modifier is specified. The number of lines per page (LPP) defaults to 60. Explicitly SETting LPP also enables the page control facility.

When page control (PC) is enabled, a title will be printed at the top of each page of the output if the TITLE has been SET. The heading is printed after the title.

The @PD[=<integer>] command modifier will cause the page number and the date to be printed at the top of each page. <integer> specifies the number to be used as the starting page number. Furthermore, @PD will enable page control (PC). The date printed will be the date of the run, unless PDATE has been SET, in which case the character string entered will be used. (SET PDATE='character string' need not be a date.)

If the @PAGE is specified as a command modifier, the first character of the first line will be PCC, usually forcing a skip to the top of the next page.

EXAMPLES: PRINT IN IT NAME, CLYR. [B-3]

 PRINT STUDENTS. [B-2]

 P IN IT CCUNT. [B-6]

 P IN IT NAME. [B-4]

 P IN NOT-JR COUNT. [B-6]

 P IN STUDENTS NAME AND STUDENT-ID-NO AND BDAY. [B-2]

 P IT. [B-3], [B-6]

 F. [B-3], [B-18]

 P@A. [B-8]

 P@A@W=60. [B-8]

 P@C=10 IN FIN-AID SOP, SOP@ABBR, SOP@VALUE, [B-2]
 SOP@ABBR@HEX, SOP@VAL@SSN, SOP@VA@DATE, SOP@DESC.

 P@C=10. [B-2]

 P@C=5 IN IT AMOUNT, INCREASE, NEW.AMOUNT, NEWCODE, [B-13]
 DATESTAMP.

 P@C=5 IN RESULT KEY, SIN@SSN, NAME, [B-7]
 DATESTAMP@DATE.

 P@C=5 RESULT. [B-16]

 P@C=5 STUDENTS. [B-16]

 P@C=5. [B-7], [B-13]

 P@C=6 IT. [B-7]

 P@C=7 IT. [B-7]

 P@COUNT=5 IN STUDENTS NAME, SIN@SSN, BDAY@DATE. [B-2]

 P@FIRST=20@INCREMENT=5 IN STUDENTS SIN, NAME. [B-2]

P@REVERSE@FIRST=5 IN STUDENTS NAME,SIN@SSN,
B@AY@DATE. [B-2]

P@ST=2@C=10 IT. [B-10]

P@STRAT=2@COUNT=60@PD IN IT RACE,SEX,CLYR,NAME,
B@AY@DATE,SIN@SSN. [B-1]

P@V IT. [B-13]

P@C=3. [B-10]

P@C=5 IT. [B-9]

P@V@C=5 RECIPIENTS. [B-10]

P@V@C=5. [B-2]

ERROR MESSAGES: 'EXTERNAL FILE NOT AVAILABLE.'
The MTS disk file containing the external
information is not available. The other
specified fields will be printed.

'"FIRST" SPECIFIED EXCEEDS CARDINALITY.'
The command modifier "FIRST" specifies a number
larger than exists in the data set specified.

NOTEWORTHY USAGE STRATEGY:

Prior to PRINTing a long listing on *PRINT* or
into a file, the user should print a few lines on
his terminal to verify the output.

250
PRINT Command

MICRO Version 5.0
October 1977

PURGE

COMMAND DESCRIPTION:

PURPOSE: To release all data sets and reinitialize the list of available data sets to those permanent sets listed in the user's directory file.

PROTOTYPE: PURGE.
Those sets which were obtained by using the GET command are removed from the universe of sets and the user's directory is reread.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

All temporary data sets, including the result set if it exists, are released.

EXAMPLE: PURGE. [B-10]

NOTEWORTHY USAGE STRATEGY:

This command can be used to force MICRO to re-read the user's directory file after the beginning of a MICRO session.

252
PURGE Command

MICRO Version 5.0
October 1977

READ

COMMAND DESCRIPTION

- PURPOSES:
- (1) To read an array containing binary and/or character records (which need not be sorted or unique) into the result set using the dictionary from the last explicitly named data set.
 - (2) To read an STDS set (see Section 2.2) into the result set using the dictionary from the last explicitly named data set.

PROTOTYPE: READ [ARRAY] FROM <file name>.

If ARRAY is specified, an array in <file name> is read using the record length implied in the dictionary. It is then sorted, weeded, and placed in the result set using the dictionary from the last explicitly named data set. If ARRAY is not specified, an STDS set is read from <file name> using the dictionary from the last explicitly named data set.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

- COMMENTS:
- (1) The last explicitly named data set can be set by the USE command.
 - (2) When ARRAY is specified, the command is similar to the GARRAY command (subroutine call) in STDS.
 - (3) When ARRAY is not specified the command is similar to the GET command (subroutine call) in STDS.
 - (4) READ ARRAY FROM <file name> is used as part of the customized programming methodology for data definition.

EXAMPLE: READ ARRAY FROM T-STUDENTS.

[A-6]

ERROR MESSAGES: 'FAILURE IN DATA SET READ IN: PLEASE TRY AGAIN.'
Usually preceded by a more explicit error
message.

 'YOU MUST PRECEDE A READ COMMAND WITH A USE
CCMMAND.'
Indicates that no implicit dictionary was
available.

RELEASE

COMMAND DESCRIPTION

PURPOSE: To release one or more data sets from active status.

COMMAND SYNONYM: R

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

The specified data set(s) are released. AND may be replaced by a comma (,).

(2) RELEASE ALL [DATA] SETS.

Where DATA is an optional phrase added for clarity.

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

(3) RELEASE [*].

This is synonymous with (2).

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

WHAT HAPPENS:

- (1) If the specified data set is not active, the command has no effect.
- (2) If the specified data set is active and is a permanent set then it is made inactive.
- (3) If the specified data set is active and is a temporary set, it is destroyed and cannot be referenced again.

COMMENTS: (1) Prototypes (2) and (3) differ from the PURGE command in that the information from other users' directories is not lost.

- (2) No more than 20 data sets (permanent and temporary) may be active at any one time. When the user attempts to create the twenty-first data set, the command will be cancelled and he will be told to destroy some sets before processing can continue. The user must then repeat the command. Saving an active data set makes it permanent and inactive and therefore reduces the number of active data sets. See the SAVE command description.

EXAMPLE: RELEASE FIN-AID. [B-5]

NOTEWORTHY USAGE STRATEGIES:

Permanent data sets which the user does not expect to reference again should be RELEASED in order to reduce the costs associated with keeping a data set active (see technical remarks below). This does not, however, preclude further reference to this permanent data set at a later time.

TECHNICAL REMARKS:

There is a buffer associated with each MTS file that is left open including the data file of each active set. If the data file is less than five pages in size, the buffer equals the size of the data file. If it is greater than or equal to five pages then a five page buffer is allocated. Additionally, the dictionary of each active set is stored in virtual memory. It pays to RELEASE unneeded sets.

REMOVE

COMMAND DESCRIPTION

- PURPOSE:
- (1) To store in the result set those records of a data set that do not meet the specified criteria.
 - (2) To store in the result set those records of one data set not found in another data set.
- PROTOTYPES:
- (1) REMOVE IN|FROM <data set name> WHERE <phrase>.
Where <phrase> refers to any phrase acceptable to the FIND command.
All records which do not meet the criteria specified in <phrase> will appear in the result set.
 - (2) REMOVE <data set name 1> FROM|IN <data set name 2>.
The field type and length of the fields of <data set name 1> and <data set name 2> are compared for compatibility. If they are compatible the result set will contain only those records in <data set name 2> which are not also contained in <data set name 1>.
- APPLICABLE COMMAND MODIFIERS: None.
- IMPLICIT EFFECT ON THE RESULT SET:
- A new result set is created and the old result set, if one existed, is released.
- COMMENTS:
- (1) This command is equivalent to the relative complement concept in set theory.
 - (2) The number of records extracted, the percentage of records extracted and the number of elements in the result set are printed.
- EXAMPLES:
- REMOVE FROM SMITH WHERE NAME IS 'SMITH MARY'.
[B-4]
- REMOVE FROM NCT-JR.
[B-6]

REMOVE IT FROM NOT-JR. [B-6]

REMOVE IT FROM SMITH. [B-4]

ERROR MESSAGES:

'BETWEEN' CLAUSE SPECIFIES AN EMPTY INTERVAL.'
MICRO assumes that when an empty interval is specified, i.e. YEAR IS BETWEEN 1977 AND 1978, that the user has made an error and that the command is cancelled.

'DATA SETS ARE NOT COMPATIBLE.'
Indicates that the records of the two specified sets are of a different length.

'ERROR - NO CONDITION SPECIFIED!'
Indicates that the command was incomplete in that no criteria was specified when one was required.

'FIELDS ARE NOT COMPATIBLE.'
Indicates that a field to field comparison cannot be performed because the two fields are not comparable.

'INTERVAL FCB < OR > IS LOGICALLY EMPTY.'
MICRO assumes that when an empty interval is specified, i.e. AGE < 0, that the user has made an error and that the command is cancelled.

'IS NOT BETWEEN' IS NOT AVAILABLE.'
'IS NOT BETWEEN' is not a defined operation.

REPLACE

COMMAND DESCRIPTION

PURPOSE: To replace the data in a data set with the data from a second identically structured set.

PROTOTYPE: REPLACE <data set name 1> WITH <data set name 2>.
The data in the first data set is lost and replaced by the data from the second data set.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

WHAT HAPPENS WHEN <data set name 1> IS A TEMPORARY SET:

- (1) The type and length of all fields of each data set are compared for compatibility. If they are not compatible, an error message is printed and the command is cancelled.
- (2) If they are compatible, <data set name 1> is destroyed and <data set name 2> is given the name of <data set name 1>.
- (3) If <data set name 2> was temporary, it is destroyed. If it was permanent, it exists as an inactive permanent data set named <data set name 2> and as an active temporary data set named <data set name 1>.

WHAT HAPPENS WHEN <data set name 2> IS A PERMANENT SET:

- (1) <data set name 1> is checked to see if it is replaceable (see Section 4.1.4). If it is not, an error message is printed and the command is cancelled.
- (2) The type and length of all fields of each data set are compared for compatibility. If they are not compatible then an error message is printed and the command is cancelled.
- (3) If <data set name 1> is replaceable then a new MTS file named L#S is created large enough to

hold the data in <data set name 2>. If sufficient permanent disk storage is not available, an error message will be printed and the command will be cancelled.

- (4) If L##S is created then the data associated with <data set name 2> is written into that file.
- (5) Then the data file associated with <data set name 1> will be emptied. If this file cannot be emptied because the user is not permitted in MTS, an error message is printed, L##S is destroyed and the command is cancelled.
- (6) Once the data file is emptied, the contents of L##S are copied into it. When the copying is completed successfully, L##S is destroyed.
- (7) If <data set name 2> is a temporary set, it is destroyed. If it is a permanent set, it becomes inactive.

- COMMENTS:
- (1) If the system should crash while replacing a permanent set, the user will either be left with the old or new data in the data file associated with <data set name 1> or with the new data in L##S. The old and new data cannot both be lost.
 - (2) <data set 2> may be a null set.

EXAMPLE: REPLACE STUDENTS WITH IT. [A-6]

- ERROR MESSAGES:
- 'DATA SETS ARE NOT COMPATIBLE.'
Indicates that if the REPLACE were allowed that the dictionary and data would not correspond.
- 'IMPROPER MTS ACCESS.'
Indicates that the user does not have MTS empty access to the original data file.
- 'ORIGINAL DATA SET STORED ON TAPE.'
Indicates that the original set is stored on tape and may not be REPLACED.
- 'PROBLEM WITH OLD FILE.'
Indicates that the original data file does not exist or the user does not have MTS read access.

'YOU ARE NOT ALLOWED TO REPLACE THAT DATA SET.'
Indicates that this user does not have MICRO
REPLACE access to that data set.

NOTEWORTHY USAGE STRATEGIES:

- (1) This command is most likely to be used to update the permanent copy of a data set after the loaded copy of a permanent set has been modified.
- (2) Replacing a permanent data set is usually less efficient (slightly more costly) than SAVING another copy of the same data set. It is more convenient and straightforward to REPLACE. If one is going to DESTROY the old data set after saving the new one and then rename it back to the original name then it costs about the same.
- (3) A cheaper alternative to the replace command involves WRITING a new data file and then using the MTS destroy and rename commands to give the recently created file the same name as the old data file. Though not as convenient, this procedure saves the cost of copying the data from I#S to the proper file. When using the WRITE command, the active core copy of the set (if it was loaded) is left unchanged. Thus it should be RELEASED or the PURGE command should be issued.

262
REPLACE Command

MICRO Version 5.0
October 1977

RESET

COMMAND DESCRIPTION

PURPOSE: To restore certain MICRO facilities to their default value or status.

- PROTOTYPES: (1) RESET INPUT.
Input to MICRO will be received from *MSOURCE*.
- (2) RESET OUTPUT.
Output from MICRO will be written to *MSINK*.
- (3) RESET HIRCC.
- (4) RESET VIRCC.
- (5) RESET HCC.
- (6) RESET PCC.
- (7) RESET TCC.
- (8) RESET.

The facilities listed on the following page are all restored to their default value or status:

<u>FACILITY</u>	<u>DEFAULT VALUE OR STATUS</u>
CLCK	OFF
COST	OFF
ECHO	ON
GBLOCK	1
HCC	0
HIRCC	BLANK
HORIZONTAL PRINT	ON
INPUT	*MSOURCE*
IPE	60
MACRO ECHO	OFF
OUTPUT	*MSINK*
PC	OFF
PCC	1
PERCENT	ON
PRINTDIRECTION	FORWARD

READY	ON
TCC	0
TITLE	OFF
VERTICAL PRINT	OFF
VIRCC	0
WINDCW	20
XTAB % PRECISION	100
XTAB PRINT	ON

IMPLICIT EFFECT ON THE RESULT SET: None.

RESTRICT

COMMAND DESCRIPTION

- PURPOSES:
- (1) To store in the result set records of a data set for which values of the specified key fields match the values of the specified key fields in a second data set.
 - (2) To extract from one data set those records whose key fields do not appear in a second data set.

- PROTOTYPES:
- (1) RESTRICT IN <data set name 1> WHERE
<field name A> [AND <field name B> ...] IS|ARE
<field name M> [AND <field name N ...] IN
<data set name 2>.

Those records in the first data set whose key value (the value combination of <field name A>, <field name B>, etc.) is the same as any key value (the value combination of <field name M>, <field name N>, etc.) of the second data set are placed in the result set.

- (2) RESTRICT IN <data set name 1> WHERE
<field name A> [AND <field name B> ...] IS|ARE
NOT <field name M> [AND <field name N> ...]
IN <data set name 2>.

Those records in the first data set whose key value does not match the key value of any record in the second data set are placed in the result set.

APPLICABLE COMMAND MODIFIERS: None.

EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

- COMMENTS:
- (1) The fields which determine the keys must be ordered in the command so that fields of the same type and length are being compared.
 - (2) If <data set name 1> and <data set name 2> are reversed, a different result set will be created

consisting of records extracted from <data set name 2>.

(3) AND may be replaced by a comma (,).

EXAMPLES:

RESTRICT IN STUDENTS WHERE SIN IS SIN IN
FIN-AID. [B-9]

RESTRICT IN STUDENTS WHERE SIN IS NOT SIN IN
FIN-AID. [B-9]

RESTRICT IN BIGJOIN WHERE SIN IS SIN IN NO-AID.
[B-10]

RESTRICT IN RECIPIENTS WHERE SIN IS SIN IN
FIN-AID. [B-10]

ERROR MESSAGES:

'ERROR - FIELDS FOR 2ND SET ARE INCONSISTENT.'
The fields listed for the second data set must be the same length as the corresponding field listed for the first data set.

'NUM OF FIDS SPECIFIED MUST BE THE SAME FOR EACH SET.'
Indicates that a different number of fields were specified for each data set.

'SAME FIELD SPECIFIED MORE THAN ONCE!'
Indicates that the same field name was specified more than once for one of the data sets.

NOTEWORTHY USAGE STRATEGY:

In restrictions involving more than one field, it is desirable to put the field that is most restrictive first. The most restrictive field is the one that eliminates the most records from <data set name 1>. This will reduce the cost of the operation by a significant amount. For example, consider the command RESTRICT IN FIN-AID WHERE SIN AND TERM IS SIN AND TERM IN IT. SIN is the more restrictive field because more records will be deleted by the SIN field than by the TERM field.

SPECIAL DESIGN CONSIDERATION:

Care should be taken to insure that the same

field type and length are given to fields appearing in data sets which the user might need to use in RESTRICT (or RESTRICTANDJOIN) commands. For example, if UNIT-CODE is defined as a fullword integer in one data set and as a one byte integer in another data set, the RESTRICT command cannot be executed with UNIT-CODE as a specified field.

RESTRICTANDJOIN

Command Description

PURPOSE: To create a result set containing records formed by merging only those records in the two data sets which have matching keys.

COMMAND SYNONYM: RAJ, BAM

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

The key value of each record in the first data set is compared with the key value of each record in the second data set. When the two values match, an expanded record is created and added to the result set. Each expanded record consists of the key fields of the first data set followed by the non-key fields in the first data set followed by the non-key fields of the second data set.

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

The action is the same as in (1) above. However, the field name(s) which make up the key for the second data set are assumed to have the same name(s) as the field name(s) which constitute the key of the first data set.

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

This prototype is like (2) above except that the field name(s) which make up the key are explicitly stated for the second data set and implied for the first data set.

(4) RESTRICTANDJOIN <data set name 1> WITH
<data set name 2>.

In this prototype no key is specified and each

record of data set 1 is expanded with each record of data set 2. This results in the cartesian product of the two sets.

APPLICABLE COMMAND MODIFIERS:

@GF=<integer>

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is lost.

- COMMENTS:
- (1) The fields which make up the keys must be ordered so that fields of the same type and length are being compared.
 - (2) The number of records in the result set depends on the number of matches on the key between the data sets and on whether the key uniquely identifies each record of the data sets. If the values of the key fields are unique for each data set (i.e., there will be a one-to-one correspondence between records with matching keys) then the number of records in the result set will be less than or equal to the number of records in the smaller of the two data sets. If the values of the key fields are not unique for one or both of the data sets (i.e., there is a one-to-many or a many-to-many correspondence between records with matching keys) then the number of records in the result set can be as great as the product of the number of records in the two data sets.

An example of a one-to-one relationship is between a person and his fingerprints. When matching a person data set with a fingerprint data set, there should be no more than one match per person. Thus, the resulting data set is never any larger than the number of records in the smaller data set.

An example of a one-to-many relationship is between a person and his telephone number. Most people will have one but some will have more than one or none. A match will occur for each telephone number associated with a person. Thus,

the resulting data set will never be any larger than the number of records (telephone numbers) in the larger data sets.

An example of a many-to-many relationship is between people and children. One person may be the parent of more than one child and one child may be the offspring of more than one parent. Large growth can occur in this situation with the largest possible set being the cartesian product of the two sets (the product of the cardinality of each set). This occurs when the key field(s) for each record in one set matches the key field(s) of all records in the second set.

It is possible to create an extremely large set as the result of this command. Unless GROWTH FACTOR is increased via the SFT command or the GF command modifier, MICRO will not always successfully create a set with more than two times larger than the storage required by the larger of the two data sets. The permissible values for GROWTH FACTOR are 1 to 9.

- (3) The RESTRICTANDJOIN command ignores the crosstabulated status of data sets. Hence, the result of this command will never be a crosstabulated set.
- (4) If there are non-key fields in both data sets with the same name the result set will contain more than one field with the same name. The second and additional fields of the same name cannot be explicitly referenced without either removing or changing the name of the first occurrence.
- (5) The result set from a RESTRICTANDJOIN command does not contain the descriptions for the fields and categories appended from the second data set. Therefore, no description will appear in dictionary documentation including the DESCRIBE, DOCUMENT and FULLDOC commands.
- (6) In previous versions of MICRO, this command was known as RESTRICTANDMERGE. It has been renamed for consistency with the JOIN command.

EXAMPLES: RESTRICTANDJOIN STUDENTS WITH FIN-AID BY SIN.
[B-10]

ERROR MESSAGES: 'INCOMPATIBLE DATA SETS - FIELD LENGTHS.'
Indicates that the two specified sets have at least one key field of differing length.

'INCOMPATIBLE DATA SETS - FIELD NAMES.'
Indicates that an implicitly specified field name does not exist.

'NO DATA SET #1 SPECIFIED.'

'NO DATA SET #2 SPECIFIED.'

'RESULT SIZE LIMIT EXCEEDED.'
Indicates that the result of the command grew more than was allowed; see the growth factor command modifier discussion.

'TOO MANY DATA SETS SPECIFIED.'
Indicates that more than two data sets were specified.

'UNABLE TO PROCESS EMPTY SET.'
Indicates that a prototype (4) was requested and at least one of the sets was empty.

'UNEQUAL FIELD LIST.'
Indicates that the number of fields specified for the two sets differ.

SPECIAL DESIGN CONSIDERATION:

Care should be taken to insure that the same field type and length are given to fields appearing in different data sets which the user might wish to RESTRICTANDJOIN together. For example, if UNIT-CODE is defined as a fullword integer in one data set and as a one byte integer in another data set then the RESTRICTANDJOIN command cannot be executed with UNIT-CODE as one of the specified key fields.

SAVE

COMMAND DESCRIPTION

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

PROTOTYPE: SAVE <data set name> [AS <new data set name>]
[ON <file name>] [AFTER SCRAMBLING]
[WITHOUT|W/O [MODIFICATION]].

Where <new data set name> is the name given to the new permanent set. If <new data set name> is omitted, then the name of the permanent set will be <data set name>.

Where <file name> is the name of the file to be created for the data associated with <data set name>. The file name can be no more than eleven characters. A # will be appended to the specified file name in order to create the dictionary file associated with <data set name>. If <file name> is omitted, <data set name> is used as the data file name.

Where AFTER SCRAMBLING indicates that the data set is to be scrambled (see Section 4.1.9) before permanently storing it on disk. It is not necessary to specify AFTER SCRAMBLING for a data set derived from a set which was scrambled.

Where WITHOUT MODIFICATION gives the new data set the same data set description, destroyable status and replaceable status as its parent data set. If the WITHOUT phrase is omitted, the user will be asked to enter a description of the new set and to state whether the new set can be replaced or destroyed.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENTS: (1) If either the dictionary or data file to be created already exists then permission will be requested to empty the file. YES indicates affirmation. If denied, another name for the

file will be requested.

- (2) The name of the data set to be SAVED cannot be the name of an existing permanent data set.
- (3) The data and CCID from which a data set was SAVED are appended to the data set description regardless of whether the description is explicitly entered or taken from the parent set.

EXAMPLES: SAVE BIGJCIN AS JOINED-SET. [B-10]
 SAVE RECIEPIENTS. [B-10]

ERROR MESSAGES: 'FAILURE IN DATA SET SAVE: PLEASE TRY AGAIN.'
 Usually preceded by a more explicit error
 message.

 'IILEGAI TO SAVE SET NAMED RESULT!'
 The data set "RESULT" must be renamed or given a
 new name before being SAVED.

 'THIS DATA SET IS ALREADY A PERMANENT DATA SET!'
 It is illegal to SAVE a permanent data set.

SELECT

COMMAND DESCRIPTION

PURPOSE: To generate a result set from the specified data set which contains a change in the order of the fields and/or reduction in the number of fields.

PROTOTYPES: (1) SELECT IN <data set name> <field name> [AND <field name> ...].

Only the fields specified will appear in the result set in the order specified.

(2) SELECT IN <data set name> ALL BUT <field name> [AND <field name> ...].

All fields except those specified will appear in the result set in the order that they occurred in the specified data set.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

WHAT HAPPENS:

(1) The result set is generated from the specified or implied fields and is sorted in logical ascending order from left to right.

(2) The result set is "weeded"; that is, duplicate records are eliminated. No count is made of the number of records that are "weeded" as in the CROSSTABULATE command. However, the message "WARNING: DATA LOSS HAS OCCURRED" is printed if "weeding" occurs.

COMMENTS: (1) AND may be replaced by a comma (,) or BY.

(2) The set resulting from a SELECT on a crosstabulated set is not a crosstabulated set. Hence the result set of this command will never be a crosstabulated set.

EXAMPLES: SELECT IN STUDENTS NAME, SIN. [B-7]

 SELECT IN RECIPIENTS YEAR, TYPE, NAME, SOP, AMT. [B-11]

 SELECT IN STUDENTS SIN, NA, RACE, SEX, BDAY, CLYR, BDAY. [B-17]

 SEL IN IT ALL BUT NAME AND SIN. [B-14]

NOTEWORTHY USAGE STRATEGIES:

- (1) By SELECTing only those fields needed to attain a desired goal, the overhead associated with processing large data sets can be reduced. This overhead consists primarily of the input and output performed by MICRO in making a pass through a data set. The greatest savings can be made when a large percentage of the fields can be eliminated and/or when the cardinality (number of records) of the set is so large that eliminating a smaller percentage of the fields would have a noticeable effect on input/output. If this strategy is used, the parent set should be RELEASEd, if possible, for additional efficiency.
- (2) If the user desires to SELECT certain fields of a data set but does not want duplicate occurrences to be eliminated through weeding, he should first KEY the data set. The KEY field will give each record a unique identifier. If the KEY field is included in the list of fields SELECTed, there will be no loss of records.
- (3) 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 insure that duplicate records are not eliminated. If a count of the duplicate records is desired, then the CROSSTABULATE command should be used to prepare the result set instead of the SELECT command. If duplicates are desired for analysis, then the KEY command should be used to key the data set before using the SELECT command. (See (2) above.)
- (4) The same field may be selected several times.

This is sometimes useful preceding the CALL command to a user-supplied subroutine as a way of getting additional working space without specifically acquiring it dynamically from the subroutine.

278
SELECT Command

MICRO Version 5.0
October 1977

SET

COMMAND DESCRIPTION

PURPOSE: To change the status or value of a MICRO facility.

COMMAND SYNONYM: TURN

PROTOTYPES: (1) SET CLOCK [=] OFF|ON. (default: OFF)

When CLOCK is enabled the elapsed time and CPU time used in the execution of each command is printed.

(2) SET CMT [=] ON. (default: OFF)

When CMT is ON the command trace facility is enabled. Once enabled it cannot be turned off for the duration of the MICRO session. See the description in Section 2.8.1.

(3) SET [COMMAND]||[CPU] TIME = <integer>. (default: 10)

Where <integer> is the amount of time in seconds available for execution of each command before interruption. Interrupted commands can be restarted for another interval of time via the MTS \$RESTART command.

(4) SET COST|C [=] OFF|ON. (default: OFF)

When COST is enabled, the cost of executing each command is printed.

(5) SET ECHO [=] OFF|ON. (default: ON)

When ECHO is disabled the remarks following the execution of various commands (such as "XX RECORDS IN RESULT SET") are suppressed.

(6) SET GBLOCK|GB = <integer>. (default: 1)

The <integer> associated with GBLOCK represents the number of eight-page blocks of data passed from the data set to MICRO as part of the PRINT

command.

- (7) SET HCC [=] <character>. (default: 0)

This redefines the HCC (header carriage control) character. See PRINT command description.

- (8) SET HIRCC [=] <character>. (default: "blank")

This redefines the HIRCC (horizontal print format interrecord carriage control) character. See the PRINT command description.

- (9) SET HORIZONTAL PRINT [=] OFF|ON. (default: ON)

SET HP [=] OFF|ON.

When the horizontal print format is enabled, the vertical print format is disabled. See the description in the PRINT command description.

- (10) SET INPUT|I = <file or device name>.
(default: *MSOURCE*)

Input to MICRO will be received from (i.e., read on SCARDS) the file or logical device specified until an end-of-file is reached or until an attention interrupt is received. At these times input is reset to *MSOURCE*.

Caution should be exercised when using the ENTERDATA and UPDATE commands in the input stream assigned by the SET command. Because both of these commands also read data on SCARDS, field values for these commands should be placed in the specified input file or device. These commands reassign SCARDS to *MSOURCE* at their completion.

- (11) SET INPUT ECHO [=] ON|OFF. (default: OFF)

When INPUT ECHO is enabled, all MICRO commands are echoed on SPRINT.

- (12) SET LPP = <integer>. (default: 60)

The LPP (lines per page) that are printed is changed and the PC (page control) facility is enabled. See the PRINT command description.

- (13) SET MACRO ECHO [=] OFF|ON. (default: OFF)

When MACRO ECHO is enabled, every MICRO command generated by a macro is printed.

- (14) SET MASTER ID = <ccid>|OFF.

The MASTER ID is either changed or set off in the user's directory. There is no immediate visible effect on the current MICRO run. However, the next time MICRO is run from this CCID the newly set MASTER ID is used. See MASTER ID description in chapter 5.

- (15) SET ORL = <integer>. (default: 118)

The ORL (output record length) which controls the maximum output width is set. See the PRINT command description.

- (16) SET OUTPUT|O = <file or device name>.
(default: *MSINK*)

All output generated by MICRO is directed to the specified file or device.

- (17) SET PC [=] OFF|ON. (default: OFF)

When PC is set to ON, the page control facility is enabled. See the PRINT command description.

- (18) SET PCC [=] <character>. (default: 1)

This redefines the PCC (page carriage control) character. See the PRINT command description.

- (19) SET PDATE = '<character string>'|OFF.
(default: OFF)

PDATE (print date) defines the character string to be printed when the PD command modifier is specified on the PRINT command. If OFF is specified, the date on which the run occurs is printed. <character string> must be exactly 12 characters long; truncation or padding will be performed. See PRINT command description.

- (20) SET PERCENT|X|XTAB% [=] OFF|ON. (default: ON)

SET XTAB PERCENT [=] OFF|ON.

When the XTAB PERCENT facility is enabled, the PERCENT field is automatically computed as part of the CRCSSTABULATE command.

- (21) SET PRINTDIRECTION|PRINT DIRECTION =
FORWARD|REVERSE
(default: FORWARD)

The PRINTDIRECTION determines whether the PRINT command will print records in a data set from the first to the last, or the last to the first. FORWARD specifies the former, and REVERSE, the latter. See the PRINT command description.

- (22) SET PVALUE [=] ON|OFF. (default: OFF)

When PVALUE (print value) is enabled, the PRINT command prints the values (not categories or external data). A command or field modifier can be used to temporarily override the setting. See PRINT command description.

- (23) SET READY [=] OFF|ON. (default: ON)

When the READY facility is disabled, the user will not be prompted for the next command by "READY". This is useful when multiple commands are being executed from a macro or input file.

- (24) SET RUNFILE [=] <filename>|OFF. (default: OFF)

The RUNFILE is either changed or set to OFF in the user's directory. There is no executable effect of this action until the next MICRO session run on this user's CCID. See RUNFILE description in Chapter 5.

- (25) SET TCC [=] <character>. (default: 0)

This redefines the TCC (title carriage control) character. See PRINT command description.

- (26) SET TITLE = '<character string>'|OFF.
(default: NULL)

Where <character string> is any series of characters, numbers, punctuation marks, etc. Two

consecutive primes (') are required to indicate a single prime within <character string>. OFF will indicate a null title. Primes are not required if there are no embedded blanks in the title. See the PRINT command description.

- (27) SET VERTICAL PRINT [=] OFF|ON.
(default: OFF)

SET VP [=] OFF|ON.

When the vertical print format is enabled the horizontal print format is disabled. See the PRINT command description.

- (28) SET VIRCC [=] <character>. (default: 0)

This redefines the VIRCC (vertical print format interrecord carriage control) character. See PRINT command description.

- (29) SET WINDOW|W = <integer>. (default: 20)

The window width which may be used in printing external data, descriptions and long character strings is changed to <integer>. See the PRINT command description.

- (30) SET [XTAB] PRECISION = 10|100|1000|10000.
(default: 100)

The precision of the PERCENT field resulting from the CROSSTABULATE command is changed. If 100 is specified, the PERCENT field is scaled to two places (i.e., it is divided by 100), etc.

- (31) SET XTAB PRINT [=] OFF|ON. (default: ON)

SET XTABPRINT [=] OFF|ON.

When the XTAB PRINT facility is enabled, the result of a CROSSTABULATE command will automatically be printed.

EXAMPLES: SET CLOCK CN. [A-1], [A-6], [B-1]
 SET COST=ON. [A-3], [A-6]

SET ORL=118. [B-3]

SET PCC=V1. [B-2]

SET TIME=1. [A-1], [B-1]

SET TITLE = 'A LISTING OF STUDENT RECORDS
STRATIFIED BY RACE AND SEX'. [B-2]

ERROR MESSAGES: 'INVALID VALUE.'
Indicates that the command failed because the
integer value specified was not acceptable.

'UNABLE TO CREATE DIRECTORY.'
Indicates that the command failed because an MTS
disk file name "DIRECTORY" did not exist and the
system could not create it.

IMPLICIT EFFECT ON THE RESULT SET: None.

SIGNOFF

COMMAND DESCRIPTION

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

PROTOTYPE: SIGNOFF [S|\$].

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

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

The result set and all other temporary data sets are released.

SORT

COMMAND DESCRIPTION

PURPOSE: To reorder the fields of a data set and, in doing so, to sort the records in the reorganized data set.

COMMAND SYNONYM: CEDER

PROTOTYPES: (1) SORT IN <data set name> <field name> [... BY <field name>].

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

The occurrence of the fields in the data set is reordered. The specified fields are first and are followed by the unspecified fields in the order that they appear in <data set name>.

(2) SCRT IN <data set name> ALL BUT <field name> [... BY <field name>].

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

The occurrence of the fields in the data set is reordered with the unspecified fields in the order that they originally appeared in <data set name> followed by the specified fields in the order of specification.

APPLICABLE COMBINE MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

A new result set is created and the old result set, if one existed, is released.

WHAT HAPPENS:

The order of the fields is changed as in the SELECT command. However, all fields are selected (either implicitly or explicitly) so there is no possibility of data loss. A logical left-to-right ascending sort is performed.

- COMMENTS:
- (1) The number of sorted records in the result set is printed.
 - (2) BY may be replaced by a comma (,) or AND.
 - (3) Signed integers are treated logically. Therefore, zero values will be followed by positive values and then negative values.
 - (4) Sorting on an external data field will be performed on the value (line number or notepointer) of the field and without any regard to the data stored in the external MTS file.
 - (5) The set resulting from the sort of a crosstabulated set is not a crosstabulated set.

EXAMPLES:

SORT IN IT CLYR, NAME.	[B-3]
SORT IN STUDENTS NAME.	[B-2]
SORT IN STUDENTS RACE, SEX, CLYR, NAME.	[B-2]

NOTEWORTHY USAGE STRATEGIES:

- (1) The same field may be specified several times. This is sometimes useful preceding the CALL command to a user-supplied subroutine as a means of getting additional working space without specifically acquiring it dynamically from the subroutine.
- (2) If one wants signed integer fields to appear as if they are sorted properly with negative numbers before positive numbers, the user should doubly select or sort the field with a negative range. (The two fields should be adjacent to each other.) Then the user should use the CHANGE command to change the first of the two fields so that all positive numbers become 2, all zeros become 1, and all negative numbers become 0. The resulting set will be sorted as desired. It's awkward, but the only way.

TECHDOC

COMMAND DESCRIPTION

PURPOSE: To document the technical dictionary information for one or all fields of a data set.

COMMAND SYNONYMS: IDOC

PROTOTYPES: (1) TECHDOC [DATA SET] <data set name>.

Where DATA SET is an optional phrase added for clarity.

All technical dictionary information for all of the fields in <data set name> is printed.

(2) TECHDOC IN|FOR <data set name>
[FIELD] <field name>.

Where FIELD is an optional phrase added for clarity.

All technical dictionary information, including any categorical information, for <field name> in <data set name> is printed.

(3) TECHDOC IN|FOR <data set name> FIELDS [ONLY].

Where ONLY is an optional phrase added for clarity.

Dictionary information, excluding categorical information, is printed for all fields.

(4) TECHDOC IN|FOR <data set name> *.

This is synonymous with (3) above.

(5) TECHDOC IN|FOR <data set name> [FIELD]
<field name> [CATEGORY] <category name>.

Where FIELD and CATEGORY are optional phrases added for clarity.

The dictionary information for <category name> of <field name> is printed.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

COMMENTS: (1) Dictionary documentation via the TECHDOC command
will include the following, if defined:

- (a) data set description;
- (b) field name;
- (c) field abbreviation;
- (d) field type;
- (e) data required status;
- (f) category name;
- (g) category abbreviation;
- (h) category type;
- (i) category value;
- (j) category required status;
- (l) field length;
- (m) field scaling function; and
- (n) field scaling factor.

(2) The dictionary documentation generated via the TECHDOC command is the same as that generated by the stand-alone MICROC.DOC program when TECH is specified. Both require 73 printing columns.

(3) More complete dictionary documentation can be generated using the FULLDOC command. Other commands which generate dictionary information are the TECHDOC, DESCRIBE and LIST commands.

EXAMPLES:	TDOC BIGJCIN.	[B-10]
	TECC IN IT *.	[B-17]
	TDOC IN IT SOA.	[B-12]
	TECHDOC IN FIN-AID SOP.	[B-12]

NOTEWORTHY USAGE STRATEGY:

The TECHDOC printout describes the data file (STDS set) record layout in a convenient format which will fit within 80 printing columns.

UPDATE

COMMAND DESCRIPTION

PURPOSE: To update a data set with new values for the same group of fields for all records matching the values of the specified key fields.

PROTOTYPE: UPDATE [IN|INTC] <data set name>
[PERMANENTLY|PERM] [FROM <file name 1>
WITH ERRORS ON <file name 2>].

Where INTC and IN are optional words which may be added for clarity.

Where PERMANENTLY and PERM are optional words which indicate that the permanent disk copy of the specified data set is to be updated. If PERMANENTLY or PERM is not specified, the updated records, combined with the unchanged records in the specified data set, will appear in the temporary result set.

Where the use of "FROM <file name 1> WITH ERRORS ON <file name 2>" is an optional phrase which indicates that the key field values and new field values will be in FREEFORM format. This raw data will be read from <file name 1> and any erroneous raw data records will be printed on <file name 2>. If this phrase is omitted, the user will be interactively prompted for the new key values and new field values.

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET:

If PERMANENTLY or PERM is specified, no changes are made to the result set. However, if neither is specified, a new result set is created and any previous result set is released.

WHAT HAPPENS: (1) After typing the command, the user will be asked to enter the names of fields used to identify records. The values given later for these fields will be used to identify the particular records which will be updated.

- (2) Next, the user will be asked to enter the names of fields to be updated or to type "STOP". All of the fields specified here will be updated in those records identified by the values of the key fields that were previously defined. When the user is through updating, he should type "STOP".
- (3) All of the field names collected up to this point are used to create an intermediate temporary data set (the updating set). Values for the fields of the updating set are obtained by using a facility very similar to the ENTERDATA facility to build a second intermediate temporary data set. Either interactive-prompting mode or batch free-format mode may be used, depending upon whether "FROM <file name 1> WITH ERRORS ON <file name 2>" is specified in the UPDATE command.
- (4) After the updating set is created, the values of the key fields in the updating set which do not match with any of the values of the key fields in the set to be updated (the set specified in the UPDATE command) are printed and the set is updated. The UPDATE command then asks again for the names of fields to be updated, as described in (2) above.

INTERACTIVE PROMPTING MODE:

If "FROM <file name 1> WITH ERRORS ON <file name 2>" is not specified, then certain rules apply:

- (1) Upon entering the interactive-prompting data-entry mode the user will be asked if he wants abbreviated prompting. "YES" or "Y" will result in the user being prompted for each field with the abbreviation for the field name. Any other reply results in the use of the full field name.
- (2) Periods (.) should not appear at the end of each line typed (as in MICRO command mode) once prompting for data begins.
- (3) Input for more than one field may be entered on one line if the user wants to anticipate the prompting for the next field(s). Prompting will

be suppressed for those fields for which data has already been provided in this manner. If too many values are provided then a warning message will be printed. The input for each field must be separated or delimited by one or more blanks. If there are one or more embedded blanks in the data for a given field then the data for that field must be surrounded by primes (''). If a prime is to be included in the string, two primes must be used to represent a single prime within the string. There is no way that the default value can be indicated for a field when data for more than one field is entered on a line.

- (4) Whenever an error message requests a new value for a specified field, the user may obtain documentation for that field and any of its categories by typing a question mark (?) in column 1.
- (5) If the user receives an error message requesting a new value for a specific field, only the one replacement value should be entered. The program will remember any additional values previously entered in anticipation.
- (6) After data values for all fields of a record have been entered, the user will be prompted for corrections to that record. The user may change the value assigned to any field by entering:

<field name> <verb> <value or category name>

Where <verb> may be an equal sign (=), comma (,), semi-colon (;), or colon (:). Only one field may be corrected per line but unlimited corrections are allowed. If no more corrections are required then a carriage return should be typed to begin data entry on the next record. If the user has completed his data entry activities, he should type DONE to return to MICRO.

- (7) "No reply" can be indicated by pushing the carriage return without typing any characters. Except as otherwise indicated, default values are obtained by giving no reply. When a reply is required, the program will try up to five times to get a reply before returning to MTS. If the

user wishes to try to reply again, \$RESTART should be issued.

- (8) All fields of a record which have yet to receive data and for which data is not required (as defined during dictionary definition) may be assigned their default value by transmitting an end-of-file indicator (via Control "C" on the terminal) or by including a \$ENDFILE in the input stream. The user will still be prompted for values for all of the remaining required fields. This indicator applies only to the current record and must be repeated for each record for which this option is desired. This indicator is identical in function to the /D in batch free-format mode.
- (9) A /S (for same) may be used as a value only for external data fields when the user wants to substitute the character string (or line number) from the previous record's input that was used as data for the same external data field. The effect is that the line number or notepointer associated with that previous character string is used as the external data field's value.

BATCH FREE-FORMAT MODE:

If "FROM <file name 1> WITH ERRORS ON <file name 2>" is specified, the rules of batch free-format mode apply:

- (1) Instead of prompting for data, data is simply read from <file name 1> with the data in the same field order as in interactive-prompting mode.
- (2) More than one value may be placed on a single line if the values are separated by blanks. Character string values with embedded blanks must be enclosed in primes ('). Single lines are processed as they are read unless a minus sign is placed in the last column transmitted before the carriage return, in which case the field is assumed to be continued in column 1 of the next input line.
- (3) A record in the file may be up to 32,767 characters long and extend over as many lines as

the user desires.

- (4) An end-of-record indicator denoting the end of input for each record is required after the last value. /E is the end-of-record indicator and must be the last characters (except for trailing blanks) in the last line for each record. Input for the first field of each record must begin on a new line. There must be sufficient data prior to the /E for all fields. If too many or too few values are provided before the end-of-record indicator, the record will be rejected and an error message will be printed.
- (5) Unlike the prompting mode, there is no way to give "no reply" for one particular field in order to give the field its default value.
- (6) All fields of a record which have yet to receive data and for which data is not required (as defined during dictionary definition) may be assigned their default value by placing a /D in the input string. The user must still provide values for all remaining required fields in that record. The /D is identical in function to the \$ENDFILE in interactive prompting mode.
- (7) A /S may be used as the value for an external data field when the user wants to refer to the same line in the external data file as the external data field did in the previous record. The effect is that the line number or notepointer associated with that previous character string is used as the external data field's value.
- (8) When the data provided for a field is interpreted to be incorrect, the program will identify the error (as in interactive-prompting mode). Then the line number containing the error, the line number of the first input line for the aborted record and all of the input lines (with their line numbers) for the aborted record will be printed on the terminal (or elsewhere if SERCOM has been reassigned). The input lines for the aborted record are also copied into <file name 2> at their same line numbers. Thus, when data entry is complete, <file name 2> is ready to be edited and used as proper input data in another

UPDATE command.

- (9) When all data has been read from <file name 1>, the number of records read, rejected and added to the data set being updated will be printed in addition to the new total number of records in the data set.

ATTENTION INTERRUPTIONS:

The user may interrupt at any time during the data collecting activities. If the user is in interactive-prompting mode and data entry has begun, then the user will be asked, "Do you want to add more data?" "YES" will start a new record. "NO" will cause the specified set to be updated using all completed records. In either case, the current record is aborted as soon as the interrupt is received.

During certain critical operations, attention interrupts are ignored.

DATA VALUES FOR SCALED FIELDS:

Data values entered for scaled fields should be entered in the way the user views the data, not the way MICRO stores it as an integer. For example, if AMOUNT was an unsigned integer field scaled by division by 100 (two decimal places as in dollars and cents), then 100 and 100.0 and 100.00 would all be viewed as 100.00, with the internal representation being 10000. If the user wants to indicate three dollars and forty-six cents, then 3.46 should be entered, not 346.

SCRAMBLING:

A scramble key will be requested if the data set being updated needs to be unscrambled. The same key will be used to rescrumble the data set, including the updated records.

The scramble key is not permanently stored anywhere by MICRO. The user is responsible for remembering it.

STAND-ALONE PROGRAMS FOR UPDATING DATA SETS:

It is more economical to enter data via a program

that performs updating only, without any of MICRO's retrieval capabilities. The result is the same and the user interactions are very similar. The shortcoming is that no notation of updating is made through the command trace facility.

To update a data set independently in interactive-prompting mode, the user should issue the following MTS command:

```
$RUN ILIR:MICRO.UP
```

To update data sets independently in batch free-format mode, the user should issue this MTS command:

```
$RUN ILIR:MICRO.UP SCARDS=inputfile  
SPUNCH=errorfile PAR=FREEFORM
```

Where the input file and error file correspond to <file name 1> and <file name 2> in the MICRO ENTERDATA command.

In stand-alone mode initial user replies are read from the MTS pseudo-device GUSER. MICRO.UP will prompt (on SERCOM) for the following information:

- (1) MICRO data set name: This may be either the name of an existing data set (as indicated by the absence of a colon (:)) in the fifth column MICRO dictionary file) or the name of an existing MICRO dictionary file (as indicated by a four-character Signon-ID followed by a colon, and the dictionary file name). If no reply is given the program will terminate.
- (2) Signon-ID of another directory or "CANCEL": Requested only if the data set name cannot be found in the directory that was searched or if that directory does not exist or is not accessible. Four-character signon-ID is all that is required. If "CANCEL" is the reply, the user will be asked for a new data set name and the search for it will resume with the user's directory. A reply is required.
- (3) Data file name: Requested only if a dictionary

file name was given instead of a MICRO data set name. The file must contain data, be sequential and be permitted RW (read-write access) to the user. The data must be compatible with the dictionary. A reply is required.

- (4) Scramble key: Requested only if the dictionary indicates that the data set is or should be scrambled. The same key will be used by MICRO.UP to unscramble the data set and to scramble it again, after it has been updated. The user must remember the key because it is not permanently stored anywhere by MICRO. Garbled data is the only indication that the data set has been unscrambled using an incorrect key. A reply is required.
- (5) Names of fields used to identify records: The field names must appear on one line separated by one or more blanks. If no reply is given the program terminates.
- (6) Names of fields to be updated or "STOP": The field names must appear on one line separated by one or more blanks. The user should type "STOP" when all updating has been completed. A reply is required.

- COMMENTS:
- (1) When providing the lists of fields for this command, the user must type all field names (or abbreviations) separated by one or more blanks. The list may be typed on one line or continued to additional lines as described in comment (11). If any of these names are in error, the command will ignore the entire list of fields and the user will be asked again for the field names.
 - (2) If no identifying fields are given, the UPDATE command is terminated and a resulting set (which replaces the permanent set if PERMANENTLY or PERM was specified) is created which is identical to the specified set.
 - (3) An attention interrupt during the printing of the identifying field values which were not found in the specified set has the effect of suppressing the remainder of the printout. The updating process continues, however.

- (4) Lines in external data files are never deleted in the process of updating. When a character string is given as the value of an external field the character string is written into the external data file and the line number (notepointer) replaces the previous line number in the data set. The old line in the file is not deleted since other records in the data set may still refer to that same character string.
- (5) The fields used to identify records may also be updated.
- (6) If the values of the identifying fields identify more than one record then all such records are changed.
- (7) A temporary set cannot be PERMANENTLY updated.
- (8) At any time in update mode, the user may go to MTS command mode by typing MTS beginning in column 1. In batch free-format mode, MTS may be in the input file. The user may return to update mode by typing \$RESTART.
- (9) At any time in update mode, the user may execute an MTS command by typing the command beginning with a dollar sign (\$) in column 1. In batch free-format mode, the MTS command may be in the input file. Returning to update mode (and to MICRO) is dependent upon which MTS command has been executed.
- (10) When the user is faced with replying with YES or NO, any reply beginning with a Y will be interpreted as affirmative and all other replies will be interpreted as negative.
- (11) Long input may be continued on additional lines by placing a minus sign or dash (-) in the last column transmitted before the carriage return. The minus sign will not be included in the input string.
- (12) A category may be specified as the value of a field by entering the category name, its abbreviation or the value associated with that category.

- ```
<TIME OUT.....
<USE "$RESTART" TO CONTINUE IF YOU ARE
 MAKING PROGRESS
```

- (15) The CPU time limit for the UPDATE command from MICRO (or any other MICROC command) defaults to 10 CPU seconds. This may be changed via the SET command.
- (16) When determining which default value to use for a field that has categories, UPDATE does not check to see if more than one category has been designated as the default field value (category type=1). The first type 1 category will be used.
- (17) Values on the same line must not be separated by more than 255 blanks.
- (18) Caution should be exercised when using the UPDATE command in an input stream assigned via the SET INPUT command. Because the UPDATE command reads data on SCARDS, field values for updating should be placed in the specified file or device. The UPDATE command reassigns SCARDS to \*MSOURCE\* at their completion.

**ERROR MESSAGES:** Most of the following error messages for this command are preceded by a phrase which indicates what item or value is incorrect. This phrase begins with the word "concerning" and then indicates the item or value. If the value for a field is incorrect then the name of the field

will be identified also. The next lines tell what is wrong and request the user to supply a replacement or make some other response.

In addition, the error messages from the ENTERDATA command apply to the data entry phase of the UPDATE command.

'DATA FILE MUST EXIST OR BE LOCATED ON THIS SIGNON-ID.'

The data file does not exist and is preceded by a signon-ID other than the current signon-ID. Files may only be created on the current signon-ID. Hence, this file cannot be created.

'DATA FILE NOT COMPATIBLE WITH THE DICTIONARY.'

Records in the data file are shorter than the record length implied in the dictionary. Probably the wrong dictionary is associated with the data file. This condition must be changed.

'ENTER A NEW DATA FILE NAME OR "CANCEL".'

Requested when the data file indicated cannot be used. If the reply is "cancel" the user is asked for a new data set name.

'ENTER A NEW DICTIONARY NAME OR "CANCEL".'

Requested when the indicated dictionary cannot be used. If the reply is "cancel" the user is asked for a new data set name.

'ENTER SCRAMBLE KEY.'

The data set is scrambled and the user must provide (and remember) the character string which is used to scramble and unscramble the data file.

'HUH?'

This message indicates that an attention interrupt is being processed and that previous activity has stopped.

'IMPROPER FORMAT FOR A DATA FILE.'

The contents of the file provided as a MICRO data file are not a MICRO data set.

'NO SPACE AVAILABLE - DATA ENTRY TERMINATED.'

Either a file could not be created because the

current signon-ID is out of disk space or (rarely) sufficient virtual memory space is not available.

'PREVIOUS FIELDS DISCARDED.'

This message is printed during the handling of an attention interrupt to indicate that the names of fields to be updated which were previously entered are now being ignored.

'READ ACCESS NOT ALLOWED.'

The file indicated cannot be read from the current MTS signon-ID.

'READ/WRITE (RW) ACCESS NOT ALLOWED.'

The file indicated cannot be read from and written into from the current signon-ID.

'THIS DIRECTORY DOES NOT CONTAIN THAT DATA SET.'

The directory indicated does not contain an entry for the previously specified data set.

'THIS FIELD IS ALREADY BEING UPDATED.'

A field name appears more than once in the list of fields to be updated.

'THIS FIELD IS ALREADY USED TO IDENTIFY RECORDS.'

A field name appears more than once in the list of fields which identify records to be updated.

'THIS FILE DOES NOT EXIST.'

The file indicated does not exist.

'THIS IS NOT A FIELD NAME OR ABBREVIATION.'

A field name is incorrect or misspelled.

'WARNING: DATA FILE NOT COMPATIBLE WITH DICTIONARY.'

Records in the data file are longer than the record length implied in the dictionary. This may be because the wrong dictionary is associated with the data file. This condition is allowed to continue.

NOTEWORTHY USAGE STRATEGIES:

- (1) This command is best used when the same group of fields is being changed in many records which can



all be identified by the values in the key fields.

- (2) The cost associated with updating records can be reduced by isolating the records to be updated. For a discussion of this technique, see Section 7.3.

#### SPECIAL DESIGN CONSIDERATIONS:

- (1) If at the time of data set definition a field was declared to require data, the user must supply a value for that field. No default value will be used. Otherwise, if no data is provided for a field (by pushing only the carriage return or sending an end-of-file), the default value of the field will be the value of a category declared as the default field value. If there is no such category for the field then zero and blank are the default values for numeric and character fields respectively.
- (2) If at the time of data set definition a field was declared to require a category and if the user provides data for a field, the data entered must be a category name, abbreviation, or the value associated with a category.

304  
UPDATE Command

MICRO Version 5.0  
October 1977

USE

COMMAND DESCRIPTION

PURPOSE: To specify which data set's dictionary is to be used with a subsequent READ command.

PROTOTYPE: USE <data set name>

APPLICABLE COMMAND MODIFIERS: None.

IMPLICIT EFFECT ON THE RESULT SET: None.

EXAMPLE: USE STUDENTS. [A-6]

WHAT HAPPENS: The specified data set name is identified as the last explicitly named data set and thus prepares the way for the next command to be a READ command. See the READ command description.

306  
USE Command

MICRO Version 5.0  
October 1977

## WRITE

### COMMAND DESCRIPTION

**PURPOSE:** To write a MICROC data set on disk storage for future use by either STDS or MIDAS.

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

Where <file name> is the optionally specified location for the data associated with <data set name>.

If <file name> is not specified, the <data set name> will be used for the <file name>.

If <file name> does not exist, it will be created.

If <file name> is not empty, a request to empty it will be made. If denied, the name of another file will be requested.

The data associated with the specified file is written in STDS format but the dictionary information is not written onto the disk file. This form of the data allows the user to use the STDS system directly.

(2) WRITE FOR ANALYSIS <data set name>  
[CN <file name>].

Where <file name> is the optionally specified location for the data associated with <data set name>.

If <file name> is not specified then the <data set name> will be used for the <file name>.

If <file name> does not exist, it will be created. In addition, a second file with a name consisting of <file name> followed by MD (for MIDAS dictionary information) will be created. Therefore <file name> should be no more than 10 characters in length.

The data written for analysis can be accessed in

MIDAS at any time after the current MICRO session by issuing the following MTS command:

\$\$SOURCE <file name>

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

All field names in <data set name> become MIDAS analytical variables which are real, double-precision numbers.

APPLICABLE COMMAND MODIFIERS:       None.

IMPLICIT EFFECT ON RESULT SET:       None.

COMMENTS:       (1) STDS is described in Sections 1.1 and 2.2.  
                  (2) Michigan Interactive Data Analysis System (MIDAS) is a statistical package available on MTS at the University of Michigan. (See Section 1.5 for references to MIDAS documentation.)

EXAMPLE:       WRITE FOR ANALYSIS MIDAS.                               [B-14]

ERROR MESSAGE:   'FAILURE IN DATA SET WRITE OUT: PLEASE TRY AGAIN.'  
                  Usually preceded by a more explicit error message.

NOTEWORTHY USAGE STRATEGY:

The WRITE command may be used instead of the REPLACE command. When the WRITE command is used, no change is made to the core copy of the set referenced by the data file. As a precaution, the RELEASE command or PURGE command should be issued before referencing that set again.

## CHAPTER 10

### THE MACRO PROCESSOR

#### 10.1 INTRODUCTION

The macro subsystem is an extension of the MICRO Command Language and is designed for advanced users. 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 command defined by the macro, is issued each time the user wants to generate the desired sequence of MICRO commands. The macro subsystem allows conditional macro generation; that is, it allows the users to alter the sequence of commands to be generated during an interactive MICRO session.

#### 10.2 MACRO LIBRARIES

All macros must reside in a macro library which is stored in an MTS line file. There are two types of macro libraries. One is automatically associated with a user through master signon-ID (see Section 2.8.1.1) and is stored in CCID:MACLIB where CCID is the master signon-ID. A second, user macro library may be assigned to logical device 0 at run-time. If a macro library is assigned to device 0, it will be searched before the master signon-ID macro library.

Both macro libraries have the same structure and function. A user will automatically have access to the master signon-ID macro library during a MICRO session if that file is permitted to the user.

A user macro library is contained in an MTS line file and consists of three parts: the count, the index and the macro definitions. Line 1 of a macro library file contains the number of macros in the library. This integer number must be left-justified starting in column 1. The index follows line 1, with one line per macro. Each line contains the name of a macro, starting in column 1, and the line number of the first line of this macro, left-justified starting in column 26. The remainder of the file consists of the macro definitions starting at the line numbers identified.

#### 10.3 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; and
- (4) A macro-definition end statement (the % END DEFINITION statement).

Neither a macro definition nor a macro command can appear within a macro definition. A macro definition must be in the macro library before its corresponding macro command is used.

#### 10.4 MACRO COMMAND

Macro commands are commands issued within the main MICRO system. Every time a command is given to the MICRO system, the macro-library index is searched to see if the command is a macro command. If no match is made then the list of regular MICRO commands is searched.

If a macro-command name is misspelled and MICRO prompts for a replacement or "CANCEL", the user must enter "CANCEL" because the search to identify the command name is restricted to regular MICRO commands because the macro libraries are not searched again. 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.

#### 10.5 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 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 a null string as its value. If an at-sign (@) is desired in the text of a MICRO command which is to be generated by a macro, it is necessary to specify two at signs (@@) in the macro model statement.

#### 10.6 COMMENTS IN THE MACRO DEFINITION

A comment in the macro definition must begin with an asterisk (\*). It will not be printed.

#### 10.7 SYNONYMS FOR MACRO COMMANDS

A synonym for a macro command can be generated by placing an extra macro-definition name/prototype statement and delimiter statements in the lines on the primary macro definition. The delimiter statements must be followed by a %GOTO branching to the first model or conditional macro-generation statement of the macro. The extra macro name must be listed in the index along with the primary name.

#### 10.8 WRITING MACRO DESCRIPTIONS/COMMANDS

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

The following macro-subsystem statement descriptions use the same notation as described in Section 9.2.

## 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 definition.
- SYNONYM:** DEF
- PROTOTYPE:** DEFINE <macro name> [<delim 1>] [<param 1>]  
[... [<delim m>] [<param n>]].
- Where <delim m> is a symbolic delimiter of the form: @Dm
- and <param n> is a symbolic parameter of the form: @n
- 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.
- EXAMPLES:**
- DEFINE PUT.
- DEFINE LOCK AT @D1 @1 @2.

DELIMITED 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 [,...<char string ])

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.

EXAMPLES: % END DEFINITION

%CUT END

GOTO

MACRO SUBSYSTEM STATEMENT

PURPOSE: To unconditionally branch ahead 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>.

COMMENT: This is a conditional generation statement.  
Caution: <label 2> must be located after the GOTO 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 further into 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 and the value of <sym var 1> equals @NULL or <char string> or <sym var 2>, then the macro statement labeled by <label 2> is processed next. Otherwise the next macro statement will be processed.

If NE is specified, the condition tested is inequality.

**EXAMPLES:**

% IF @1 EQ PLOT GOTO THERE

%BYTEST IF @L2 NE @NULL GOTO OUT

MACRO COMMAND

MACRO SUBSYSTEM STATEMENT

PURPOSE: To initiate 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 begins generating MICRO commands. The values specified by the macro command replace the corresponding symbolic variables in the definition.

(2) Each MICRO command is processed as it is generated.

(3) Normally the MICRO commands generated are not printed. See Section 10.10.

EXAMPLE: 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 delimiter and/or parameter variables. The MICRO command end is indicated by a single period.
- COMMENTS:**
- (1) The text is inserted into the MICRO command being generated. However, any symbolic delimiters or symbolic parameters are replaced by the actual values in the macro command.
  - (2) If a period (.) is desired in the text of the MICRO command, then two periods (..) must be used in the macro model statement. (See example below.)
  - (3) If an at-sign (@) is desired in the text of the MICRO command, then two at-signs (@@) must be used in the macro model statement. (See example below.)
- EXAMPLES:**
- FIND IN @1 WHERE @2 @D5.
- COMPUTE IN @1 SUCH THAT SALARY = SALARY \* 1..05.
- PRINT@@VERTICAL IN @2 @3,@4,@5.

NOP

MACRO SUBSYSTEM STATEMENT

PURPOSE: To provide a point of reference (a label) for use  
in conditional macro execution.

PROTOTYPE: %[<label>] NCF

COMMENT: The subsystem just passes over NOP statements.

EXAMPLE: %THERE NCF

### 10.9 MACRO EXAMPLE

The following is a sample macro-definition:

```
DEFINE GIVE @D1 @D2 @1 @D3 @2.
%D1 (STATISTICS, RANGE, MOMENTS)
%D2 (FOR,CF)
%D3 (IN)
SELECT IN @2 @1.
CALL TALLYHC
% IF @D1 EQ STATISTICS GOTO THREE
% IF @D1 EQ RANGE GOTC ONE
 PAR=2.
% GOTC END
%THREE NOP
 PAR=3.
% GOTC END
%ONE NOP
 PAR=1.
%END FND 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 TALIHC PAR=1.
```

While forms (2) and (4) would produce:

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

```
CALL TALIHC PAR=3.
```

and form (3) would produce:

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

```
CALL TALIHC PAR=2.
```

#### 10.10 MACRO ECHO

The lines of a macro will be printed as they are read when MACRO ECHO is set on. See the SET command description.

## APPENDIX A

```

#SCOMMENT
#SCOMMENT - THERE ARE SIX DATA ENTRY EXERCISES EXECUTED IN APPENDIX A. THEY ARE LABELED A-1 THROUGH A-6:
#SCOMMENT
#SCOMMENT
#SCOM A-1: PROMPTED FREE-FORM DATA ENTRY (ON-LINE) IN MICRO VIA THE ENTERDATA COMMAND
#SCOM (ON THE FIN-AID DATA SET).
#SCOM A-2: PROMPTED FREE-FORM DATA ENTRY (ON-LINE) VIA THE STAND-ALONE MICRO.ADD PROGRAM
#SCOM (ON THE STUDENTS DATA SET).
#SCOM A-3: BATCH FREE-FORM DATA ENTRY (FROM A FILE) IN MICRO VIA THE ENTERDATA COMMAND
#SCOM (ON THE STUDENTS DATA SET).
#SCOM A-4: BATCH FREE-FORM DATA ENTRY (FROM A FILE) VIA THE STAND-ALONE MICRO.ADD PROGRAM
#SCOM (ON THE FIN-AID DATA SET).
#SCOM A-5: BATCH FIXED-FORMAT DATA ENTRY (FROM A FILE) USING THE STAND-ALONE MICRO.FORM AND
#SCOM MICRO.CNVRT PROGRAMS (ON THE FIN-AID DATA SET).
#SCOM A-6: CUSTOMIZED PROGRAM METHOD OF DATA ENTRY (ON THE STUDENTS DATA SET).
#SCOMMENT
#SCOMMENT
#SCOM COST=ON TDR=ON $1.97T
#T=0.001 DR=0 $0.00,
#SCOMMENT

```

```

#T=0.004 DR=0 $.00, $1.98T
#SCOMMENT - EXERCISE A-1: PROMPTED FREE-FORM DATA ENTRY FROM MICRO VIA THE ENTERDATA COMMAND.
#T=0.003 DR=0 $.00, $1.98T
#RUN ILIR:MICRO
#EXECUTION BEGINS
{STDS/UM:<02,21,77>}
20:13:05
06-17-77
[VM BUFFERS(8-PAGE): 32 1 1]

```

```

*** M I C R O ***
VERSION 5.0 (06-17-77)
20:13:09
JUN 17, 1977

```

\*READY:

```

*READY:
-SET CLOCK ON.
* 0.001 CPU SEC.
* 0.414 ELAPSED SEC.

```

```

*READY:
-SET COST ON.

```

```

* $.07, $2.33T
* 0.003 CPU SEC.
* 1.166 ELAPSED SEC.

```

```

*READY:
-SET TIME=1.

```

```

* $.01, $2.33T
* 0.003 CPU SEC.
* 1.156 ELAPSED SEC.

```

```

★READY:
-ENTERDATA INTO FIN-AID PERMANENTLY.

%CONCERNING: ILIR:FIN-AID
%DATA FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.
%Y
%ENTER SCRAMBLE KEY.
%test
%AFTER HOW MANY RECORDS SHOULD DATA BE SAVED ?
%6
%ABBREVIATED PROMPTING ?
%n

%STUDENT-ID-N0 : 913222340
%YEAR-AWARDED : 74
%TERM-AWARDED : 2
%CLASS-LEVEL : sr
%TYPE-OF-AID : ws
%SOURCE-OF-FUNDS : fgov
%AMOUNT : 350.00
%REMARK* : 'outstanding senior'

%CORRECTIONS:
%
%STUDENT-ID-N0 : 133680736
%YEAR-AWARDED : 74
%TERM-AWARDED : 1
%CLASS-LEVEL : 3
%TYPE-OF-AID : 2
%SOURCE-OF-FUNDS : 3
%AMOUNT : 485
%REMARK* :

%CORRECTIONS:
%
%STUDENT-ID-N0 : 304789983
%YEAR-AWARDED : 75
%TERM-AWARDED : fall
%CLASS-LEVEL : so
%TYPE-OF-AID : 3
%SOURCE-OF-FUNDS : 8

%CONCERNING: 8 FOR FIELD SOURCE-OF-FUNDS
%VALUE MUST BE A CATEGORY NAME OR CATEGORY VALUE.
%NEW VALUE ?
%3
%AMOUNT : 128.00
%REMARK* :

%CORRECTIONS:
%796731703

```

%CONCERNING: 796731703  
%FIELD MUST BE SET OFF BY "=" OR ":" OR "," OR ";".

%CORRECTIONS:  
%  
%STUDENT-ID-N0 : 796731703  
%YEAR-AWARDED : 76  
%TERM-AWARDED : s  
%CLASS-LEVEL : sophomore  
%TYPE-OF-AID : tuition  
%SOURCE-OF-FUNDS : 2  
%AMOUNT : 295.  
%REMARK\* : 'regent's' award'

%CORRECTIONS:  
%

%STUDENT-ID-N0 : 360984224  
%YEAR-AWARDED : 77  
%TERM-AWARDED : winter  
%CLASS-LEVEL : 2  
%TYPE-OF-AID : 2  
%SOURCE-OF-FUNDS : univ  
%AMOUNT : 175.00  
%REMARK\* : 'one time award'

%CORRECTIONS:  
%  
%STUDENT-ID-N0 : 229644658  
%YEAR-AWARDED : 75  
%TERM-AWARDED : 1  
%CLASS-LEVEL : so  
%TYPE-OF-AID : 5  
%SOURCE-OF-FUNDS : univ  
%AMOUNT : 400  
%REMARK\* : /s

%CORRECTIONS:  
%  
% 6 RECORDS ARE BEING ADDED.  
%[KEY] T E S I  
% E3 C5 E2 E3  
%[KEY] T E S I  
% E3 C5 E2 E3  
% 6 RECORDS IN PERMANENT SET.  
%SAVE COMPLETED.  
%STUDENT-ID-N0 : 686850331 74 2  
%CLASS-LEVEL : senior  
%TYPE-OF-AID : rb  
%SOURCE-OF-FUNDS : 1 525 //



```

%CORRECTIONS:
%STUDENT-ID-N0 : 480780530 /6 f fr 3 sgov 375 'governor's award'
%CORRECTIONS:
%STUDENT-ID-N0 : 480780530 76 w fr 3 sgov 3/b /s
%CORRECTIONS:
%STUDENT-ID-N0 : 1399094799 73 f 4 4 3 180 ''
%CORRECTIONS:
%STUDENT-ID-N0 : 142668782 74 f jr tu Industry 1050.00 '
new york'
%CORRECTIONS:
%stop 5 RECORDS ARE BEING ADDED.
%[(KEY] T E S T
% E3 C5 E2 E3
%[(KEY] T E S T
% E3 C5 E2 E3
% 11 RECORDS IN PERMANENT SET.
%MICRO TIME OUT.....
%TYPE *RES " TO CONTINUE
#T=1.839 DR=143 $1.11, $3.09
#res
%SAVE COMPLETED.
* $.80, $3.13T
* 0.560 CPU SEC.
* 252.954 ELAPSED SEC.

*READY:

* this is the end of exercise a-1.

* $.01, $3.15T
* 0.003 CPU SEC.
* 1.334 ELAPSED SEC.

*READY:

stop.
*20:25:23
*JUN 17, 1977
#EXECUTION TERMINATED
#T=0.104 DR=36 $.09, $3.17T

```

from international business systems of armonk,-

```

$COMMENT - EXERCISE A-2: PROMPTED (INTERACTIVE) FREE-FORM DATA ENTRY VIA THE STAND-ALONE MICRO.ADD PROGRAM.
#T=0.004 DR=6 $2.02, $2.87T
#RUN ILIR:MICRO.ADD
#EXECUTION BEGINS

```

```

% %% MICRO DATA ENTRY SYSTEM %%
%21:05:05
%JUN 17, 1977
%*[STDS/JM:<02,21,77>]*
%21:05:07
%06-17-77
%[VM BUFFERS(8-PAGE): 32 1 1]
%[VM BUFFERS(8-PAGE): 32 8 1]
%ENTER MICRO DATA SET NAME.
%students

```

```

%CONCERNING: ILIR:STUDENTS
%DATA FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.

```

```

%Y
%ENTER SCRAMBLE KEY.
%test

```

```

%AFTER HOW MANY RECORDS SHOULD DATA BE SAVED ?
%40

```

```

%ABBREVIATED PROMPTING ?

```

```

%Y
%$IN : 949705539
%NA : white albert 1'#

```

```

%NA : 'white albert 1'
%$RACE : 1
%$SEX : m
%$BDAY : 590911
%$CLYR : 80

```

```

%CORRECTIONS:
%

```

```

%$IN : ros#
%$IN : 8J4351704
%NA : 'rosen larry z'
%$RACE : 1
%$SEX : m
%$BDAY : 581005
%$CLYR : 80

```

## %CORRECTIONS:

%  
 %SIN : 360984224  
 %NA : 'bates leslie ann'  
 %RACE : white  
 %SEX : female  
 %BDAY : 581115  
 %CLYR : 80

## %CORRECTIONS:

%  
 %SIN : 430226647 'furguson william' wh ma 550708 77

## %CORRECTIONS:

%

%SIN : 41837339

%NA : 'wu hua chung'

%RACE : oriental

%SEX : m

%BDAY : 560620

%CLYR : 78

## %CORRECTIONS:

%sin=418373C39

## %CORRECTIONS:

%

%SIN : 498830942 'newar evelyn' l f 550513 //

## %CORRECTIONS:

%

%SIN : 13222340 'richardson deborah' black female 550310 //

## %CORRECTIONS:

%

%SIN : 967397398

%NA : 'mendez jose roberto' spanish male

%BDAY : 560708 78

## %CORRECTIONS:

%

%SIN : 173049236 crosby jane lillian l f 570510 79

%CONCERNING: JANE FOR FIELD RACE

%VALUE MUST BE A CATEGORY NAME OR CATEGORY VALUE.

%NEW VALUE ?

%!

```

%HUH ?
%RECORD ABORTED.
%DO YOU WISH TO ADD MORE DATA ?
%yes
%$IN : 173049236 'crosby Jane lillian' 1 f 570510 79

%CORRECTIONS:
%
%$IN : 336522698 'bishop arthur b' 1 m
%BDAY : 570930 79

%CORRECTIONS:
%stop
% 10 RECORDS ARE BEING ADDED.
%[KEY] T E S T
% E3 C5 E2 E3
%[KEY] T E S T
% E3 C5 E2 E3
% 10 RECORDS IN PERMANENT SET.
%SAVE COMPLETED.
#EXECUTION TERMINATED
#T=1.320 DR=194 $.94, $3.81T
#$COMMENT - THIS IS THE END OF EXERCISE A-2.

```

SCOMMENT - EXERCISE A-3: BATCH (FROM A FILE) FREE-FORM DATA ENTRY FROM MICRO VIA THE ENTERDATA COMMAND.

#T=0.005 DR=7 \$ .02, \$4.06T

#R ILIR:MICRO

#EXECUTION BEGINS

\*ISTDS/UM:<02,21,77>]\*

21:28:09

06-17-77

[VM BUFFERS(8-PAGE): 32 1 1]

\*\*\* M I C R O \*\*\*

VERSION 5.0 (06-17-77)

21:28:13

JUN 17, 1977

\*READY:

-SET COST=ON.

\* \$ .06, \$4.41T

\*READY:

-SET CLOCK=ON.

\* \$ .01, \$4.41T

\* 0.003 CPU SEC.

\* 0.747 ELAPSED SEC.

\*READY:

-ENTERDATA INTO STUDENTS PERMANENTLY FROM STUDENTS-NEW WITH ERRORS ON BADRECORDS.

%ENTER SCRAMBLE KEY.

%TEST

%[KEY] T E S T

% E3 C5 E2 E3

%CONCERNING: 945031192 FOR FIELD ??????????????

%TOO MUCH DATA HAS BEEN PROVIDED FOR THIS RECORD.

%ERROR OCCURRED ON LINE 30.000

%RECORD BEGAN ON LINE 29.000

% 29.000 530254690 'BRINHURST F RICHARD' 1 M 581018 80

% 30.000 945031192 'LIPSCOMB ALVIN D' 1 M 560910 78 /E

% 28 RECORDS ARE BEING ADDED.

%[KEY] T E S T

% E3 C5 E2 E3

%[KEY] T E S T

% E3 C5 E2 E3

% 38 RECORDS IN PERMANENT SET.

%SAVE COMPLETED.

% 29 RECORDS WERE READ IN.

% 1 RECORDS WERE REJECTED.

% 28 RECORDS ARE BEING ADDED.

% 38 RECORDS IN PERMANENT SET.

\* \$40, \$4.81T  
 \* 0.441 CPU SEC.  
 \* 11.844 ELAPSED SEC.

\*READY:  
 -STOP.

\*21:29:54

\*JUN 17, 1977

\*EXECUTION TERMINATED

#T=1.658 DR=73 \$4.80, \$4.86T

#L BADRECORDS

> 29 530254690 'BRINGHURST F RICHARD' 1 M 581018 80

> 30 945031192 'LIPSCOMB ALVIN D' 1 M 560910 78 /E

#END OF FILE

#T=0.020 DR=2 \$0.01, \$4.87T

\$LIST ILIR:STUDENTS-NEW

> 1 391775833 'ARMSTRONG NORA M' 2 F 570611 79 /E  
 > 2 95887328 'POND STUART LESLIE' 1 M 560528 78 /E  
 > 3 733267655 'SMOTHERS MAE S' 1 F 550820 77 /E  
 > 4 805219718 'NEUMANN CURTIS ARTHUR' 1 M 580228 80 /E  
 > 5 610499543 'FALKENZIE PHILIP' 1 M 570321 79 /E  
 > 6 514312603 'DECKER GERTRUDE MARY' 1 F 550706 77 /E  
 > 7 741997403 'SMITH MARY' 1 F 560603 78 /E  
 > 8 552229252 'CARTER ANN' 1 F 501205 79 /E  
 > 9 490697986 'MEYER LINDA SUE' 2 F 580104 80 /E  
 > 10 430150047 'FITZGERALD JOAN' 2 F 581015 80 /E  
 > 11 613835160 'KNAPP FREDERICK HAROLD' 1 M 550118 77 /E  
 > 12 787195848 'MILLER SUSAN ANN' 1 F 570308 79 /E  
 > 13 472342535 'HOGGES THOMAS' 1 M 560419 78 /E  
 > 14 752155359 'ERICKSON SHEILA MARY' 3 F 581010 UNKNOWN /E  
 > 15 236028948 'NELSON RICHARD ELMER' 1 M 551106 77 /E  
 > 16 133680736 'LEE SUE JEAN' 5 F 560716 78 /E  
 > 17 886122443 'STARR ELLEN ELIZABETH' 1 F 481105 77 /E  
 > 18 802810964 'MACK JOSEPH' 1 M 580401 80 /E  
 > 19 921293985 'ARRILLAGA PAUL' 6 M 560225 78 /E  
 > 20 214724324 'LANE ANNE' 1 F 570310 79 /E  
 > 21 796731703 'KENT JERRY' 1 M 570220 79 /E  
 > 22 796202403 'ROGERS LUCILLE A' 1 F 550712 77 /E  
 > 23 907175979 'SCHMIDT RAYMOND F' 1 M 590505 80 /E  
 > 24 403773575 'FIELDS R A' 1 U 570910 79 /E  
 > 25 179410240 'TAYLOR VERA M' 2 F 501004 73 /E  
 > 26 508992890 'CURTIS ELIZABETH ANN' 1 F 551015 77 /E  
 > 27 701385226 'ARNOLD KEITH R' 2 M 560611 78 /E  
 > 28 186549615 'CUNNINGHAM SARAH' 1 F 570905 79 /E  
 > 29 530254690 'BRINGHURST F RICHARD' 1 M 581018 80  
 > 30 945031192 'LIPSCOMB ALVIN D' 1 M 560910 78 /E

#END OF FILE

#T=0.049 DR=14 \$0.06, \$1.89T

#COMMENT - END OF EXERCISE A-3.

BATCH (FROM A FILE) FREE-FORM DATA ENTRY VIA THE STAND-ALONE MICRO.ADD PROGRAM.  
 THE DATA USED IN THIS EXERCISE IS LISTED AT THE END OF THE EXERCISE. IT IS IDENTICAL  
 IN CONTENT TO THE DATA USED IN EXERCISE A-5; HOWEVER, THE FORMAT IS SLIGHTLY DIFFERENT.  
 COMPARE THE ASSOCIATED COSTS!

```

$COMMENT - EXERCISE A-4:
#$COMMENT -
#$COMMENT -
#$COMMENT--
#SSEI COST=ON TDR=ON
#T=0.001 DR=0 $.00, $1.92T
#$COMMENT

```

```

#T=0.005 DR=0 $.01, $1.94T
#R ILIR:MICRO.ADD SCARDS=FA-DATA SPUNCH=-ERRORS PAR=FREEFORM
#EXECUTION BEGINS

```

```

% %% MICRO DATA ENTRY SYSTEM %%
%22:28:35
%JUN 17, 1977
%*[STDS/UM:<02,21,77>]*
%22:28:37
%06-17-77
%(VM BUFFERS(8-PAGE): 32 1 1)
%(VM BUFFERS(8-PAGE): 32 8 1)
%ENTER MICRO DATA SET NAME.
%FIN-AID
%ENTER SCRAMBLE KEY.
%TEST
%(KEY) T E S T
% E3 C5 E2 E3
%

```

```

%CONCERNING: 8 FOR FIELD TYPE-OF-AID
%VALUE MUST BE A CATEGORY NAME OR CATEGORY VALUE.
%ERROR OCCURRED ON LINE 15.000
%RECORD BEGAN ON LINE 15.000
% 15.000 613335160 77 2 4 8 5 175.00 'FORK FOUNDATION GRANT' /E
%

```

%CONCERNING: 7 FOR FIELD TYPE-OF-AID  
 %VALUE MUST BE A CATEGORY NAME OR CATEGORY VALUE.  
 %ERROR OCCURRED ON LINE 19.000  
 %RECORD BEGAN ON LINE 19.000  
 % 19.000 436226647 73 1 4 7 6 210.00 'ANONYMOUS GIFT OF TAX-EXEMPT BONDS TO THE CLASS VALEDICTORIAN' /E  
 % 23 RECORDS ARE BEING ADDED.  
 % [KEY] T E S T  
 % E3 C5 E2 E3  
 % [KEY] T E S T  
 % E3 C5 E2 E3  
 % 34 RECORDS IN PERMANENT SET.  
 % SAVE COMPLETED.  
 % 25 RECORDS WERE READ IN.  
 % 2 RECORDS WERE REJECTED.  
 % 23 RECORDS ARE BEING ADDED.  
 % 34 RECORDS IN PERMANENT SET.  
 % EXECUTION TERMINATED  
 #T=1.290 DR=0 \$ .46, \$2.39T  
 #

## LIST FA-DATA

|      |           |    |   |   |   |   |         |                                                                    |    |
|------|-----------|----|---|---|---|---|---------|--------------------------------------------------------------------|----|
| > 1  | 251760359 | 73 | 1 | 4 | 2 | 1 | 500.00  | //                                                                 | /E |
| > 2  | 741997403 | 74 | 1 | 3 | 1 | 1 | 350.00  | //                                                                 | /E |
| > 3  | 281482860 | 75 | 3 | 3 | 5 | 3 | 400.00  | 'FEDERAL WORK-STUDY WITH THE DEPT. OF LABOR' /E                    |    |
| > 4  | 733267655 | 74 | 2 | 4 | 3 | 2 | 50.00   | 'STATE GRANT TO STUDY AUTOMOBILE INDUSTRY' /E                      |    |
| > 5  | 906922815 | 73 | 1 | 3 | 2 | 2 | 125.50  | //                                                                 | /E |
| > 6  | 123456789 | 75 | 1 | 2 | 1 | 1 | 235.00  | //                                                                 | /E |
| > 7  | 949705539 | 76 | 2 | 1 | 2 | 2 | 280.00  | //                                                                 | /E |
| > 8  | 149514226 | 75 | 2 | 3 | 3 | 4 | 100.00  | 'GRANT FROM THE OILY PETROLEUM CO. TO STUDY OIL SLICKS' /E         |    |
| > 9  | 805219718 | 77 | 2 | 1 | 4 | 1 | 150.00  | //                                                                 | /E |
| > 10 | 521491463 | 76 | 1 | 2 | 6 | 1 | 200.00  | 'PART-TIME EMPLOYMENT GRADING PAPERS' /E                           |    |
| > 11 | 179410240 | 75 | 3 | 3 | 1 | 1 | 540.00  | //                                                                 | /E |
| > 12 | 945031192 | 74 | 2 | 3 | 3 | 4 | 325.00  | 'GRANT FROM BELT TELEPHONE CO. TO STUDY TELEPATHY' /E              |    |
| > 13 | 259964164 | 76 | 1 | 1 | 5 | 2 | 150.50  | 'WORK IN OFFICE OF SECRETARY OF STATE' /E                          |    |
| > 14 | 408823168 | 76 | 1 | 2 | 4 | 1 | 225.00  | //                                                                 | /E |
| > 15 | 613835160 | 77 | 2 | 4 | 8 | 5 | 175.00  | 'FORK FOUNDATION GRANT' /E                                         |    |
| > 16 | 521690114 | 75 | 2 | 3 | 2 | 2 | 200.00  | //                                                                 | /E |
| > 17 | 113591707 | 74 | 1 | 4 | 1 | 1 | 1200.00 | //                                                                 | /E |
| > 18 | 409142875 | 75 | 2 | 3 | 1 | 3 | 550.00  | //                                                                 | /E |
| > 19 | 436226647 | 73 | 1 | 4 | 7 | 6 | 210.00  | 'ANONYMOUS GIFT OF TAX-EXEMPT BONDS TO THE CLASS VALEDICTORIAN' /E |    |
| > 20 | 787195848 | 75 | 1 | 2 | 3 | 1 | 75.00   | //                                                                 | /E |
| > 21 | 854351704 | 76 | 1 | 1 | 2 | 1 | 250.00  | //                                                                 | /E |
| > 22 | 556052771 | 74 | 1 | 3 | 1 | 4 | 150.00  | 'SCHOLARSHIP FROM AMALGAMATED INDUSTRIES, INC.' /E                 |    |
| > 23 | 463898163 | 74 | 3 | 4 | 1 | 1 | 125.00  | //                                                                 | /E |
| > 24 | 498930942 | 73 | 1 | 4 | 4 | 2 | 820.00  | //                                                                 | /E |
| > 25 | 173049236 | 75 | 1 | 2 | 6 | 1 | 100.00  | 'PART-TIME EMPLOYMENT GRADING PAPERS' /E                           |    |



```

#END OF FILE
#T=0.039 DR=0 $.02, $2.41T
#

L -ERRORS
> 15 613835160 77 2 4 8 5 175.00 'FORK FOUNDATION GRANT' /E
> 19 436226647 73 1 4 7 6 210.00 'ANONYMOUS GIFT OF TAX-EXEMPT BONDS TO THE CLASS VALEDICTORIAN' /E
#END OF FILE
#T=0.019 DR=0 $.01, $2.42T
#

$COMMENT - END OF EXERCISE A-4.

```

SCOMMENT - EXERCISE A-5: BATCH FIXED-FORMAT DATA ENTRY FROM A FILE USING THE STAND-ALONE MICRO-FORM AND MICRO.CNVRT  
 #SCOMMENT PPROGRAMS. THE FIXED FORMAT IS DISPLAYED VIA THE SHOW COMMAND BELOW. THE DATA IS LISTED  
 AT THE END OF THE EXERCISE.

#SCOMMENT  
 #SSET IDR=ON COST=ON  
 #I=0.003 DR=0 \$.00, \$4.56T  
 #R ILIR:MICRO.FORM  
 #EXECUTION BEGINS

% %% MICRO DATA ENTRY SYSTEM %%  
 %23:28:57  
 %JUN 28, 1977  
 %ENTER MICRO DATA SET NAME.  
 %FIN-AID  
 %ENTER NAME OF CONVERSION FORMAT FILE.  
 %FIN-AID.FORM

%CONCERNING: ILIR:FIN-AID.FORM  
 %CONVERSION FORMAT FILE DOES NOT EXIST. TYPE "YES" TO HAVE IT CREATED.

%Y

%ENTER THE MAXIMUM TOTAL NUMBER OF COLUMNS ON ALL CARDS.

%1

%ENTER MAXIMUM NUMBER OF CARDS IN ONE RECORD.

%1

%DO YOU WISH TO INCLUDE OR EXCLUDE ONLY CERTAIN RECORDS ? ENTER "E" OR "I" OR RETURN TO IGNORE.

%

%ENTER RECORD SAMPLING STRING OF ZEROS AND ONES OR RETURN TO IGNORE. (O=SKIP I=KEEP)

%

%FIELD STUDENT-ID-NO

%STARTING COL (OR SUBROUTINE FILE) ?

%1

%INPUT TYPE ?

%1

%LENGTH ?

%9

%MISSING DATA VALUE ?

%

%RECODE TYPE ?

%

%FIELD YEAR-AWARDED

%STARTING COL (OR SUBROUTINE FILE) ?

%11 1 2

%MISSING DATA VALUE ?

%

%RECODE TYPE ?

%

%FIELD TERM-AWARDED  
 %STARTING COL (OR SUBROUTINE FILE) ?  
 %14 I 1  
 %MISSING DATA VALUE ?  
 %  
 %RECODE TYPE ?  
 %

%FIELD CLASS-LEVEL  
 %STARTING COL (OR SUBROUTINE FILE) ?  
 %16 I 1  
 %MISSING DATA VALUE ?  
 %  
 %RECODE TYPE ?  
 %

%FIELD TYPE-OF-AID  
 %STARTING COL (OR SUBROUTINE FILE) ?  
 %18 I 1  
 %MISSING DATA VALUE ?  
 %  
 %RECODE TYPE ?  
 %

%FIELD SOURCE-OF-FUNDS  
 %STARTING COL (OR SUBROUTINE FILE) ?  
 %20 I 1  
 %MISSING DATA VALUE ?  
 %  
 %RECODE TYPE ?  
 %

%FIELD AMOUNT  
 %STARTING COL (OR SUBROUTINE FILE) ?  
 %22 I 4  
 %MISSING DATA VALUE ?  
 %  
 %RECODE TYPE ?  
 %

%ALL  
 %NEW VALUE ?  
 %\* 100  
 %FIELD REMARK\*  
 %STARTING COL (OR SUBROUTINE FILE) ?  
 %30  
 %CHARACTER INPUT ASSUMED.  
 %LENGTH ?  
 %61

%THE CONVERSION FORMAT HAS BEEN COMPLETED.  
 %CORRECTIONS:  
 %

SHOW

%NUMBER OF CARDS = 1  
 %COLUMNS PER CARD = 91  
 %TOTAL INPUT RECORD LENGTH = 91

| %FIELD                                       | TYPE | CARD | COLUMN | LENGTH | MISSING-DATA |
|----------------------------------------------|------|------|--------|--------|--------------|
| %STUDENT-ID-NO                               | I    | 1    | 1      | 9      | 0            |
| %YEAR-AWARDED                                | I    | 1    | 11     | 2      | 0            |
| %TERM-AWARDED                                | I    | 1    | 14     | 1      | 0            |
| %CLASS-LEVEL                                 | I    | 1    | 16     | 1      | 0            |
| %TYPE-OF-AID                                 | I    | 1    | 18     | 1      | 0            |
| %SOURCE-OF-FUNDS                             | I    | 1    | 20     | 1      | 0            |
| %AMOUNT                                      | I    | 1    | 22     | 4      | 0            |
| %RECORDS                                     |      |      |        |        | 100          |
| % 1 -2147483648 THROUGH 2147483647 BECOMES * |      |      |        |        |              |
| %REMARK*                                     | C    | 1    | 30     | 61     | 0            |
| %CORRECTIONS:                                |      |      |        |        |              |
| %STOP                                        |      |      |        |        |              |
| %EXECUTION TERMINATED                        |      |      |        |        |              |
| %I=0.754 DR=32 S.32, \$4.87T                 |      |      |        |        |              |
| %                                            |      |      |        |        |              |

#SCOMMENT - EXERCISE A-6: CUSTOMIZED PROGRAM METHOD OF DATA ENTRY (FOR ADVANCED USERS). THE FORMAT OF THE DATA IS:

| COLUMN | FIELD NAME         |
|--------|--------------------|
| 1      | STATUS (NOT USED)  |
| 2-10   | STUDENT-ID-NO      |
| 11-34  | NAME               |
| 35-36  | STATE              |
| 37-42  | CITIZEN (NOT USED) |
| 43     | SEX                |
| 44     | ETHNICITY          |
| 46-47  | CLASS-YEAR         |

THE LISTING OF THE DATA APPEARS AT THE END OF THE EXERCISE. THE PROGRAM LISTING FOR THIS CUSTOM PROGRAM APPEARS IN APPENDIX C.

#T=0.003 DR=0 \$5.20T

#SCOMMENT

#T=0.005 DR=0 \$5.20T

#R CONVERTOBJ 5=ILIR:STUDENTS-OLD 6=T-STUDENTS 7=-BAD

#EXECUTION BEGINS 47 RECORDS PROCESSED/ 3 RECORDS REJECTED

#EXECUTION TERMINATED

#T=0.147 DR=0 \$5.24T

#SCOMMENT

#T=0.006 DR=2 \$5.00, \$2.51T

#SCOMMENT - NOW RUN MICRO TO READ CONVERTED DATA (IN T-STUDENTS) AND COMBINE IT WITH STUDENTS.

#T=0.005 DR=10 \$5.01, \$2.52T

#R ILIR:MICRO

#EXECUTION BEGINS

\*[STDS/UM:<02,21,77>]\*

23:06:43

06-28-77

[VM BUFFERS(8-PAGE): 32 1 1]

\*\*\* M I C R O \*\*\*  
 VERSION 5.0 (06-17-77)  
 23:06:52  
 JUN 28, 1977

\*READY:  
 -SET COST=ON.

\* \$ .07, \$2.87T

\*READY:  
 -SET CLOCK ON.

\* \$ .00, \$2.87T  
 \* 0.003 CPU SEC.  
 \* 0.749 ELAPSED SEC.

\*READY:  
 -USE STUDENTS  
 +.

\* \$ .01, \$2.89T  
 \* 0.023 CPU SEC.  
 \* 0.767 ELAPSED SEC.

\*READY:  
 -READ ARRAY FROM T-STUDENTS.  
 \*\*\* NO TAU ORDERING SPECIFIED! \*\*  
 \* 47 RECORDS IN RESULT SET

\* \$ .13, \$3.01T  
 \* 0.056 CPU SEC.  
 \* 2.011 ELAPSED SEC.

\*READY:  
 -COMBINE IT WITH STUDENTS. " IS TO BE UNSCRAMBLED. PLEASE ENTER KEY.  
 \*DATA SET "STUDENTS  
 ?TEST  
 \*[(KEY) T E S I  
 \* E3 C5 E2 E3  
 \* 85 RECORDS IN RESULT SET

\* \$.05, \$3.06T  
 \* 0.107 CPU SEC.  
 \* 6.823 ELAPSED SEC.

\*READY:  
 -REPLACE STUDENTS WITH IT. " IS TO BE SCRAMBLED. PLEASE ENTER KEY.

\*DATA SET "RESULT  
 ?TEST  
 \*{KEY] T E S T  
 \* E3 C5 E2 E3

\* \$.16, \$3.22T  
 \* 0.377 CPU SEC.  
 \* 14.264 ELAPSED SEC.

\*READY:  
 -STOP.  
 \*23:08:35  
 \*JUN 28, 1977  
 #EXECUTION TERMINATED  
 #T=1.927 DR=182 \$.73, \$3.25T  
 #

| LIST STUDENTS-OLD |    |                                 |
|-------------------|----|---------------------------------|
| >                 | 1  | 0806746481HARDY COLEEN          |
| >                 | 2  | 233272752FELDMAN MELVIN SAMUEL  |
| >                 | 3  | 1149514225SCOTT MARY GRACE      |
| >                 | 4  | 0755039207FUNK ELMER J          |
| >                 | 5  | 1795419523LEACH KAREN LEE       |
| >                 | 6  | 1221835343WANDELLA JOANNE D     |
| >                 | 7  | 1142668792COKK CYNTHIA ANNA     |
| >                 | 8  | 0181703558HALE ROBERT SAMUEL    |
| >                 | 9  | 2707973903SANCHEZ MARIA         |
| >                 | 10 | 1304789983BORG ELIZABETH M      |
| >                 | 11 | 0499446234ALTERMAN JAMES        |
| >                 | 12 | 1540134327SEGNALLA LYNN MARIE   |
| >                 | 13 | 1200914081SCHILLER ESTHER       |
| >                 | 14 | 2724416980BLUM JOSEPH WILLIAM   |
| >                 | 15 | 2463898153BORCHINSKY ETHEL      |
| >                 | 16 | 0806182554NEWTON ISADOR         |
| >                 | 17 | 0866722832DAWSON EMILY E        |
| >                 | 18 | 1886124715REDPATH JOHN          |
| >                 | 19 | 2234567890WILTON THORNTON       |
| >                 | 20 | 0878948465HAYES HAZEL T         |
| >                 | 21 | 253101049MANN LAURA D           |
| >                 | 22 | 1521690114SMITH HENRY ALBERT    |
| >                 | 23 | 1243155995KLEIN BONNIE          |
| >                 | 24 | 0556052771WARD JULIA CAROL      |
| >                 | 25 | 0770441422KING THEODORE DAVID   |
| >                 | 26 | 1628249652NUCILLI ARTHUR        |
| >                 | 27 | 2555180626BENSON SHELIA JOAN    |
| >                 | 28 | 0409142975FORINO JOSEPH LOUIS   |
| >                 | 29 | 0207920096CONNERS SUZANNE L     |
| >                 | 30 | 0514215872SWIFT SALLY JANE      |
| >                 | 31 | 0264872649FOLEY DONALD C        |
| >                 | 32 | 0251760359JONES DOROTHY         |
| >                 | 33 | 2540744330HUGHES DAVID CRAIG    |
| >                 | 34 | 1686603907BLOCK CYRUS           |
| >                 | 35 | 1259964164WONG DAVID LEE        |
| >                 | 36 | 2210999051HUFFMAN DIANE CAROL   |
| >                 | 37 | 0432523832LYNCH JOHN PHILIP     |
| >                 | 38 | 0229644658MENKE SUSAN A         |
| >                 | 39 | 1113591707JONES WILLIAM         |
| >                 | 40 | 1498966992SMITHIE JEAN DIANE    |
| >                 | 41 | 1231482860CROWFOOT WILLIAM      |
| >                 | 42 | 0199472144ALMONDIZER KAREN S    |
| >                 | 43 | 0122045525GRACE KENNETH R       |
| >                 | 44 | 2816115339GOULD HARRY EDWARD    |
| >                 | 45 | 1636850331BRUZA LESLIE          |
| >                 | 46 | 0614627931SKRZYNSKI VINCENT P   |
| >                 | 47 | 2754292583KUESTER ROBERT E      |
| >                 | 48 | 2521491463HERNANDEZ RAFAEL JOSE |
| >                 | 49 | 1906922815FOSTER HAROLD PETER   |
| >                 | 50 | 1139904799BAILEY EDITH          |
| >                 |    | #END OF FILE                    |
| >                 |    | #T=0.055 DR=5 S.04, \$3.28T     |
| >                 |    | #                               |

AL0102551F177  
MI1021571M179  
IL1230561F173  
IN1108581M 80  
WI1020572F279  
OH0811581F180  
PA0908561F273  
NY1011551M177  
CA1201561F478  
CT0921571F179  
NJ0108581M180  
IL0620552F177  
WI0728571F179  
MI0501561M173  
MI0123551F177  
NY09185812180  
NH0608521F273  
ID0808571M379  
MN1013571M179  
CT1111582F180  
ME1006581F280  
MI1209561 173  
AL1113571F179  
NY1212561F173  
MI0807551M277  
NY0604551M177  
CA0803581F180  
NJ0907561M173  
OH1013571F179  
KY1130581F280  
MI0210571M279  
IN0314551F177  
MI0111561M173  
MI1011571M179  
OH1120582M580  
NY0908561F173  
MI1011571M179  
CA1218571F179  
UT1010551M177  
WV1208561F173  
MI0102561M373  
CT0215581F180  
VA0418581M180  
PA0313561M273  
CA0511552F477  
MI0730551M177  
MI0812581M180  
MN0111471M479  
MI1130561M673  
NY0913571F277



IN1108581M 80  
NY0919581?180  
MI1209561 179

L -BAD  
> 1 0755039207FUNK ELMER J  
> 2 0806182554NEWTON ISADOR  
> 3 1521690114SWITH HENRY ALBERT  
#END OF FILE  
#T=0.022 DR=4 \$ .01, \$3.29T  
#SCOMMENT - END OF EXERCISE A-6



## APPENDIX B - EXECUTED EXERCISES

#SCOMMENT - THIS EXERCISE IS BEING EXECUTED ON A WESTERN UNION EDIT 1200 TERMINAL WITH A 120 COLUMN CARTRIDGE.  
 #SCOMMENT - THIS SERIES OF EXERCISES WAS EXECUTED AT LOW PRIORITY.

#SET TDR=ON COST=ON \$00, \$0.68T  
 #T=0.002 DR=2  
 #RUN ILIR:MICRO

#EXECUTION BEGINS

\*[STD5/UW:<02,21,77>]\*

22:15:58

06-29-77

IVM BUFFERS(8-PAGE): 32 1 11

\*\*\* MICRO \*\*\*

VERSION 5.0 (06-17-77)

22:16:02

JUN 29, 1977

\*READY:

\*READY:  
 -\* THE PREVIOUS COMMAND SET INPUT ECHO ON BUT THE COMMAND WAS NOT ECHOED.

\*READY:  
 -SET TIME=1.

\*READY:  
 -SET COST ON.

\* \$03, \$1.04T

\*READY:  
 -SET CLOCK ON.

\* \$00, \$1.05T

\* 0.003 CPU SEC.

\* 1.182 ELAPSED SEC.

\*READY:  
 -\*

## EXERCISE B-1 - USING MICRO'S SELF-DOCUMENTING CAPABILITIES.

\* \$ .00, \$1.05T  
 \* 0.003 CPU SEC.  
 \* 3.983 ELAPSED SEC.

\*READY\*  
 -LIST COMMANDS.

| *COMMAND NAME  | MINIMUM ABBREVIATION |
|----------------|----------------------|
| **             | *                    |
| *ALTER         | A                    |
| *CALL          | CA                   |
| *CHANGE        | C                    |
| *COMBINE       | CO                   |
| *COMMENT       | COMM                 |
| *COMPUTE       | COMP                 |
| *CROSSTABULATE | CR                   |
| *DATESTAMP     | DA                   |
| *DESCRIBE      | DE                   |
| *DESTROY       | DEST                 |
| *DISPLAY       | D                    |
| *DOCUMENT      | DO                   |
| *END           | END                  |
| *ENTERDATA     | E                    |
| *EXPLAIN       | EX                   |
| *FAM           | FA                   |
| *FDOC          | FD                   |
| *FIND          | F                    |
| *FULLDOC       | FU                   |
| *GET           | G                    |
| *JOIN          | J                    |
| *KEY           | K                    |
| *LIST          | L                    |
| *MESSAGE       | ME                   |
| *MIS           | M                    |
| *NAME          | N                    |
| *NUMBER        | NU                   |
| *ORDER         | O                    |
| *PLOT          | PL                   |
| *PRINT         | P                    |
| *PURGE         | PU                   |
| *R             | R                    |
| *RAJ           | RA                   |
| *RAM           | RA                   |
| *RE            | RE                   |
| *READ          | REA                  |
| *RELEASE       | REL                  |
| *REMOVE        | REM                  |
| *RENAME        | REN                  |
| *REPLACE       | REP                  |

```

*RESET
*RESTRICT
*RESTRICTANDJOIN
*SAVE
*SELECT
*SET
*SIGNOFF
*SORT
*STOP
*SYSTEM
*TABULATE
*IDOC
*TECHDOC
*TURN
*UPDATE
*USE
*WRITE
*XIAB
*
* $.03, $1.09T
* 0.053 CPU SEC.
* 32.265 ELAPSED SEC.

```

```

*READY:
-LIST ALL DATA SETS.

```

```

*DATA SET NAME STATUS
*CARS DISK
*ART DISK
*FIN-AID DISK
*STUDENTS DISK
*
* $.01, $1.09T
* 0.008 CPU SEC.
* 3.941 ELAPSED SEC.

```

```

*READY:
-GET K59L.

```

```

* $.01, $1.10T
* 0.024 CPU SEC.
* 1.674 ELAPSED SEC.

```

```

*READY:
-* NOTICE THE ADDED DATA SET FROM THE OTHER USER SIGNON-ID.

```

```

* $.00, $1.10T
* 0.003 CPU SEC.
* 1.542 ELAPSED SEC.

```

\*READY:  
-LIST ALL DATA SETS.

\*DATA SET NAME STATUS  
\*CARS DISK  
\*ART DISK  
\*FIN-AID DISK  
\*STUDENTS DISK  
\*PUBLIC.DATA DISK

\* S.01, \$1.11T  
\* 0.008 CPU SEC.  
\* 4.427 ELAPSED SEC.

\*READY:  
-DESCRIBE ALL SETS.

\*DATA SET : CARS  
\*EXAMPLE DATA SET OF USED CAR INVENTORY

\*DATA SET : ART  
\*SAMPLE DATA SET DESCRIBING ITEMS IN AN ART MUSEUM

\*DATA SET : FIN-AID  
\*DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL:DEALS WITH FINANCIAL AID INFORMATION. CREATED BY MIKE KAHN. 1  
\*0-31-76/ILIR

\*DATA SET : STUDENTS  
\*DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL: DEALS WITH DEMOGRAPHIC DATA ON STUDENTS. CREATED BY MIKE KA  
\*HN. 10-31-76/ILIR

\*DATA SET : PUBLIC.DATA  
\*PUBLIC DATA SET FOR ANYONE TO USE. CREATED BY MIKE KAHN FOR THE MICRO MANUAL. 04-25-77/K59L

\* S.11, \$1.22T  
\* 0.278 CPU SEC.  
\* 13.448 ELAPSED SEC.

\*READY:  
-DESCRIBE STUDENTS.  
\*DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL: DEALS WITH DEMOGRAPHIC DATA ON STUDENTS. CREATED BY MIKE KA  
\*HN. 10-31-76/ILIR

\* S.02, \$1.24T  
\* 0.049 CPU SEC.  
\* 3.199 ELAPSED SEC.

\*READY:  
-DESCRIBE IN STUDENTS BIRTHDATE.  
\*DATE OF STUDENT'S BIRTH IN YYYYMMDD FORMAT

\* S.01, \$1.25T  
\* 0.021 CPU SEC.  
\* 1.965 ELAPSED SEC.

\*READY:  
-DOCUMENT STUDENTS.

\*FOR: STUDENTS  
\*DICTIONARY LOCATION: ILIR:STUDENTS#  
\*DATE: JUN 29, 1977

\*THIS DATASET CANNOT BE DESTROYED.  
\*THIS DATASET CAN BE REPLACED.  
\*THIS DATASET IS SCRAMBLED.  
\*USE COUNT: 1

\*DATA SET DESCRIPTION:  
\*DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL: DEALS WITH  
\*DEMOGRAPHIC DATA ON STUDENTS. CREATED BY MIKE KAHN. 10-31-76/ILIR

| *F(1)             | *F(2)             | *F(3)             | *F(4)             |
|-------------------|-------------------|-------------------|-------------------|
| STUDENT-ID-NO     | NAME              | ETHNICITY         | SEX               |
| SIN               | NAME              | ETHNICITY         | SEX               |
| REQUIRED          | REQUIRED          | REQUIRED          | REQUIRED          |
| 0                 | 0                 | 0                 | M                 |
| 1                 | 1                 | 1                 | F                 |
| 2                 | 2                 | 2                 | U                 |
| 3                 | 3                 | 3                 |                   |
| 4                 | 4                 | 4                 |                   |
| 5                 | 5                 | 5                 |                   |
| 6                 | 6                 | 6                 |                   |
| UNKNOWN           | UNKNOWN           | UNKNOWN           | MALE              |
| WHITE             | WHITE             | WHITE             | FEMALE            |
| BLACK             | BLACK             | BLACK             | UNKNOWN           |
| AMER-INDIAN       | AMERICAN INDIAN   | AMERICAN INDIAN   |                   |
| SPANISH           | SPANISH           | SPANISH           |                   |
| ORIENTAL          | ORIENTAL          | ORIENTAL          |                   |
| OTHER             | OTHER             | OTHER             |                   |
| CATEGORIES (ONLY) | CATEGORIES (ONLY) | CATEGORIES (ONLY) | CATEGORIES (ONLY) |
| MALE              | MALE              | MALE              | MALE              |
| FEMALE            | FEMALE            | FEMALE            | FEMALE            |
| UNKNOWN           | UNKNOWN           | UNKNOWN           | UNKNOWN           |

\*F(5) BIRTHDATE BDAY REQUIRED DATE OF STUDENT'S BIRTH IN YYMMDD  
 \* CATEGORIES MISSING 0 DATE OF BIRTH IS MISSING  
 \* CLASS-YEAR CLYR CLASS YEAR IN YY FORMAT  
 \* CATEGORIES UNKNOWN -1 UNKNOWN  
 \* \$ .04, \$1.29T  
 \* 0.076 CPU SEC.  
 \* 32.349 ELAPSED SEC.

\*READY:  
 -DOC IN FIN-AID TYPE.

| *F(1) | TYPE-OF-AID CATEGORIES (ONLY) | TYPE | REQUIRED | VALUE | DESCRIPTION        |
|-------|-------------------------------|------|----------|-------|--------------------|
| *     | TUITION                       | TU   | 1        |       | TUITION            |
| *     | ROOM-BOARD                    | RB   | 2        |       | ROOM AND BOARD     |
| *     | CASH                          | CA   | 3        |       | CASH               |
| *     | LOAN                          | LO   | 4        |       | LOAN               |
| *     | WORK-STUDY                    | WS   | 5        |       | WORK STUDY PROGRAM |
| *     | OTHER                         | OTH  | 6        |       | OTHER              |
| *     | \$ .03, \$1.32T               |      |          |       |                    |
| *     | 0.057 CPU SEC.                |      |          |       |                    |
| *     | 10.957 ELAPSED SEC.           |      |          |       |                    |

\*READY:  
 -DOC IN STUDENTS \*

| *F(1) | STUDENT-ID-NO | SIN  | REQUIRED                                                                | VALUE | DESCRIPTION                                 |
|-------|---------------|------|-------------------------------------------------------------------------|-------|---------------------------------------------|
| *     | NAME          | NA   | REQUIRED <td></td> <td>NAME OF STUDENT IN LAST FIRST MIDDLE FORMAT</td> |       | NAME OF STUDENT IN LAST FIRST MIDDLE FORMAT |
| *     | ETHNICITY     | RACE | REQUIRED <td></td> <td>ETHNICITY</td>                                   |       | ETHNICITY                                   |
| *     | SEX           | SEX  | REQUIRED <td></td> <td>SEX OF STUDENT</td>                              |       | SEX OF STUDENT                              |



\*F(5) BIRTHDATE BDAY REQUIRED DATE OF STUDENT'S BIRTH IN YYYYMMDD  
\*

\*F(6) CLASS-YEAR CLYR CLASS YEAR IN YY FORMAT

\* \$ .02, \$1.34T  
\* 0.035 CPU SEC.  
\* 13.170 ELAPSED SEC.

\*READY:  
-DOC IN FIN-AID SOF UNIV.

| CATEGORIES | ABBR | VALUE | DESCRIPTION |
|------------|------|-------|-------------|
| UNIVERSITY | UNIV | 1     | UNIVERSITY  |

\* \$ .01, \$1.35T  
\* 0.024 CPU SEC.  
\* 3.066 ELAPSED SEC.

\*READY:  
-FULLDOC FIN-AID.

\*FOR: FIN-AID  
\*DICTIONARY LOCATION: ILIR:FIN-AID#  
\*DATE: JUN 29, 1977

\*THIS DATASET CANNOT BE DESTROYED.  
\*THIS DATASET CAN BE REPLACED.  
\*THIS DATASET IS SCRAMBLED.  
\*USE COUNT:  
1

\*DATA SET DESCRIPTION:  
\*DEMONSTRATION DATA SET TO ACCOMPANY NEW MICRO MANUAL:DEALS WITH FINANCIAL  
\*AID INFORMATION. CREATED BY MIKE KAHN. 10-31-76/ILIR

| *F(1) | FIELD NAME        | ABBR  | VALUE    | DESCRIPTION                                        | TYPE LENGTH | DISP SCALE | FACTOR |
|-------|-------------------|-------|----------|----------------------------------------------------|-------------|------------|--------|
| *F(1) | STUDENT-ID-NO     | SIN   | REQUIRED | STUDENT IDENTIFICATION NUMBER                      | U           | 4          | 0      |
| *F(2) | YEAR-AWARDED      | YEAR  | REQUIRED | YEAR THAT FINANCIAL AID WAS AWARDED (IN YY FORMAT) | U           | 1          | 4      |
| *F(3) | TERM-AWARDED      | TERM  | REQUIRED | TERM IN WHICH FINANCIAL AID WAS AWARDED            | UC          | 1          | 5      |
|       | CATEGORIES (ONLY) |       |          |                                                    |             |            |        |
|       | FALL              | F     | 1        | FALL TERM                                          |             |            |        |
|       | WINTER            | W     | 2        | WINTER TERM                                        |             |            |        |
|       | SUMMER            | S     | 3        | SUMMER TERM                                        |             |            |        |
| *F(4) | CLASS-LEVEL       | CL    | REQUIRED | CLASS-LEVEL                                        | UC          | 1          | 6      |
|       | CATEGORIES (ONLY) |       |          |                                                    |             |            |        |
|       | FRESHMAN          | FR    | 1        | FRESHMAN                                           |             |            |        |
|       | SOPHOMORE         | SO    | 2        | SOPHOMORE                                          |             |            |        |
|       | JUNIOR            | JR    | 3        | JUNIOR                                             |             |            |        |
|       | SENIOR            | SR    | 4        | SENIOR                                             |             |            |        |
| *F(5) | TYPE-OF-AID       | TYPE  | REQUIRED | TYPE-OF-AID                                        | UC          | 1          | 7      |
|       | CATEGORIES (ONLY) |       |          |                                                    |             |            |        |
|       | TUITION           | TU    | 1        | TUITION                                            |             |            |        |
|       | ROOM-BOARD        | RB    | 2        | ROOM AND BOARD                                     |             |            |        |
|       | CASH              | CA    | 3        | CASH                                               |             |            |        |
|       | LOAN              | LO    | 4        | LOAN                                               |             |            |        |
|       | WORK-STUDY        | WS    | 5        | WORK STUDY PROGRAM                                 |             |            |        |
|       | OTHER             | OTH   | 6        | OTHER                                              |             |            |        |
| *F(6) | SOURCE-OF-FUNDS   | SOF   | REQUIRED | SOURCE OF FUNDS OF AWARDED FINANCIAL AID           | UC          | 1          | 8      |
|       | CATEGORIES (ONLY) |       |          |                                                    |             |            |        |
|       | UNIVERSITY        | UNIV  | 1        | UNIVERSITY                                         |             |            |        |
|       | STATE-GOV         | SGOV  | 2        | STATE GOVERNMENT                                   |             |            |        |
|       | FEDERAL-GOV       | FGOV  | 3        | FEDERAL GOVERNMENT                                 |             |            |        |
|       | INDUSTRY          | IND   | 4        | INDUSTRY                                           |             |            |        |
|       | FOUNDATION        | FOUND | 5        | FOUNDATION                                         |             |            |        |
|       | OTHER             | OTH   | 6        | OTHER SOURCE                                       |             |            |        |
| *F(7) | AMOUNT            | AMT   | REQUIRED | AMOUNT OF FINANCIAL AID AWARDED                    | S           | 4          | 9      |
|       |                   |       |          |                                                    |             |            | 100    |

\*FOR: FIN-AID

| *F(1) | FIELD NAME | ABBR | VALUE | DESCRIPTION | TYPE LENGTH | DISP SCALE | FACTOR |
|-------|------------|------|-------|-------------|-------------|------------|--------|
| *F(1) | FIELD NAME | ABBR | VALUE | DESCRIPTION | TYPE LENGTH | DISP SCALE | FACTOR |
| *     |            |      |       |             |             |            |        |
| *     |            |      |       |             |             |            |        |

E 4 13

## ADDITIONAL COMMENTS (OPTIONAL)

REM\*

\*F(9) REMARK\*

FILE NAME

ILIR:REMARKS\*

\* \$ .05, \$1.41T  
 \* 0.095 CPU SEC.  
 \* 51.368 ELAPSED SEC.

\*READY:  
 \*-EXPLAIN COMMAND EXPLAIN.

## EXPLAIN

## COMMAND DESCRIPTION

PURPOSE: To print an explanation of the specified MICRO command.

## PROTOTYPE AND DESCRIPTION:

EXPLAIN [COMMAND] &lt;command name&gt;.

where COMMAND is an optional phrase added for clarity and <command name> can be any valid MICRO command or any of its synonyms.

COMMENTS: (1) For a complete listing of the latest version of the Technical Reference Manual

\$COPY ILIR:TECHREFMAN TO \*PRINT\*

where \*PRINT\* has been set for a TN printer, or purchase a copy of it from the Institute.

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

\* \$ .08, \$1.49T  
 \* 0.194 CPU SEC.  
 \* 19.519 ELAPSED SEC.

\*READY:  
 \*-DISPLAY ALL.  
 \*22:20:00  
 \*JUN 29, 1977  
 \*VM SIZE = 83 PAGES

\* \$ .01, \$1.49T  
\* \$ .00, \$1.49T  
\* 0.008 CPU SEC.  
\* 3.154 ELAPSED SEC.

\*READY:  
-\* END EXERCISE B-1.

\* \$ .00, \$1.50T  
\* 0.003 CPU SEC.  
\* 1.227 ELAPSED SEC.

\*READY:  
-\*

## EXERCISE B-2 - EXHIBIT SOME OF MICRO'S FACILITIES FOR PRINTING A DATA SET.

\* \$ .00, \$1.50I  
 \* 0.003 CPU SEC.  
 \* 4.112 ELAPSED SEC.

\*READY:

\*-PRINT STUDENTS.

\*DATA SET "STUDENTS

\*test

\*[KEY] T E S T

\* E3 C5 E2 E3

" IS TO BE UNSCRAMBLED. PLEASE ENTER KEY.

| * STUDENT-ID-NO | * NAME                | * ETHNICITY | * SEX   | * BIRTHDATE | * CLASS-YEAR |
|-----------------|-----------------------|-------------|---------|-------------|--------------|
| 13222340        | RICHARDSON DEBORAH    | BLACK       | FEMALE  | 550310      | 77           |
| 95887328        | POND STUART LESLIE    | WHITE       | MALE    | 560528      | 78           |
| 113591707       | JOYES WILLIAM         | WHITE       | MALE    | 551010      | 77           |
| 122045525       | GRACE KENNETH R       | WHITE       | MALE    | 580418      | 80           |
| 133680735       | LEE SUE JEAN          | ORIENTAL    | FEMALE  | 560716      | 78           |
| 139904799       | BAILEY EDITH          | BLACK       | FEMALE  | 570913      | 77           |
| 142668782       | COOK CYNTHIA ANNA     | BLACK       | FEMALE  | 560908      | 78           |
| 149514225       | SCOTT MARY GRACE      | WHITE       | FEMALE  | 561230      | 78           |
| 173049236       | CROSBY JANE LILLIAN   | WHITE       | FEMALE  | 570510      | 79           |
| 179410240       | TAYLOR VERA W         | BLACK       | FEMALE  | 501004      | 78           |
| 181703553       | HALE ROBERT SAMUEL    | WHITE       | MALE    | 551011      | 77           |
| 186549615       | CUNNINGHAM SARAH      | WHITE       | FEMALE  | 570805      | 79           |
| 199472144       | ALMONDIZER KAREN S    | WHITE       | FEMALE  | 580215      | 80           |
| 200914081       | SCHILLER ESTHER       | WHITE       | FEMALE  | 570728      | 79           |
| 207920096       | CONNERS SUZANNE L     | WHITE       | FEMALE  | 571013      | 79           |
| 210999051       | HUFFMAN DIANE CAROL   | WHITE       | FEMALE  | 560908      | 78           |
| 214724324       | LANE ANNE             | WHITE       | FEMALE  | 570310      | 79           |
| 221885343       | MANDELLA JOANNE D     | WHITE       | FEMALE  | 580911      | 80           |
| 229644658       | WENKE SUSAN A         | WHITE       | FEMALE  | 571218      | 79           |
| 234567890       | WILTON THORNTON       | WHITE       | MALE    | 571013      | 79           |
| 236028948       | NELSON RICHARD ELMER  | WHITE       | MALE    | 551106      | 77           |
| 243155995       | KLEIN BONNIE          | WHITE       | FEMALE  | 571113      | 79           |
| 251760359       | JONES DOROTHY         | WHITE       | FEMALE  | 550314      | 77           |
| 259964154       | WONG DAVID LEE        | ORIENTAL    | MALE    | 581120      | 80           |
| 264872649       | FOLEY DONALD C        | BLACK       | MALE    | 570210      | 79           |
| 281482860       | CROWFOOT WILLIAM      | AMER-INDIAN | MALE    | 560102      | 78           |
| 304789983       | BORG ELIZABETH W      | WHITE       | FEMALE  | 570921      | 79           |
| 333272752       | FELDMAN MELVIN SAMUEL | WHITE       | MALE    | 571021      | 79           |
| 336522668       | BISHOP ARTHUR B       | WHITE       | MALE    | 570930      | 79           |
| 360984224       | BATES LESLIE ANN      | WHITE       | FEMALE  | 581115      | 80           |
| 391775933       | ARMSTRONG NORA M      | BLACK       | FEMALE  | 570611      | 79           |
| 403773575       | FIELDS R A            | WHITE       | UNKNOWN | 570910      | 79           |
| 409142875       | FORINO JOSEPH LOUIS   | WHITE       | MALE    | 560907      | 78           |
| 419373039       | WU HUA CHUNG          | ORIENTAL    | MALE    | 560620      | 78           |
| 430150047       | FITZGERALD JOAN       | BLACK       | FEMALE  | 541015      | 80           |
| 432523932       | LYNCH JOHN PHILIP     | WHITE       | MALE    | 571011      | 79           |

|           |                        |             |        |        |      |
|-----------|------------------------|-------------|--------|--------|------|
| 435226647 | FURGUSON WILLIAM       | WHITE       | MALE   | 550708 | 77   |
| 453893163 | BORCHINSKY ETHEL       | WHITE       | FEMALE | 550123 | 77   |
| 472342535 | HODGES THOMAS          | WHITE       | MALE   | 560419 | 78   |
| 490697986 | MEYER LINDA SUE        | BLACK       | FEMALE | 580104 | 80   |
| 498830942 | NEWAR EVELYN           | WHITE       | FEMALE | 550513 | 77   |
| 498966992 | SMITHIE JEAN DIANE     | WHITE       | FEMALE | 561208 | 78   |
| 499446234 | ALIERMAN JAMES         | WHITE       | MALE   | 580108 | 90   |
| 508992390 | CURTIS ELIZABETH ANN   | WHITE       | FEMALE | 551015 | 77   |
| 514215872 | SWIFT SALLY JANE       | BLACK       | FEMALE | 581130 | 80   |
| 514312603 | DECKER GERTRUDE MARY   | WHITE       | FEMALE | 550706 | 77   |
| 521491463 | HERNANDEZ RAFAEL JOSE  | SPANISH     | MALE   | 470111 | 79   |
| 533101049 | WANN LAURA D           | BLACK       | FEMALE | 581006 | 80   |
| 540134327 | SEGNALLA LYNN MARIE    | WHITE       | FEMALE | 550620 | 77   |
| 540744380 | HUGHES DAVID CRAIG     | WHITE       | MALE   | 560111 | 78   |
| 552229252 | CARTER ANN             | WHITE       | FEMALE | 501205 | 79   |
| 555180626 | BENSON SHELIA JOAN     | WHITE       | FEMALE | 580903 | 80   |
| 556052771 | WARD JULIA CAROL       | WHITE       | FEMALE | 561212 | 78   |
| 610499543 | FALKENZIE PHILIP       | WHITE       | MALE   | 570321 | 79   |
| 613835160 | KNAPP FREDERICK HAROLD | WHITE       | MALE   | 550118 | 77   |
| 614627931 | SKRZYNSKI VINCENT P    | WHITE       | MALE   | 550730 | 77   |
| 628249652 | NUCILLI ARTHUR         | WHITE       | MALE   | 550604 | 77   |
| 686603907 | BLOCK CYRUS            | WHITE       | MALE   | 571011 | 79   |
| 686850331 | BRUZA LESLIE           | SPANISH     | FEMALE | 550511 | 77   |
| 701385226 | ARNOLD KEITH R         | BLACK       | MALE   | 560611 | 78   |
| 707973903 | SANCHEZ MARIA          | SPANISH     | FEMALE | 561201 | 78   |
| 724416930 | BLUM JOSEPH WILLIAM    | WHITE       | MALE   | 560501 | 78   |
| 733267655 | SMOTHERS MAE S         | WHITE       | FEMALE | 550820 | 77   |
| 741997403 | SMITH MARY             | WHITE       | FEMALE | 560609 | 78   |
| 752155359 | ERICKSON SHEILA MARY   | AMER-INDIAN | FEMALE | 581010 | UNKN |
| 754292583 | KUESTER ROBERT E       | WHITE       | MALE   | 580912 | 90   |
| 770441422 | KING THEODORE DAVID    | BLACK       | MALE   | 550807 | 77   |
| 797195848 | MILLER SUSAN ANN       | WHITE       | FEMALE | 570308 | 79   |
| 795419523 | LEACH KAREN LEE        | BLACK       | FEMALE | 571020 | 79   |
| 796202403 | ROGERS LUCILLE A       | WHITE       | FEMALE | 550712 | 77   |
| 796731703 | KENT JERRY             | WHITE       | MALE   | 570220 | 79   |
| 802810964 | MACK JOSEPH            | WHITE       | MALE   | 580401 | 80   |
| 805219718 | NEUMANN CURTIS ARTHUR  | WHITE       | MALE   | 580228 | 80   |
| 805746481 | HARDY COLEEN           | WHITE       | FEMALE | 550102 | 77   |
| 816115339 | GOULD HARRY EDWARD     | BLACK       | MALE   | 560313 | 78   |
| 854351704 | ROSEN LARRY Z          | WHITE       | MALE   | 531005 | 80   |
| 966722832 | DAWSON EMILY E         | BLACK       | FEMALE | 520608 | 78   |
| 878948455 | HAYES HAZEL T          | WHITE       | FEMALE | 531111 | 80   |
| 836122443 | STARR ELLEN ELIZABETH  | WHITE       | FEMALE | 431105 | 77   |
| 886124715 | REDPAITH JOHN          | AMER-INDIAN | MALE   | 570808 | 79   |
| 906922815 | FOSTER HAROLD PETER    | OTHER       | MALE   | 561130 | 78   |
| 907175979 | SCHWIDT RAYMOND F      | WHITE       | MALE   | 580505 | 80   |
| 921293985 | ARRILAGA PAUL          | OTHER       | MALE   | 560225 | 78   |
| 949705539 | WHITE ALBERT L         | WHITE       | MALE   | 590911 | 80   |
| 967397393 | VENDEZ JOSE ROBERTO    | SPANISH     | MALE   | 560703 | 78   |

\$ .11, \$1.60T  
 0.153 CPU SEC.  
 92.000 ENAPSED SEC.

\*READY\*  
-P IN STUDENTS NAME AND STUDENT-ID-NO AND BDAY.

| * NAME                  | STUDENT-ID-NO | BIRTHDATE |
|-------------------------|---------------|-----------|
| * RICHARDSON DEBORAH    | 13222340      | 550310    |
| * POND STUART LESLIE    | 95987328      | 560523    |
| * JONES WILLIAM         | 113591707     | 551010    |
| * GRACE KENNETH R       | 122045525     | 530413    |
| * LEE SUE JEAN          | 133680736     | 560716    |
| * BAILEY EDITH          | 139904799     | 570913    |
| * COOK CYNTHIA ANNA     | 142668782     | 560903    |
| * SCOTT MARY GRACE      | 149514226     | 561230    |
| * CROSBY JANE LILLIAN   | 173049236     | 570510    |
| * TAYLOR VERA M         | 179410240     | 501004    |
| * HALE ROBERT SAMUEL    | 181703558     | 551011    |
| * CUNNINGHAM SARAH      | 186549615     | 570305    |
| * ALWONDIZER KAREN S    | 199472144     | 580215    |
| * SCHILLER ESTHER       | 200914081     | 570728    |
| * CONNERS SUZANNE L     | 207920096     | 571013    |
| * HUFFMAN DIANE CAROL   | 210999051     | 560903    |
| * LANE ANNE             | 214724324     | 570310    |
| * WANDELLA JOANNE D     | 221885343     | 580311    |
| * MENKE SUSAN A         | 229644658     | 571213    |
| * WILTON THORNTON       | 234567890     | 571013    |
| * NELSON RICHARD ELMER  | 236028948     | 551106    |
| * KLEIN BONNIE          | 243155995     | 571113    |
| * JONES DOROTHY         | 251760359     | 550314    |
| * WONG DAVID LEE        | 259964164     | 581120    |
| * FOLEY DONALD C        | 264872649     | 570210    |
| * CROWFOOT WILLIAM      | 281482960     | 560102    |
| * BORG ELIZABETH M      | 304789983     | 570921    |
| * FELDMAN MELVIN SAMUEL | 333272752     | 571021    |
| * BISHOP ARTHUR B       | 336522683     | 570930    |
| * BATES LESLIE ANN      | 360984224     | 581115    |
| * ARMSTRONG NORA M      | 391775833     | 570611    |
| * FIELDS R A            | 403773575     | 570910    |
| * FORINO JOSEPH LOUIS   | 409142975     | 560907    |
| * WU HUA CHUNG          | 418373039     | 560620    |
| * FITZGERALD JOAN       | 430150047     | 581015    |
| * LYNCH JOHN PHILIP     | 432523832     | 571011    |
| * FERGUSON WILLIAM      | 436226647     | 550703    |
| * BORCHINSKY ETHEL      | 463899163     | 550123    |
| * HODGES THOMAS         | 472342535     | 560419    |
| * MEYER LINDA SUE       | 490697986     | 580104    |
| * NEWAR EVELYN          | 498330942     | 550513    |
| * SWITHIE JEAN DIANE    | 493966992     | 561203    |
| * ALTERMAN JAMES        | 499446234     | 580103    |
| * CURTIS ELIZABETH ANN  | 508992390     | 551015    |
| * SWIFT SALLY JANE      | 514215872     | 531130    |
| * DECKER GERIUDE MARY   | 514312603     | 550706    |
| * HERNANDEZ RAFAEL JOSE | 521491463     | 470111    |
| * MANN LAURA D          | 533101049     | 581005    |

|                          |           |        |
|--------------------------|-----------|--------|
| * SEGALLA LYNN MARIE     | 540134327 | 550620 |
| * HUGHES DAVID CRAIG     | 540744380 | 560111 |
| * CARTER ANN             | 552229252 | 501205 |
| * BENSON SHELIA JOAN     | 555180626 | 580803 |
| * WARD JULIA CAROL       | 556052771 | 561212 |
| * FALKENZIE PHILIP       | 610499543 | 570321 |
| * KNAPP FREDERICK HAROLD | 613835160 | 550118 |
| * SKRZYNSKI VINCENT P    | 614627981 | 550730 |
| * NUCILLI ARTHUR         | 628249652 | 550604 |
| * BLOCK CYRUS            | 686603907 | 571011 |
| * BRUZA LESLIE           | 636850331 | 550511 |
| * ARNOLD KEITH R         | 701385226 | 560511 |
| * SANCHEZ MARIA          | 707973903 | 561201 |
| * BLUM JOSEPH WILLIAM    | 724416980 | 560501 |
| * SMOTHERS MAE S         | 733267655 | 550820 |
| * SMITH MARY             | 741997403 | 560609 |
| * ERICKSON SHEILA MARY   | 752155359 | 581010 |
| * KUESTER ROBERT E       | 754292583 | 580912 |
| * KING THEODORE DAVID    | 770441422 | 550807 |
| * WILLER SUSAN ANN       | 787195848 | 570308 |
| * LEACH KAREN LEE        | 795419523 | 571020 |
| * ROGERS LUCILLE A       | 796202403 | 550712 |
| * KENT JERRY             | 796731703 | 570220 |
| * WACK JOSEPH            | 802810964 | 580401 |
| * NEUMANN CURTIS ARTHUR  | 805219718 | 580228 |
| * HARDY COLEEN           | 806746481 | 550102 |
| * GOULD HARRY EDWARD     | 816115339 | 560313 |
| * ROSEN LARRY Z          | 854351704 | 581005 |
| * DAWSON EMILY E         | 866722932 | 520608 |
| * HAYES HAZEL T          | 878948465 | 581111 |
| * STARR ELLEN ELIZABETH  | 896122443 | 481105 |
| * REDPATH JOHN           | 886124715 | 570808 |
| * FOSTER HAROLD PETER    | 906922815 | 561130 |
| * SCHMIDT RAYMOND F      | 907175979 | 580505 |
| * ARRILAGA PAUL          | 921293985 | 560225 |
| * WHITE ALBERT L         | 949705539 | 590911 |
| * MENDEZ JOSE ROBERTO    | 967397399 | 560708 |
| * * \$ .06, \$1.66T      |           |        |
| * * 0.092 CPU SEC.       |           |        |
| * * 66.292 ELAPSED SEC.  |           |        |



\*READY:  
 -\* THE NEXT COMMAND PRINTS ONLY THE FIRST 5 RECORDS  
 + AND PRINTS THE SIN AND BIRTHDATE IN MORE TRADITIONAL FORMATS.

\* \$ .00, \$1.67T  
 \* 0.004 CPU SEC.  
 \* 1.579 ELAPSED SEC.

\*READY:  
 -P@COUNT=5 IN STUDENT'S NAME, SIN@SSN, BDAY@DATE.

| * NAME               | STUDENT-ID-NO | BIRTHDATE |
|----------------------|---------------|-----------|
| * RICHARDSON DEBORAH | 013-22-2340   | 03-10-55  |
| * POND STUART LESLIE | 095-88-7328   | 05-28-56  |
| * JONES WILLIAM      | 113-59-1707   | 10-10-55  |
| * GRACE KENNETH R    | 122-04-5525   | 04-18-58  |
| * LEE SUE JEAN       | 133-68-0736   | 07-16-56  |

\* \$ .01, \$1.69T  
 \* 0.018 CPU SEC.  
 \* 6.988 ELAPSED SEC.

\*READY:  
 -\* PRINTING THE FIRST 5 RECORDS IN THE SET IN REVERSE ORDER.

\* \$ .00, \$1.68T  
 \* 0.003 CPU SEC.  
 \* 1.398 ELAPSED SEC.

\*READY:  
 -P@REVERSE@FIRST=5 IN STUDENT'S NAME, SIN@SSN, BDAY@DATE.

| * NAME               | STUDENT-ID-NO | BIRTHDATE |
|----------------------|---------------|-----------|
| * LEE SUE JEAN       | 133-68-0736   | 07-16-56  |
| * GRACE KENNETH R    | 122-04-5525   | 04-18-58  |
| * JONES WILLIAM      | 113-59-1707   | 10-10-55  |
| * POND STUART LESLIE | 095-88-7328   | 05-28-56  |
| * RICHARDSON DEBORAH | 013-22-2340   | 03-10-55  |

\* \$ .01, \$1.69T  
 \* 0.018 CPU SEC.  
 \* 6.957 ELAPSED SEC.

\*READY:  
--\* EVERY FIFTH RECORD WILL BE PRINTED BEGINNING WITH THE TWENTIETH RECORD.

\* \$ .00, \$1.70T  
\* 0.003 CPU SEC.  
\* 1.675 ELAPSED SEC.

\*READY:  
-P@FIRST=20@INCREMENT=5 IN STUDENTS SIN, NAME.

| * STUDENT-ID-NO       | * NAME                   |
|-----------------------|--------------------------|
| * 234567890           | * WILSON THORNTON        |
| * 264872649           | * FOLEY DONALD C         |
| * 360994224           | * BATES LESLIE ANN       |
| * 430150047           | * FITZGERALD JOAN        |
| * 490697986           | * MEYER LINDA SUE        |
| * 514215872           | * SWIFT SALLY JANE       |
| * 540744380           | * HUGHES DAVID CRAIG     |
| * 613835160           | * KNAPP FREDERICK HAROLD |
| * 701385226           | * ARNOLD KEITH R         |
| * 752155359           | * ERICKSON SHEILA MARY   |
| * 796202403           | * ROGERS LUCILLE A       |
| * 816115339           | * GOULD HARRY EDWARD     |
| * 886124715           | * REDPATH JOHN           |
| * 967397398           | * MENDEZ JOSE ROBERTO    |
| * \$ .02, \$1.71T     |                          |
| * 0.026 CPU SEC.      |                          |
| * 10.947 ELAPSED SEC. |                          |

\*READY:  
--\* THE NAMES IN STUDENTS ARE NOT IN ALPHABETICAL ORDER. TO SORT THE STUDENTS DATA SET  
\* BY NAME, THE SORT COMMAND IS USED.

\* \$ .00, \$1.72T  
\* 0.004 CPU SEC.  
\* 1.358 ELAPSED SEC.

\*READY:  
-SORT IN STUDENTS NAME.  
\* 85 RECORDS IN RESULT SET

\* \$ .07, \$1.78T  
\* 0.166 CPU SEC.  
\* 2.558 ELAPSED SEC.

\*READY:  
 --\* THE NAME FIELD HAS BEEN MOVED TO THE FRONT OF THE DATA SET AS SHOWN BY PRINTING THE RESULT SET. \*  
 + THE RESULT SET IS PRINTED BY DEFAULT WHEN NO DATA SET IS EXPLICITLY SPECIFIED.

\* \$ .00, \$1.79T  
 \* 0.004 CPU SEC.  
 \* 1.728 ELAPSED SEC.

\*READY:  
 -P@C=10.

| * NAME               | STUDENT-ID-NO | ETHNICITY | SEX    | BIRTHDATE | CLASS-YEAR |
|----------------------|---------------|-----------|--------|-----------|------------|
| * ALWONDIZER KAREN S | 199472144     | WHITE     | FEMALE | 580215    | 90         |
| * ALTERMAN JAMES     | 499446234     | WHITE     | MALE   | 590109    | 90         |
| * ARMSTRONG NORA M   | 391775833     | BLACK     | FEMALE | 570611    | 79         |
| * ARNOLD KEITH R     | 701385226     | BLACK     | MALE   | 560611    | 78         |
| * ARRILLAGA PAUL     | 921293985     | OTHER     | MALE   | 560225    | 78         |
| * BAILEY EDITH       | 139904799     | BLACK     | FEMALE | 570913    | 77         |
| * BATES LESLIE ANN   | 360984224     | WHITE     | FEMALE | 581115    | 90         |
| * BENSON SHELIA JOAN | 555180626     | WHITE     | FEMALE | 580903    | 80         |
| * BISHOP ARTHUR B    | 336522688     | WHITE     | MALE   | 570930    | 79         |
| * BLOCK CYRUS        | 586603907     | WHITE     | MALE   | 571011    | 79         |

\* \$ .02, \$1.80T  
 \* 0.022 CPU SEC.  
 \* 14.021 ELAPSED SEC.

\*READY:  
 --\* THE FIRST 5 RECORDS ARE PRINTED USING THE VERTICAL PRINT FORMAT.

\* \$ .00, \$1.81T  
 \* 0.004 CPU SEC.  
 \* 1.616 ELAPSED SEC.

\*READY:  
 -P@VC=5.

|                |                    |
|----------------|--------------------|
| *NAME          | ALWONDIZER KAREN S |
| *STUDENT-ID-NO | 199472144          |
| *ETHNICITY     | WHITE              |
| *SEX           | FEMALE             |
| *BIRTHDATE     | 580215             |
| *CLASS-YEAR    | 90                 |

|                |                |
|----------------|----------------|
| *NAME          | ALTERMAN JAMES |
| *STUDENT-ID-NO | 499446234      |
| *ETHNICITY     | WHITE          |
| *SEX           | MALE           |
| *BIRTHDATE     | 580103         |
| *CLASS-YEAR    | 80             |

```

*NAME ARMSTRONG NORA M
*STUDENT-ID-NO 391775833
*ETHNICITY BLACK
*SEX FEMALE
*BIRTHDATE 570611
*CLASS-YEAR 79

*NAME ARNOLD KEITH R
*STUDENT-ID-NO 701385226
*ETHNICITY BLACK
*SEX MALE
*BIRTHDATE 560611
*CLASS-YEAR 78

*NAME ARRILLAGA PAUL
*STUDENT-ID-NO 921293985
*ETHNICITY OTHER
*SEX MALE
*BIRTHDATE 560225
*CLASS-YEAR 78

* $.02, $1.83T
* 0.037 CPU SEC.
* 17.852 ELAPSED SEC.

```

\*READY:

-- THE NEXT COMMAND DEMONSTRATES THE VARIOUS WAYS IN WHICH A CATEGORICAL NUMERICAL FIELD CAN BE PRINTED. \*

-- NOTE THAT THE USE OF THE FIELD MODIFIERS IN THIS EXAMPLE IS ILLUSTRATIVE, I.E., IT DOES NOT MAKE MUCH SENSE.

```

* $.01, $1.83T
* 0.004 CPU SEC.
* 1.974 ELAPSED SEC.

```

\*READY:

-P=C=10 IN FIN-AID SOF, SOF#ABBR, SOF#VALUE, SOF#ABBR#EX, SOF#VAL#SSN, SOF#VAL#DATE, SOF#DESC.

\*DATA SET "FIN-AID

?test

\*[KEY] I E S I

\* E3 C5 E2 E3

| * SOURCE-OF-FUNDS | SOF | SOURCE-OF-FUNDS | SOF         | SOURCE-OF-FUNDS | SOURCE-OF-FUNDS    |
|-------------------|-----|-----------------|-------------|-----------------|--------------------|
| UNIVERSITY UNIV   | 1   | 01              | 000-00-0001 | 00-01-00        | UNIVERSITY         |
| UNIVERSITY UNIV   | 1   | 01              | 000-00-0001 | 00-01-00        | UNIVERSITY         |
| FEDERAL-GOV FG0V  | 3   | 03              | 000-00-0003 | 00-03-00        | FEDERAL GOVERNMENT |
| INDUSTRY IND      | 4   | 04              | 000-00-0004 | 00-04-00        | INDUSTRY           |
| INDUSTRY IND      | 4   | 04              | 000-00-0004 | 00-04-00        | INDUSTRY           |
| UNIVERSITY UNIV   | 1   | 01              | 000-00-0001 | 00-01-00        | UNIVERSITY         |
| UNIVERSITY UNIV   | 1   | 01              | 000-00-0001 | 00-01-00        | UNIVERSITY         |
| UNIVERSITY UNIV   | 1   | 01              | 000-00-0001 | 00-01-00        | UNIVERSITY         |
| UNIVERSITY UNIV   | 1   | 01              | 000-00-0001 | 00-01-00        | UNIVERSITY         |
| STATE-GOV SG0V    | 2   | 02              | 000-00-0002 | 00-02-00        | STATE GOVERNMENT   |

```

* $.05, $1.89T
* 0.125 CPU SEC.
* 22.566 ELAPSED SEC.

```

\*READY:  
 -\* NOW A LOOK AT SOME OF THE REPORT GENERATING FEATURES AVAILABLE IN MICRO. \*  
 \*\* THE NEXT SET COMMAND SETS THE PAGE CONTROL CHARACTER FOR QUARTER PAGE SKIPS  
 + BY USING THE VERTICAL TAB ("VT") FACILITY ON A SPECIALLY EQUIPPED TERMINAL. \*  
 + THE SAME QUARTER PAGE SKIPPING CAN BE ACCOMPLISHED ON A PRINTER BY SETTING THE PCC TO 4. \*  
 \*\* THE FOLLOWING SET COMMAND SPECIFIES A TITLE TO BE USED BY THE PRINT COMMAND.

\* \$ .01, \$1.90T  
 \* 0.004 CPU SEC.  
 \* 1.699 ELAPSED SEC. \*

\*READY:  
 -SET PCC=VT .

\* \$ .00, \$1.90T  
 \* 0.003 CPU SEC.  
 \* 1.168 ELAPSED SEC. \*

\*READY:  
 -SET TITLE = 'A LISTING OF STUDENT RECORDS STRATIFIED BY RACE AND SEX'.

\* \$ .00, \$1.91T  
 \* 0.003 CPU SEC.  
 \* 1.644 ELAPSED SEC. \*

\*READY:  
 -SORT IN STUDENTS RACE, SEX, CLYR, NAME.  
 \* 85 RECORDS IN RESULT SET

\* \$ .02, \$1.93T  
 \* 0.050 CPU SEC.  
 \* 1.922 ELAPSED SEC. \*

\*READY:  
 -\* NOTICE HOW USE OF THE STRATIFIED PRINT SAVES SPACE (HORIZONTALLY)  
 +AND THAT THE PAGE AND DATE IS PRINTED.

\* \$ .01, \$1.93T  
 \* 0.003 CPU SEC.  
 \* 1.390 ELAPSED SEC. \*

\*READY:  
 -PAGSTRAT=2,COUNT=60,PPD IN IT RACE,SEX,CLYR,NAME,BDAY,DATE,SIN,SSN.  
 \*

JUN 29, 1977

PAGE 01

## \*A LISTING OF STUDENT RECORDS STRATIFIED BY RACE AND SEX

| *ETHNICITY | *SEX | *CLASS-YEAR | NAME                  | BIRTH-DATE | STUDENT-ID-Nº |
|------------|------|-------------|-----------------------|------------|---------------|
|            |      |             | WHITE                 |            |               |
|            |      |             | FEMALE                |            |               |
| *          | *    | *           |                       |            |               |
|            |      | 77          | BORCHINSKY ETHEL      | 01-23-55   | 463-89-8163   |
| *          | *    | 77          | CURTIS ELIZABETH ANN  | 10-15-55   | 508-99-2890   |
| *          | *    | 77          | DECKER GERTRUDE MARY  | 07-06-55   | 514-31-2603   |
| *          | *    | 77          | HARDY COLEEN          | 01-02-55   | 906-74-6481   |
| *          | *    | 77          | JONES DOROTHY         | 03-14-55   | 251-75-0359   |
| *          | *    | 77          | NEWAR EVELYN          | 05-13-55   | 499-83-0942   |
| *          | *    | 77          | ROGERS LUCILLE A      | 07-12-55   | 796-20-2403   |
| *          | *    | 77          | SEGNALLA LYNN MARIE   | 06-20-55   | 540-13-4327   |
| *          | *    | 77          | SMOTHERS MAE S        | 08-20-55   | 733-26-7655   |
| *          | *    | 77          | STARR ELLEN ELIZABETH | 11-05-49   | 886-12-2443   |
| *          | *    | 78          | HUFFMAN DIANE CAROL   | 09-08-56   | 210-99-9051   |
| *          | *    | 78          | SCOTT MARY GRACE      | 12-30-56   | 149-51-4226   |
| *          | *    | 79          | SMITH MARY            | 06-09-56   | 741-99-7403   |
| *          | *    | 78          | SMITHIE JEAN DIANE    | 12-08-56   | 498-96-6992   |
| *          | *    | 79          | WARD JULIA CAROL      | 12-12-56   | 556-05-2771   |
| *          | *    | 79          | BORG ELIZABETH M      | 09-21-57   | 304-78-9983   |
| *          | *    | 79          | CARTER ANN            | 12-05-50   | 552-22-9252   |
| *          | *    | 79          | CONNERS SUZANNE L     | 10-13-57   | 207-92-0096   |
| *          | *    | 79          | CROSBY JANE LILLIAN   | 05-10-57   | 173-04-9236   |
| *          | *    | 79          | CUNNINGHAM SARAH      | 03-05-57   | 196-54-9615   |
| *          | *    | 79          | KLEIN BONNIE          | 11-13-57   | 243-15-5995   |
| *          | *    | 79          | LANE ANNE             | 03-10-57   | 214-72-4324   |
| *          | *    | 79          | MENKE SUSAN A         | 12-18-57   | 229-64-4659   |
| *          | *    | 79          | MILLER SUSAN ANN      | 03-08-57   | 787-19-5848   |
| *          | *    | 79          | SCHILLER ESTHER       | 07-29-57   | 200-91-4081   |
| *          | *    | 80          | ALMONDIZER KAREN S    | 02-15-58   | 199-47-2144   |
| *          | *    | 80          | BATES LESLIE ANN      | 11-15-58   | 360-93-4224   |
| *          | *    | 80          | BENSON SHELIA JOAN    | 08-03-58   | 555-18-0626   |
| *          | *    | 90          | HAYES HAZEL T         | 11-11-58   | 978-94-8465   |
| *          | *    | 80          | MANDELLA JOANNE D     | 08-11-58   | 221-83-5343   |

# EXISTING OF STUDENT RECORDS STRATIFIED BY RACE AND SEX

| * ETHNICITY | * CLASS-YEAR | * NAME                 | * BIRTH-DATE | * STUDENT-ID-NO |
|-------------|--------------|------------------------|--------------|-----------------|
| WHITE       | 77           | FURGUSON WILLIAM       | 07-03-55     | 436-22-6547     |
| MALE        | 77           | HALE ROBERT SAMUEL     | 10-11-55     | 191-70-3558     |
|             | 77           | JONES WILLIAM          | 10-10-55     | 113-59-1707     |
|             | 77           | KNAPP FREDERICK HAROLD | 01-18-55     | 613-83-5160     |
|             | 77           | NELSON RICHARD ELMER   | 11-06-55     | 236-02-8949     |
|             | 77           | NUCILLI ARTHUR         | 06-04-55     | 628-24-9652     |
|             | 77           | SKRZYNSKI VINCENT P    | 07-30-55     | 614-52-7991     |
|             | 77           | BLUM JOSEPH WILLIAM    | 05-01-56     | 724-41-5930     |
|             | 78           | FORINO JOSEPH LOUIS    | 09-07-56     | 409-14-2875     |
|             | 78           | HODGES THOMAS          | 04-19-56     | 472-34-2535     |
|             | 78           | HUGHES DAVID CRAIG     | 01-11-56     | 540-74-4390     |
|             | 78           | POND STUART LESLIE     | 05-28-56     | 095-89-7329     |
|             | 79           | BISHOP ARTHUR B        | 09-30-57     | 336-52-2699     |
|             | 79           | BLOCK CYRUS            | 10-11-57     | 686-60-3907     |
|             | 79           | FALKENZIE PHILIP       | 03-21-57     | 610-42-9543     |
|             | 79           | FELDMAN MELVIN SAMUEL  | 10-21-57     | 333-27-2752     |
|             | 79           | KENT JERRY             | 02-20-57     | 796-73-1703     |
|             | 79           | LYNCH JOHN PHILIP      | 10-11-57     | 432-52-3932     |
|             | 79           | WILTON THORNTON        | 10-13-57     | 234-56-7890     |
|             | 80           | ALTERMAN JAMES         | 01-08-58     | 499-44-6234     |
|             | 80           | GRACE KENNETH R        | 04-18-58     | 122-04-5525     |
|             | 80           | KUESTER ROBERT E       | 08-12-58     | 754-29-2583     |
|             | 80           | WACK JOSEPH            | 04-01-58     | 802-81-0954     |
|             | 80           | NEUMAN CURTIS ARTHUR   | 02-29-58     | 805-21-0718     |
|             | 90           | ROSEN LARRY Z          | 10-05-58     | 954-35-1704     |
|             | 80           | SCHMIDT RAYMOND F      | 05-05-59     | 907-17-5979     |
|             | 90           | WHITE ALBERT L         | 09-11-59     | 949-70-5539     |

# ANALYSIS OF STUDENT RECORDS STRATIFIED BY RACE AND SEX

|              |         |            |               |             |  |
|--------------|---------|------------|---------------|-------------|--|
| *ETHNICITY   | WHITE   |            |               |             |  |
| *SEX         | UNKNOWN |            |               |             |  |
| * CLASS-YEAR | NAME    | BIRTHDATE  | STUDENT-ID-N° |             |  |
| * **         | 79      | FIELDS R A | 09-10-57      | 403-77-3575 |  |
| * **         |         |            |               |             |  |
| * **         |         |            |               |             |  |

PAGE 04

JUN 29, 1977

## \*A LISTING OF STUDENT RECORDS STRATIFIED BY RACE AND SEX

| *ETHNICITY   | BLACK   |                    |               |             |  |
|--------------|---------|--------------------|---------------|-------------|--|
| *SEX         | FEMALE  |                    |               |             |  |
| * CLASS-YEAR | NAME    | BIRTHDATE          | STUDENT-ID-NO |             |  |
| *            | 77      | BAILEY EDITH       | 09-13-57      | 139-90-4799 |  |
| *            | 77      | RICHARDSON DEBORAH | 03-10-55      | 013-22-2340 |  |
| *            | \$ .06, | \$2.00T            |               |             |  |
| *            | 0.093   | CPU SEC.           |               |             |  |
| *            | 51.717  | ELAPSED SEC.       |               |             |  |

\*READY:

-\* END EXERCISE B-2.

|   |         |              |
|---|---------|--------------|
| * | \$ .00, | \$2.00T      |
| * | 0.003   | CPU SEC.     |
| * | 1.225   | ELAPSED SEC. |

\*READY:

-\*



## EXERCISE B-3 - FIND, SORT AND SET ORL COMMANDS.

\* \$ .00, \$2.01T  
 \* 0.003 CPU SEC.  
 \* 3.978 ELAPSED SEC.

\*READY:  
 -FIND IN STUDENTS W CLYR IS 80 AND RACE IS BLACK.  
 \* 04 (4.70%) RECORDS FOUND  
 \* 04 RECORDS IN RESULT SET

\* \$ .12, \$2.13T  
 \* 0.299 CPU SEC.  
 \* 4.143 ELAPSED SEC.

\*READY:  
 -PRINT IN IT NAME, CLYR.

| * NAME             | CLAS-SEAR |
|--------------------|-----------|
| * FITZGERALD JOAN  | 80        |
| * MEYER LINDA SUE  | 80        |
| * SWIFT SALLY JANE | 80        |
| * MANN LAURA D     | 80        |

\* \$ .01, \$2.14T  
 \* 0.015 CPU SEC.  
 \* 5.258 ELAPSED SEC.

\*READY:  
 -SORT IN IT CLYR,NAME.  
 \* 04 RECORDS IN RESULT SET

\* \$ .01, \$2.15T  
 \* 0.032 CPU SEC.  
 \* 1.754 ELAPSED SEC.

\*READY:  
 -- THE OUTPUT RECORD LENGTH (ORL) IS SHORTENED TO SHOW THE EFFECT OF 'WRAP-AROUND' IN THE PRINT COMMAND.

\* \$ .00, \$2.15T  
 \* 0.003 CPU SEC.  
 \* 1.927 ELAPSED SEC.

\*READY:  
 -SET ORL=65.

\* \$ .00, \$2.16T  
 \* 0.003 CPU SEC.  
 \* 1.160 ELAPSED SEC.

\*READY:  
 -P.

| * CLASS-YEAR | NAME             | STUDENT-ID-NO |
|--------------|------------------|---------------|
| * ETHNICITY  | SEX              | BIRTHDATE     |
| * 80         | FITZGERALD JOAN  | 531015        |
| * BLACK      | FEMALE           | 531015        |
| * 80         | WANN LAURA D     | 531006        |
| * BLACK      | FEMALE           | 531006        |
| * 80         | MEYER LINDA SUE  | 530104        |
| * BLACK      | FEMALE           | 530104        |
| * 80         | SWIFT SALLY JANE | 531130        |
| * BLACK      | FEMALE           | 531130        |

\* \$ .01, \$2.17T  
 \* 0.021 CPU SEC.  
 \* 9.464 ELAPSED SEC.

\*READY:  
 -\* THE ORL IS RESET AND THE RESULT SET IS PRINTED AGAIN

\* \$ .00, \$2.18T  
 \* 0.003 CPU SEC.  
 \* 1.524 ELAPSED SEC.

\*READY:  
 -SET ORL=118.

\* \$ .00, \$2.18T  
 \* 0.003 CPU SEC.  
 \* 1.172 ELAPSED SEC.

\*READY:  
 -P IT.

| CLASS-YEAR | NAME             | STUDENT-ID-NO | ETHNICITY | SEX    | BIRTHDATE |
|------------|------------------|---------------|-----------|--------|-----------|
| 80         | FITZGERALD JOAN  | 430150047     | BLACK     | FEMALE | 581015    |
| 80         | MANN LAURA D     | 533101049     | BLACK     | FEMALE | 581006    |
| 80         | MEYER LINDA SUE  | 490697936     | BLACK     | FEMALE | 580104    |
| 80         | SWIFT SALLY JANE | 514215872     | BLACK     | FEMALE | 581130    |

\* \$ .01, \$2.19T  
 \* 0.017 CPU SEC.  
 \* 7.710 ELAPSED SEC.

\*READY:  
 -\* END EXERCISE B-3.

\* \$ .00, \$2.19T  
 \* 0.003 CPU SEC.  
 \* 1.216 ELAPSED SEC.

\*READY:  
 -\*

♦♦ EXERCISE B-4 - EXAMPLES OF FIND, PRINT AND REMOVE COMMANDS: ALSO THE DIFFERENCE BETWEEN THE "WATCH" VERB AND "IS" VERB IN THE FIND COMMAND.

\* \$ .01, \$2.20T  
\* 0.003 CPU SEC.  
\* 1.410 ELAPSED SEC.

\*READY:  
-FIND IN STUDENTS WHERE NAME MATCHES SMITH.  
\* 02 (2.35%) RECORDS FOUND  
\* 02 RECORDS IN RESULT SET

\* \$ .02, \$2.22T  
\* 0.044 CPU SEC.  
\* 2.450 ELAPSED SEC.

\*READY:  
-P IN IT NAME.

\* NAME  
\*-----  
\* SMITHIE JEAN DIANE  
\* SMITH MARY

\* \$ .01, \$2.23T  
\* 0.013 CPU SEC.  
\* 3.202 ELAPSED SEC.

\*READY:  
-NAME IT SMITH.

\* \$ .00, \$2.23T  
\* 0.003 CPU SEC.  
\* 1.185 ELAPSED SEC.

\*READY:  
-\* ANY NAME EXACTLY MATCHING "SMITH MARY" FOLLOWED BY TRAILING BLANKS TO THE WIDTH OF THE NAME FIELD WILL BE REMOVED.

\* \$ .00, \$2.23T  
\* 0.003 CPU SEC.  
\* 2.023 ELAPSED SEC.

\*READY:  
-REMOVE FROM SMITH WHERE NAME IS 'SMITH MARY'.  
\* 01 (50.00%) RECORDS REMOVED  
\* 01 RECORDS IN RESULT SET

\* \$ .02, \$2.25T  
\* 0.050 CPU SEC.  
\* 2.497 ELAPSED SEC.

\*READY:  
-P IN IT NAME.

\* NAME  
\*-----  
\* SMITHIE JEAN DIANE  
\*  
\* \$ .01, \$2.27T  
\* 0.012 CPU SEC.  
\* 2.304 ELAPSED SEC.  
\*

\*READY:  
-\* HERE IS ANOTHER WAY OF ACCOMPLISHING THE SAME TASK.

\* \$ .00, \$2.27T  
\* 0.003 CPU SEC.  
\* 1.505 ELAPSED SEC.  
\*

\*READY:  
-FIND IN SMITH WHERE NAME IS 'SMITH WARY'.  
\* 01 (50.00%) RECORDS FOUND  
\* 01 RECORDS IN RESULT SET

\* \$ .02, \$2.29T  
\* 0.034 CPU SEC.  
\* 2.453 ELAPSED SEC.  
\*

\*READY:  
-REMOVE IT FROM SMITH.  
\* 01 (50.00%) RECORDS REMOVED  
\* 01 RECORDS IN RESULT SET

\* \$ .01, \$2.30T  
\* 0.024 CPU SEC.  
\* 2.300 ELAPSED SEC.  
\*

\*READY:  
-P IN IT NAME.

\* NAME  
\*-----  
\* SMITHIE JEAN DIANE  
\*  
\* \$ .01, \$2.30T  
\* 0.012 CPU SEC.  
\* 2.313 ELAPSED SEC.  
\*

```
*READY:
-* END EXERCISE B-4.
* $.00, $2.31T
* 0.003 CPU SEC.
* 1.224 ELAPSED SEC.

*READY:
-*
```

## EXERCISE B-5 - EXAMPLES OF FIND, NAME, LIST &amp; RELEASE COMMANDS.

```
* $.00, $2.31T
* 0.003 CPU SEC.
* 4.007 ELAPSED SEC.
```

```
*READY:
-FIND IN FIN-AID WHERE SOF IS UNIVERSITY OR FEDERAL-GOV OR STATE-GOV.
* 30 (88.23%) RECORDS FOUND
* 30 RECORDS IN RESULT SET
```

```
* $.02, $2.33T
* 0.038 CPU SEC.
* 2.570 ELAPSED SEC.
```

```
*READY:
-* NOTICE THE PRESENCE OF THE RESULT SET.
```

```
* $.00, $2.33T
* 0.003 CPU SEC.
* 1.393 ELAPSED SEC.
```

```
*READY:
-LIST ACTIVE SETS.
```

```
*DATA SET NAME STATUS
RESULT TEMP
SMITH TEMP
FIN-AID DISK
STUDENTS DISK

* $.01, $2.34T
* 0.003 CPU SEC.
* 3.976 ELAPSED SEC.
```

```
*READY:
-NAME IT PUBLIC-AID.
```

```
* $.00, $2.35T
* 0.003 CPU SEC.
* 1.226 ELAPSED SEC.
```

```
*READY:
-* NOTICE THE ABSENCE OF THE RESULT SET AND THE ADDITION OF PUBLIC-AID TO THE LIST.
```

```
* $.00, $2.35T
* 0.003 CPU SEC.
* 1.737 ELAPSED SEC.
```

\*READY:  
-LIST ACTIVE SETS.

| *DATA SET NAME | STATUS       |
|----------------|--------------|
| *PUBLIC-AID    | TEMP*        |
| *SMITH         | TEMP*        |
| *FIN-AID       | DISK*        |
| *STUDENTS      | DISK*        |
| * \$ .01,      | \$2.36T      |
| * \$ 0.008     | CPU SEC.     |
| * \$ 3.954     | ELAPSED SEC. |

\*READY:  
-RELEASE FIN-AID.

|            |              |
|------------|--------------|
| * \$ .00,  | \$2.36T      |
| * \$ 0.005 | CPU SEC.     |
| * \$ 1.204 | ELAPSED SEC. |

\*READY:  
-\* FIN-AID IS NO LONGER AN ACTIVE SET.

|            |              |
|------------|--------------|
| * \$ .00,  | \$2.36T      |
| * \$ 0.003 | CPU SEC.     |
| * \$ 1.374 | ELAPSED SEC. |

\*READY:  
-LIST ACTIVE SETS.

| *DATA SET NAME | STATUS       |
|----------------|--------------|
| *PUBLIC-AID    | TEMP*        |
| *SMITH         | TEMP*        |
| *STUDENTS      | DISK*        |
| * \$ .01,      | \$2.37T      |
| * \$ 0.007     | CPU SEC.     |
| * \$ 3.454     | ELAPSED SEC. |

\*READY:  
-RELEASE ALL DATA SETS.

|            |              |
|------------|--------------|
| * \$ .01,  | \$2.38T      |
| * \$ 0.009 | CPU SEC.     |
| * \$ 1.256 | ELAPSED SEC. |



\*READY:  
 --\* ONLY PERMANENT SETS REMAIN FROM THE USER'S DIRECTORY;  
 + NOTICE THE IMPROPER USE OF PRIMES IN THE PREVIOUS LINE.  
 +UNBALANCED PRIMES - YOU NEED TWO PRIMES(')

\* \$ .01, \$2.39T  
 \* 0.004 CPU SEC.  
 \* 2.252 ELAPSED SEC.

\*READY:  
 -LIST SETS.

| *DATA SET NAME       | STATUS |
|----------------------|--------|
| *CARS                | DISK   |
| *ART                 | DISK   |
| *FIN-AID             | DISK   |
| *STUDENTS            | DISK   |
| *PUBLIC.DATA         | DISK   |
| * \$ .01, \$2.39T    |        |
| * 0.008 CPU SEC.     |        |
| * 4.353 ELAPSED SEC. |        |

\*READY:  
 --\* END EXERCISE B-5.

\* \$ .00, \$2.39T  
 \* 0.003 CPU SEC.  
 \* 1.204 ELAPSED SEC.

\*READY:  
 --\*

## EXERCISE B-6 - CROSSTABULATE (XTAB), COMBINE AND REMOVE.

\* \$ .00, \$2.40T  
 \* 0.003 CPU SEC.  
 \* 3.935 ELAPSED SEC.

\*READY:  
 -XTAB IN FIN-AID CLASS-LEVEL.  
 \*DATA SET "FIN-AID " IS TO BE UNSCRAMBLED. PLEASE ENTER KEY.  
 ?test  
 \*(KEY) T E S T  
 \* E3 C5 E2 E3  
 \* 04 RECORDS IN RESULT SET  
 \* 34 RECORDS REPRESENTED

| * CLASS-LEVEL         | COUNT | PERCENT |
|-----------------------|-------|---------|
| * FRESHMAN            | 6     | 17.64   |
| * SOPHOMORE           | 9     | 26.47   |
| * JUNIOR              | 11    | 32.35   |
| * SENIOR              | 8     | 23.52   |
| * \$ .15, \$2.54T     |       |         |
| * 0.359 CPU SEC.      |       |         |
| * 13.325 ELAPSED SEC. |       |         |

\*READY:  
 -FIND IN FIN-AID WHERE CLASS-LEVEL IS FRESHMAN.  
 \* 06 (17.64%) RECORDS FOUND  
 \* 06 RECORDS IN RESULT SET

\* \$ .02, \$2.56T  
 \* 0.038 CPU SEC.  
 \* 2.866 ELAPSED SEC.

\*READY:  
 -NAME IT FROSH.

\* \$ .00, \$2.56T  
 \* 0.004 CPU SEC.  
 \* 1.187 ELAPSED SEC.

\*READY:  
 -\* THE FOLLOWING TABULATION PROVES THAT THE FIND COMMAND FOUND ONLY FRESHMEN.

\* \$ .00, \$2.57T  
 \* 0.003 CPU SEC.  
 \* 1.493 ELAPSED SEC.

\*READY:  
 -XTAB IN FROSH CLASS-LEVEL.  
 \* 01 RECORDS IN RESULT SET  
 \* 06 RECORDS REPRESENTED

| * CLASS-LEVEL        | COUNT   | PERCENT |
|----------------------|---------|---------|
| * FRESHMAN           | 6       | 100.00  |
| * \$ .04,            | \$2.61T |         |
| * 0.089 CPU SEC.     |         |         |
| * 4.941 ELAPSED SEC. |         |         |

\*READY:  
 -FIND IN FIN-AID WHERE CLASS-LEVEL IS NOT JUNIOR.  
 \* 23 (67.64%) RECORDS FOUND  
 \* 23 RECORDS IN RESULT SET

|                      |         |
|----------------------|---------|
| * \$ .02,            | \$2.62T |
| * 0.039 CPU SEC.     |         |
| * 2.520 ELAPSED SEC. |         |

\*READY:  
 -NAME IT NOT-JR.

|                      |         |
|----------------------|---------|
| * \$ .00,            | \$2.63T |
| * 0.004 CPU SEC.     |         |
| * 1.194 ELAPSED SEC. |         |

\*READY:  
 -XTAB IN NOT-JR CLASS-LEVEL.  
 \* 03 RECORDS IN RESULT SET  
 \* 23 RECORDS REPRESENTED

| * CLASS-LEVEL        | COUNT   | PERCENT |
|----------------------|---------|---------|
| * FRESHMAN           | 6       | 26.08   |
| * SOPHOMORE          | 9       | 39.13   |
| * SENIOR             | 8       | 34.78   |
| * \$ .04,            | \$2.67T |         |
| * 0.092 CPU SEC.     |         |         |
| * 6.176 ELAPSED SEC. |         |         |

\*READY:  
 -REMOVE FROSH FROM NOT-JR.  
 \* 05 (26.08%) RECORDS REMOVED  
 \* 17 RECORDS IN RESULT SET

\* \$ .01, \$2.69T  
 \* 0.024 CPU SEC.  
 \* 2.334 ELAPSED SEC.

\*READY:  
 -\* THE PREVIOUS COMMAND CREATED A SET WITHOUT FRESHMEN OR JUNIORS.

\* \$ .00, \$2.69T  
 \* 0.003 CPU SEC.  
 \* 1.591 ELAPSED SEC.

\*READY:  
 -NAME RESULT EVENYEARS.

\* \$ .00, \$2.69T  
 \* 0.003 CPU SEC.  
 \* 1.245 ELAPSED SEC.

\*READY:  
 -X IN EVENYEARS CLASS-LEVEL.  
 \* 02 RECORDS IN RESULT SET  
 \* 17 RECORDS REPRESENTED

| * CLASS-LEVEL | COUNT | PERCENT |
|---------------|-------|---------|
| * SOPHOMORE   | 9     | 52.94   |
| * SENIOR      | 8     | 47.05   |

\* \$ .04, \$2.73T  
 \* 0.090 CPU SEC.  
 \* 5.021 ELAPSED SEC.

\*READY:  
 -\* NOW THE FRESHMEN ARE INCLUDED AGAIN TO CREATE A SET EQUIVALENT TO NOT-JR.

\* \$ .00, \$2.73T  
 \* 0.004 CPU SEC.  
 \* 1.686 ELAPSED SEC.

\*READY:  
 -COMBINE FROSH AND EVENYEARS.  
 \* 23 RECORDS IN RESULT SET

\* \$ .01, \$2.74T  
 \* 0.025 CPU SEC.  
 \* 2.194 ELAPSED SEC.

\*READY:  
 -\* IF THE SETS ARE IN FACT IDENTICAL THEN COMBINING THESE SETS SHOULD  
 +CREATE A SET WITH NO NEW RECORDS.\*  
 +THIS IS TRUE BECAUSE IDENTICAL RECORDS ARE "WEEDED" AWAY.

\* \$ .01, \$2.75T  
 \* 0.004 CPU SEC.  
 \* 1.497 ELAPSED SEC.

\*READY:  
 -P IN IT COUNT.  
 \*RESULT  
 \*COUNT = 23

\* \$ .01, \$2.75T  
 \* 0.006 CPU SEC.  
 \* 1.941 ELAPSED SEC.

\*READY:  
 -P IN NOT-JR COUNT.  
 \*NOT-JR  
 \*COUNT = 23

\* \$ .01, \$2.76T  
 \* 0.006 CPU SEC.  
 \* 1.985 ELAPSED SEC.

\*READY:  
 -COMBINE IT WITH NOT-JR.  
 \* 23 RECORDS IN RESULT SET

\* \$ .01, \$2.77T  
 \* 0.027 CPU SEC.  
 \* 3.366 ELAPSED SEC.

\*READY:  
 -\* ANOTHER WAY TO PROVE TWO SETS ARE EQUIVALENT IS TO "SUBTRACT" ONE  
 +FROM THE OTHER. THE RESULT SHOULD BE A NULL SET. ( ZERO RECORDS IN THE RESULT SET.)  
 +REMOVE IT FROM NOT-JR.

\* \$ .01, \$2.78T  
 \* 0.006 CPU SEC.  
 \* 1.249 ELAPSED SEC.

\*READY:  
--\* ANOTHER XTAB EXAMPLE.

\* \$ .00, \$2.78T  
\* 0.006 CPU SEC.  
\* 1.233 ELAPSED SEC.

\*READY:  
-XTAB IN STUDENT'S RACE BY SEX.  
\*DATA SET "STUDENTS" IS TO BE UNSCRAMBLED. PLEASE ENTER KEY.

?test  
\* [KEY] T E S T  
\* E3 C5 E2 E3  
\* 12 RECORDS IN RESULT SET  
\* 85 RECORDS REPRESENTED

| * ETHNICITY   | SEX     | COUNT | PERCENT |
|---------------|---------|-------|---------|
| * WHITE       | FEMALE  | 30    | 35.29   |
| * WHITE       | MALE    | 27    | 31.76   |
| * WHITE       | UNKNOWN | 1     | 1.17    |
| * BLACK       | FEMALE  | 11    | 12.94   |
| * BLACK       | MALE    | 4     | 4.70    |
| * AMER-INDIAN | FEMALE  | 1     | 1.17    |
| * AMER-INDIAN | MALE    | 2     | 2.35    |
| * SPANISH     | FEMALE  | 2     | 2.35    |
| * SPANISH     | MALE    | 2     | 2.35    |
| * ORIENTAL    | FEMALE  | 1     | 1.17    |
| * ORIENTAL    | MALE    | 2     | 2.35    |
| * OTHER       | MALE    | 2     | 2.35    |

\* \$ .09, \$2.88T  
\* 0.208 CPU SEC.  
\* 19.548 ELAPSED SEC.

\*READY:  
-XTAB@10P@10% IN FIN-AID TYPE BY AVE AMOUNT.  
\* 06 RECORDS IN RESULT SET  
\* 34 RECORDS REPRESENTED

\* \$ .03, \$2.91T  
\* 0.075 CPU SEC.  
\* 3.261 ELAPSED SEC.

\*READY:  
-P IT.

| * TYPE-OF-AID        | AVE. AMOUNT | COUNT |
|----------------------|-------------|-------|
| * TUITION            | 499.4444    | 9     |
| * ROOM-BOARD         | 317.5000    | 8     |
| * CASH               | 204.0000    | 7     |
| * LOAN               | 343.7500    | 4     |
| * WORK-STUDY         | 325.0000    | 4     |
| * OTHER              | 150.0000    | 2     |
| * \$ .01,            | \$2.92T     |       |
| * 0.019 CPU SEC.     |             |       |
| * 6.204 ELAPSED SEC. |             |       |

\*READY:  
 -XTAB IN FIN-AID TYPE BY YEAR BY TOTAL AMOUNT BY MIN AMOUNT BY MEAN AMT  
 \* BY MAXIMUM AMOUNT.  
 \* 19 RECORDS IN RESULT SET  
 \* 34 RECORDS REPRESENTED

| * TYPE-OF-AID | YEAR-AWARDED | TOT. AMOUNT | MIN. AMOUNT | AVE. AMOUNT | MAX. AMOUNT | COUNT | PERCENT |
|---------------|--------------|-------------|-------------|-------------|-------------|-------|---------|
| * TUITION     | 74           | 2875.00     | 125.00      | 575.0000    | 1200.00     | 5     | 14.70   |
| * TUITION     | 75           | 1325.00     | 235.00      | 441.6667    | 550.00      | 3     | 8.82    |
| * TUITION     | 76           | 295.00      | 295.00      | 295.0000    | 295.00      | 1     | 2.94    |
| * ROOM-BOARD  | 73           | 625.00      | 125.00      | 312.5000    | 500.00      | 2     | 5.88    |
| * ROOM-BOARD  | 74           | 1010.00     | 485.00      | 505.0000    | 525.00      | 2     | 5.88    |
| * ROOM-BOARD  | 75           | 200.00      | 200.00      | 200.0000    | 200.00      | 1     | 2.94    |
| * ROOM-BOARD  | 76           | 530.00      | 250.00      | 265.0000    | 280.00      | 2     | 5.88    |
| * ROOM-BOARD  | 77           | 175.00      | 175.00      | 175.0000    | 175.00      | 1     | 2.94    |
| * CASH        | 74           | 375.00      | 50.00       | 187.5000    | 325.00      | 2     | 5.88    |
| * CASH        | 75           | 303.00      | 75.00       | 101.0000    | 128.00      | 3     | 8.82    |
| * CASH        | 76           | 750.00      | 375.00      | 375.0000    | 375.00      | 2     | 5.88    |
| * LOAN        | 73           | 1000.00     | 180.00      | 500.0000    | 820.00      | 2     | 5.88    |
| * LOAN        | 76           | 225.00      | 225.00      | 225.0000    | 225.00      | 1     | 2.94    |
| * LOAN        | 77           | 150.00      | 150.00      | 150.0000    | 150.00      | 1     | 2.94    |
| * WORK-STUDY  | 74           | 350.00      | 350.00      | 350.0000    | 350.00      | 1     | 2.94    |
| * WORK-STUDY  | 75           | 900.00      | 400.00      | 400.0000    | 400.00      | 2     | 5.88    |
| * WORK-STUDY  | 76           | 150.00      | 150.00      | 150.0000    | 150.00      | 1     | 2.94    |
| * OTHER       | 75           | 100.00      | 100.00      | 100.0000    | 100.00      | 1     | 2.94    |
| * OTHER       | 76           | 200.00      | 200.00      | 200.0000    | 200.00      | 1     | 2.94    |

\* \$ .07, \$2.99T  
 \* 0.145 CPU SEC.  
 \* 27.107 ELAPSED SEC.

\*READY:  
 -\* END EXERCISE B-6.  
 \* \$ .00, \$2.99T  
 \* 0.003 CPU SEC.  
 \* 1.226 ELAPSED SEC.

\*READY:  
 -\*

## EXERCISE B-7 - THE SELECT, DATESTAMP AND KEY COMMANDS.

```

* $.00, $.2.99T
* 0.004 CPU SEC.
* 3.939 ELAPSED SEC.

```

```

*READY:
-SELECT IN STUDENTS NAME,SIN.
* 95 RECORDS IN RESULT SET

```

```

* $.02, $.3.02T
* 0.050 CPU SEC.
* 2.422 ELAPSED SEC.

```

```

*READY:
-PC=7 IT.

```

| * NAME               | STUDENT-ID-NO |
|----------------------|---------------|
| * ALMONDIZER KAREN S | 199472144     |
| * ALTERMAN JAMES     | 499446234     |
| * ARMSTRONG NORA A   | 391775833     |
| * ARNOLD KEITH R     | 701385226     |
| * ARRILAGA PAUL      | 921293985     |
| * BAILEY EDITH       | 139904799     |
| * BATES LESLIE ANN   | 363984224     |

```

* $.01, $.3.03T
* 0.018 CPU SEC.
* 7.220 ELAPSED SEC.

```

```

*READY:
-NAME IT S.

```

```

* $.00, $.3.03T
* 0.003 CPU SEC.
* 1.131 ELAPSED SEC.

```

```

*READY:
-KEY S.
* 95 RECORDS IN RESULT SET

```

```

* $.07, $.3.10T
* 0.172 CPU SEC.
* 3.651 ELAPSED SEC.

```



\*READY:  
-POC=6 IT.

| KEY | NAME               | STUDENT-ID-NO |
|-----|--------------------|---------------|
| 1   | ALMONDIZER KAREN S | 199472144     |
| 2   | ALTERMAN JAMES     | 499446234     |
| 3   | ARMSTRONG NORA M   | 391775833     |
| 4   | ARNOLD KEITH R     | 701385226     |
| 5   | ARRILLAGA PAUL     | 921293985     |
| 6   | BAILEY EDITH       | 139904799     |

\* \$ .01, \$3.12T  
\* 0.018 CPU SEC.  
\* 7.314 ELAPSED SEC.

\*READY:  
-KEY S STARTING WITH 1004.  
\* 95 RECORDS IN RESULT SET

\* \$ .01, \$3.13T  
\* 0.021 CPU SEC.  
\* 1.819 ELAPSED SEC.

\*READY:  
-POC=5.

| KEY  | NAME               | STUDENT-ID-NO |
|------|--------------------|---------------|
| 1004 | ALMONDIZER KAREN S | 199472144     |
| 1005 | ALTERMAN JAMES     | 499446234     |
| 1006 | ARMSTRONG NORA M   | 391775833     |
| 1007 | ARNOLD KEITH R     | 701385226     |
| 1008 | ARRILLAGA PAUL     | 921293985     |

\* \$ .01, \$3.14T  
\* 0.017 CPU SEC.  
\* 6.550 ELAPSED SEC.

\*READY:  
-\* THE EFFECT OF DATESTAMP COMMAND IS SHOWN.

\* \$ .00, \$3.14T  
\* 0.003 CPU SEC.  
\* 1.423 ELAPSED SEC.

\*READY:  
-DATESTAMP IT.  
\* 85 RECORDS IN RESULT SET

\* \$ .07, \$3.22T  
\* 0.180 CPU SEC.  
\* 2.511 ELAPSED SEC.

\*READY:  
-P@C=5 IN RESULT KEY, SIN@SSN, NAME, DATESTAMP@DATE.

| * * * | KEY   | STUDENT-ID-NO | NAME               | DATESTAMP |
|-------|-------|---------------|--------------------|-----------|
| * * * | 1004  | 199-47-2144   | ALMONDIZER KAREN S | 06-29-77  |
| * * * | 1005  | 499-44-6234   | ALTERMAN JAMES     | 06-29-77  |
| * * * | 1006  | 391-77-5833   | ARNSTRONG NORA M   | 06-29-77  |
| * * * | 1007  | 701-38-5226   | ARNOLD KEITH R     | 06-29-77  |
| * * * | 1008  | 921-29-3985   | ARRILLAGA PAUL     | 06-29-77  |
| * * * | s.01, | \$3.23I       |                    |           |
| * * * | 0.017 | CPU SEC.      |                    |           |
| * * * | 7.562 | ELAPSED SEC.  |                    |           |

\*READY:  
-\* END EXERCISE B-7.  
  
\* s.00, \$3.23I  
\* 0.003 CPU SEC.  
\* 1.228 ELAPSED SEC.

\*READY:  
-\*



```
* $.01, $3.39T
* 0.004 CPU SEC.
* 1.614 ELAPSED SEC.
```

```
*READY:
-FIND IN FIN-AID WHERE TYPE-OF-AID IS TUITION OR ROOM-BOARD OR
+WORK-STUDY.
* 21 (61.76%) RECORDS FOUND
* 21 RECORDS IN RESULT SET

* $.02, $3.40T
* 0.038 CPU SEC.
* 2.319 ELAPSED SEC.
```

```
*READY:
-NAME RESULT TEMP1.

* $.00, $3.41T
* 0.003 CPU SEC.
* 1.214 ELAPSED SEC.
```

```
*READY:
-* *****
+ STEP 2
+ "AND SOURCE-OF-FUNDS IS UNIVERSITY"
+ Reference is made to the previous intermediate result set (from
+ Step (1) because there was no previous OR clause
+ *****
* $.01, $3.42T
* 0.004 CPU SEC.
* 1.613 ELAPSED SEC.
```

```
*READY:
-FIND IN TEMP1 WHERE SOURCE-OF-FUNDS IS UNIVERSITY.
* 10 (47.61%) RECORDS FOUND
* 10 RECORDS IN RESULT SET

* $.02, $3.43T
* 0.033 CPU SEC.
* 2.529 ELAPSED SEC.
```

```
*READY:
-NAME RESULT TEMP2.

* $.00, $3.44T
* 0.003 CPU SEC.
* 1.215 ELAPSED SEC.
```

```

*READY:
-* *****
+ STEP 3
+ "OR CLASS-LEVEL IS JUNIOR"
+ Reference is made to TEMP1 because it is the intermediate result
+ set before the previous AND clause (in Step 2)
** *****
* $.01, $3.44T
* $ 0.004 CPU SEC.
* $ 1.612 ELAPSED SEC.

*READY:
-FIND IN TEMP1 WHERE CLASS-LEVEL IS JUNIOR.
* 09 (42.85%) RECORDS FOUND
* 09 RECORDS IN RESULT SET
* $.02, $3.46T
* $ 0.037 CPU SEC.
* $ 2.754 ELAPSED SEC.

*READY:
-NAME RESULT TEMP3.
* $.00, $3.46T
* $ 0.003 CPU SEC.
* $ 1.217 ELAPSED SEC.

*READY:
-COMBINE TEMP2 AND TEMP3.
* 17 RECORDS IN RESULT SET
* $.01, $3.48T
* $ 0.022 CPU SEC.
* $ 1.783 ELAPSED SEC.

*READY:
-NAME RESULT TEMP4.
* $.00, $3.49T
* $ 0.003 CPU SEC.
* $ 1.201 ELAPSED SEC.

*READY:
-RELEASE TEMP2.

```

```

* $.00, $3.481
* 0.005 CPU SEC.
* 1.184 ELAPSED SEC.

```

```

*READY:
-* *****
+ STEP 4
+ "AND TERM IS FALL"
+ Reference is made to TEMP4 because it is the intermediate result
+ set generated after the previous OR clause (in Step 3)
+ *****

```

```

* $.01, $3.491
* 0.004 CPU SEC.
* 1.608 ELAPSED SEC.

```

```

*READY:
-FIND IN TEMP4 WHERE TERM IS FALL.
* 10 (58.82%) RECORDS FOUND
* 10 RECORDS IN RESULT SET

```

```

* $.02, $3.511
* 0.035 CPU SEC.
* 2.385 ELAPSED SEC.

```

```

*READY:
-NAME RESULT TEMP5.

```

```

* $.00, $3.511
* 0.003 CPU SEC.
* 1.222 ELAPSED SEC.

```

```

*READY:
-RELEASE TEMP1.

```

```

* $.00, $3.521
* 0.005 CPU SEC.
* 1.174 ELAPSED SEC.

```

```

*READY:
-* *****
+ STEP 5
+ "OR TERM IS WINTER"
+ Reference is made to TEMP4 because it is the intermediate result
+ set generated (in Step 3) before the previous AND clause
+ *****

```

\* \$ .01, \$3.52T  
 \* 0.004 CPU SEC.  
 \* 1.617 ELAPSED SEC.

\*READY:  
 -FIND IN TEMP4 WHERE TERM IS WINTER.  
 \* 04 (23.52%) RECORDS FOUND  
 \* 04 RECORDS IN RESULT SET

\* \$ .02, \$3.54T  
 \* 0.038 CPU SEC.  
 \* 4.471 ELAPSED SEC.

\*READY:  
 -NAME RESULT TEMP6.

\* \$ .00, \$3.55T  
 \* 0.004 CPU SEC.  
 \* 1.209 ELAPSED SEC.

\*READY:  
 -RELEASE TEMP4.

\* \$ .00, \$3.55T  
 \* 0.005 CPU SEC.  
 \* 1.550 ELAPSED SEC.

\*READY:  
 -COMBINE TEMP5 AND TEMP6.  
 \* 14 RECORDS IN RESULT SET

\* \$ .01, \$3.56T  
 \* 0.024 CPU SEC.  
 \* 1.791 ELAPSED SEC.

\*READY:  
 -NAME RESULT TEMP7.

\* \$ .00, \$3.57T  
 \* 0.004 CPU SEC.  
 \* 1.224 ELAPSED SEC.

\*READY:  
 -RELEASE TEMP5 AND TEMP6.

\* \$ .00, \$3.57T  
 \* 0.006 CPU SEC.  
 \* 1.310 ELAPSED SEC.

```

*READY:
-* *****
+ STEP 6
+ . "AND AMOUNT IS FROM 0 TO 150"
+ Reference is made to TEMP7 because it is the intermediate result
+ set generated (in Step 5) after the previous OR clause
+* *****

```

```

* $.01, $3.53I
* 0.004 CPU SEC.
* 1.566 ELAPSED SEC.

```

```

*READY:
-RELEASE TEMP3.

```

```

* $.01, $3.59I
* 0.005 CPU SEC.
* 1.186 ELAPSED SEC.

```

```

*READY:
-FIND IN TEMP7 WHERE AMOUNT IS FROM 0 TO 150.
* 02 (14.28%) RECORDS FOUND
* 02 RECORDS IN RESULT SET

```

```

* $.02, $3.60I
* 0.035 CPU SEC.
* 2.473 ELAPSED SEC.

```

```

*READY:
-NAME RESULT TEMP8.

```

```

* $.00, $3.61I
* 0.004 CPU SEC.
* 1.216 ELAPSED SEC.

```

```

*READY:
-* *****
+ STEP 7
+ "OR AMOUNT > 750"
+ Reference is made to TEMP7 because it is the intermediate result
+ set generated (in Step 7) before the previous AND clause
+* *****

```

```

* $.01, $3.61I
* 0.004 CPU SEC.
* 1.283 ELAPSED SEC.

```



\*READY:  
-RELEASE TEMP4.

\* \$ .00, \$3.62T  
\* 0.004 CPU SEC.  
\* 1.193 ELAPSED SEC.

\*READY:  
-FIND IN TEMP7 WHERE AMOUNT > 750.  
\* 02 (14.28%) RECORDS FOUND  
\* 02 RECORDS IN RESULT SET

\* \$ .02, \$3.64T  
\* 0.036 CPU SEC.  
\* 3.039 ELAPSED SEC.

\*READY:  
-NAME RESULT TEMP9.

\* \$ .00, \$3.64T  
\* 0.003 CPU SEC.  
\* 1.222 ELAPSED SEC.

\*READY:  
-RELEASE TEMP7.

\* \$ .01, \$3.64T  
\* 0.005 CPU SEC.  
\* 1.192 ELAPSED SEC.

\*READY:  
-COMBINE TEMP8 AND TEMP9.  
\* 04 RECORDS IN RESULT SET

\* \$ .01, \$3.65T  
\* 0.024 CPU SEC.  
\* 1.819 ELAPSED SEC.

\*READY:  
-NAME RESULT TEMP10.

\* \$ .00, \$3.66T  
\* 0.003 CPU SEC.  
\* 1.226 ELAPSED SEC.

\*READY:  
-RELEASE TEMP8.

\* \$ .01, \$3.66T  
\* 0.005 CPU SEC.  
\* 1.578 ELAPSED SEC.

\*READY:  
-\* \*\*\*\*\*  
+ STEP 8  
+ "ALSO WHERE YEAR IS 73 OR 74"  
+ Reference is made to FIN-AID, the originally specified data set  
+ \*\*\*\*\*

\* \$ .01, \$3.67T  
\* 0.004 CPU SEC.  
\* 1.591 ELAPSED SEC.

\*READY:  
-RELEASE TEMP9.

\* \$ .01, \$3.68T  
\* 0.005 CPU SEC.  
\* 1.481 ELAPSED SEC.

\*READY:  
-FIND IN FIN-AID WHERE YEAR IS 73 OR 74.  
\* 14 (41.17%) RECORDS FOUND  
\* 14 RECORDS IN RESULT SET

\* \$ .02, \$3.69T  
\* 0.037 CPU SEC.  
\* 2.437 ELAPSED SEC.

\*READY:  
-NAME RESULT TEMP11.

\* \$ .00, \$3.70T  
\* 0.003 CPU SEC.  
\* 1.223 ELAPSED SEC.

\*READY:  
-\* \*\*\*\*\*  
+ STEP 9  
+ "AND SOURCE-OF-FUND IS INDUSTRY"  
+ Reference is made to TEMP11 because it is the previous  
+ intermediate result set (there was no previous 'OR' because of the  
+ ALSO WHERE)  
+ \*\*\*\*\*

\* \$.01, \$3.70T  
 \* 0.004 CPU SEC.  
 \* 1.613 ELAPSED SEC.

\*READY:  
 -FIND IN TEMP11 WHERE SOURCE-OF-FUNDS IS INDUSTRY.  
 \* 03 (21.42%) RECORDS FOUND  
 \* 03 RECORDS IN RESULT SET

\* \$.02, \$3.72T  
 \* 0.036 CPU SEC.  
 \* 2.514 ELAPSED SEC.

\*READY:  
 -NAME RESULT TEMP12.

\* \$.00, \$3.73T  
 \* 0.004 CPU SEC.  
 \* 1.234 ELAPSED SEC.

\*READY:  
 -RELEASE TEMP11.

\* \$.00, \$3.73T  
 \* 0.005 CPU SEC.  
 \* 1.197 ELAPSED SEC.

\*READY:  
 -\* \*\*\*\*\*  
 + STEP 10  
 + " " (PERIOD INDICATING END OF COMMAND)  
 +\* \*\*\*\*\*

\* \$.01, \$3.74T  
 \* 0.003 CPU SEC.  
 \* 1.614 ELAPSED SEC.

\*READY:  
 -COMBINE TEMP10 AND TEMP12.  
 \* 05 RECORDS IN RESULT SET

\* \$.01, \$3.75T  
 \* 0.023 CPU SEC.  
 \* 1.799 ELAPSED SEC.

\*READY:  
 -RELEASE TEMP10 AND TEMP12.

\* \$.00, \$3.75T  
 \* 0.006 CPU SEC.  
 \* 1.288 ELAPSED SEC.

\*READY:  
 -\* Result set contains records meeting specified criteria.

\* \$ .00, \$3.76T  
 \* 0.003 CPU SEC.  
 \* 1.540 ELAPSED SEC.

\*READY:  
 -P@A@W=60.

| * SIN                | YEAR | TERM | CL | TYPE | SOF  | AWT     | REM*                                                          |
|----------------------|------|------|----|------|------|---------|---------------------------------------------------------------|
| * 113591707          | 74   | F    | SR | TU   | UNIV | 1200.00 | AWARD FROM INTERNATIONAL BUSINESS SYSTEMS OF ARMONK, NEW YORK |
| * 142668782          | 74   | F    | JR | TU   | IND  | 1050.00 |                                                               |
| * 556052771          | 74   | F    | JR | TU   | IND  | 150.00  | SCHOLARSHIP FROM AXALGAVATED INDUSTRIES, INC.                 |
| * 906922815          | 73   | F    | JR | RB   | SGOV | 125.00  |                                                               |
| * 945031192          | 74   | W    | JR | CA   | IND  | 325.00  | GRANT FROM BELT TELEPHONE CO. TO STUDY TELEPATHY              |
| * \$ .02, \$3.78T    |      |      |    |      |      |         |                                                               |
| * 0.037 CPU SEC.     |      |      |    |      |      |         |                                                               |
| * 9.222 ELAPSED SEC. |      |      |    |      |      |         |                                                               |

\*READY:  
 -\* END EXERCISE B-8.

\* \$ .00, \$3.78T  
 \* 0.003 CPU SEC.  
 \* 1.225 ELAPSED SEC.

\*READY:  
 -\*

EXERCISE B-9 - THE RESTRICT AND COMBINE COMMANDS. \*  
 +THE NEXT COMMANDS SPLIT UP THE STUDENTS DATA SET ACCORDING TO THE RECEIPT OF FINANCIAL AID  
 + AS INDICATED IN FIN-AID.

\* \$ .01, \$3.79T  
 \* 0.004 CPU SEC.  
 \* 1.287 ELAPSED SEC.

\*READY:  
 -RESTRICT IN STUDENTS WHERE SIN IS NOT SIN IN FIN-AID.  
 \* 60 RECORDS IN RESULT SET

\* \$ .07, \$3.86T  
 \* 0.178 CPU SEC.  
 \* 2.949 ELAPSED SEC.

\*READY:  
 -NAME IT NO-AID.

\* \$ .00, \$3.86T  
 \* 0.004 CPU SEC.  
 \* 1.137 ELAPSED SEC.

\*READY:  
 -RESTRICT IN STUDENTS WHERE SIN IS SIN IN FIN-AID.  
 \* 25 RECORDS IN RESULT SET

\* \$ .02, \$3.98T  
 \* 0.043 CPU SEC.  
 \* 1.992 ELAPSED SEC.

\*READY:  
 -NAME RESULT GOT-AID.

\* \$ .00, \$3.89T  
 \* 0.003 CPU SEC.  
 \* 1.234 ELAPSED SEC.

\*READY:  
 -\* NOW THE TWO PARTS ARE COMBINED TO CREATE A SET EQUIVALENT TO STUDENTS.

\* \$ .00, \$3.89T  
 \* 0.003 CPU SEC.  
 \* 1.647 ELAPSED SEC.

\*READY:  
-COMBINE NO-AID AND GOT-AID.  
\* 85 RECORDS IN RESULT SET

\* \$ .01, \$3.91T  
\* 0.028 CPU SEC.  
\* 1.805 ELAPSED SEC.

\*READY:

--\* END EXERCISE B-9.

\* \$ .00, \$3.91T  
\* 0.003 CPU SEC.  
\* 1.226 ELAPSED SEC.

\*READY:

--\*

EXERCISE B-10 - THE RESTRICTANDJOIN, JOIN AND SAVE COMMANDS.

\* \$ .00, \$3.91T  
 \* 0.003 CPU SEC.  
 \* 4.002 ELAPSED SEC.

\*READY:  
 -\* CREATING COMPOSITE RECORDS FOR STUDENTS WHO RECEIVE AID.

\* \$ .01, \$3.92T  
 \* 0.003 CPU SEC.  
 \* 1.546 ELAPSED SEC.

\*READY:  
 -RESTRICTANDJOIN STUDENTS WITH FIN-AID BY SIN.  
 \* 25 RECORDS IN RESULT SET

\* \$ .06, \$3.98T  
 \* 0.150 CPU SEC.  
 \* 2.570 ELAPSED SEC.

\*READY:  
 -NAME IT RECIPIENTS.

\* \$ .00, \$3.99T  
 \* 0.003 CPU SEC.  
 \* 1.226 ELAPSED SEC.

\*READY:  
 -\* NOTICE THAT THERE IS NO DATA SET DESCRIPTION FOR A TEMPORARY SET.

\* \$ .00, \$3.99T  
 \* 0.003 CPU SEC.  
 \* 1.624 ELAPSED SEC.

\*READY:  
 -FULLDOC RECIPIENTS.

\*FOR: RECIPIENTS  
 \*DICTIONARY LOCATION: -MIC#001  
 \*DATE: JUN 29, 1977

\*THIS DATASET CANNOT BE DESTROYED.  
 \*THIS DATASET CAN BE REPLACED.  
 \*THIS DATASET IS SCRAMBLED.  
 \*USE COUNT: 1

\*DATA SET DESCRIPTION:  
 \*NO DESCRIPTION EXISTS

| *F(1) | FIELD NAME        | ABBR  | VALUE    | DESCRIPTION                                        | TYPE LENGTH DISP SCALE FACTOR |   |   |     |
|-------|-------------------|-------|----------|----------------------------------------------------|-------------------------------|---|---|-----|
| *F(1) | STUDENT-ID-NO     | SIN   | REQUIRED | STUDENT IDENTIFICATION NUMBER                      | U                             | 4 | 0 |     |
| *F(2) | YEAR-AWARDED      | YEAR  | REQUIRED | YEAR THAT FINANCIAL AID WAS AWARDED (IN YY FORMAT) | U                             | 1 | 4 |     |
| *F(3) | TERM-AWARDED      | TERM  | REQUIRED | TERM IN WHICH FINANCIAL AID WAS AWARDED            | UC                            | 1 | 5 |     |
| *     | CATEGORIES (ONLY) |       |          |                                                    |                               |   |   |     |
| *     | FALL              | F     | 1        | FALL TERM                                          |                               |   |   |     |
| *     | WINTER            | W     | 2        | WINTER TERM                                        |                               |   |   |     |
| *     | SUMMER            | S     | 3        | SUMMER TERM                                        |                               |   |   |     |
| *F(4) | CLASS-LEVEL       | CL    | REQUIRED | CLASS-LEVEL                                        | UC                            | 1 | 6 |     |
| *     | CATEGORIES (ONLY) |       |          |                                                    |                               |   |   |     |
| *     | FRESHMAN          | FR    | 1        | FRESHMAN                                           |                               |   |   |     |
| *     | SOPHOMORE         | SO    | 2        | SOPHOMORE                                          |                               |   |   |     |
| *     | JUNIOR            | JR    | 3        | JUNIOR                                             |                               |   |   |     |
| *     | SENIOR            | SR    | 4        | SENIOR                                             |                               |   |   |     |
| *F(5) | TYPE-OF-AID       | TYPE  | REQUIRED | TYPE-OF-AID                                        | UC                            | 1 | 7 |     |
| *     | CATEGORIES (ONLY) |       |          |                                                    |                               |   |   |     |
| *     | TUITION           | TU    | 1        | TUITION                                            |                               |   |   |     |
| *     | ROOM-BOARD        | RB    | 2        | ROOM AND BOARD                                     |                               |   |   |     |
| *     | CASH              | CA    | 3        | CASH                                               |                               |   |   |     |
| *     | LOAN              | LO    | 4        | LOAN                                               |                               |   |   |     |
| *     | WORK-STUDY        | WS    | 5        | WORK STUDY PROGRAM                                 |                               |   |   |     |
| *     | OTHER             | OTH   | 6        | OTHER                                              |                               |   |   |     |
| *F(6) | SOURCE-OF-FUNDS   | SOF   | REQUIRED | SOURCE OF FUNDS OF AWARDED FINANCIAL AID           | UC                            | 1 | 8 |     |
| *     | CATEGORIES (ONLY) |       |          |                                                    |                               |   |   |     |
| *     | UNIVERSITY        | UNIV  | 1        | UNIVERSITY                                         |                               |   |   |     |
| *     | STATE-GOV         | SGOV  | 2        | STATE GOVERNMENT                                   |                               |   |   |     |
| *     | FEDERAL-GOV       | FGOV  | 3        | FEDERAL GOVERNMENT                                 |                               |   |   |     |
| *     | INDUSTRY          | IND   | 4        | INDUSTRY                                           |                               |   |   |     |
| *     | FOUNDATION        | FOUND | 5        | FOUNDATION                                         |                               |   |   |     |
| *     | OTHER             | OTH   | 6        | OTHER SOURCE                                       |                               |   |   |     |
| *F(7) | AMOUNT            | AMT   | REQUIRED | AMOUNT OF FINANCIAL AID AWARDED                    | S                             | 4 | 9 | 100 |

\*FOR: RECIPIENTS



| *F(8)                       | FIELD NAME          | ABBR    | VALUE    | DESCRIPTION                                 | TYPE LENGTH DISP SCALE FACTOR |
|-----------------------------|---------------------|---------|----------|---------------------------------------------|-------------------------------|
| *                           | REMARK*             | REMARK* |          | ADDITIONAL COMMENTS (OPTIONAL)              | E 4 13                        |
| *                           | FILE NAME           |         |          |                                             |                               |
| *                           | ILIR:REMARKS*       |         |          |                                             |                               |
| *F(9)                       | NAME                | NA      | REQUIRED | NAME OF STUDENT IN LAST FIRST MIDDLE FORMAT | C 24 17                       |
| *                           |                     |         |          | ETHNICITY                                   | UC 1 41                       |
| *F(10)                      | ETHNICITY           | RACE    | REQUIRED | ETHNICITY                                   | DEFAULT                       |
| *                           | CATEGORIES (ONLY)   |         | 0        | UNKNOWN                                     |                               |
| *                           | UNKNOWN             | UNK     | 1        | WHITE                                       |                               |
| *                           | WHITE               | WH      | 2        | BLACK                                       |                               |
| *                           | BLACK               | BL      | 3        | AMERICAN INDIAN                             |                               |
| *                           | AMER-INDIAN         | AI      | 4        | SPANISH SURAMED AMERICAN                    |                               |
| *                           | SPANISH             | SP      | 5        | ORIENTAL                                    |                               |
| *                           | ORIENTAL            | OR      | 6        | OTHER                                       |                               |
| *                           | OTHER               | OTH     |          |                                             |                               |
| *F(11)                      | SEX                 | SEX     | REQUIRED | SEX OF STUDENT                              | CC 1 42                       |
| *                           | CATEGORIES (ONLY)   |         |          |                                             |                               |
| *                           | MALE                | MA      | M        | MALE                                        |                               |
| *                           | FEMALE              | FE      | F        | FEMALE                                      | DEFAULT                       |
| *                           | UNKNOWN             | UNK     | U        | UNKNOWN                                     | UC 3 43                       |
| *F(12)                      | BIRTHDATE           | BDAY    | REQUIRED | DATE OF STUDENT'S BIRTH IN YYMMDD FORMAT    | DEFAULT                       |
| *                           | CATEGORIES          |         | 0        | DATE OF BIRTH IS MISSING                    |                               |
| *                           | MISSING             | MISS    |          |                                             | SC 1 46                       |
| *F(13)                      | CLASS-YEAR          | CLYR    |          | CLASS YEAR IN YY FORMAT                     | DEFAULT                       |
| *                           | CATEGORIES          |         | -1       | UNKNOWN                                     |                               |
| *                           | UNKNOWN             | UNK     |          |                                             |                               |
| *                           | \$ .07,             | \$4.06T |          |                                             |                               |
| *                           | 0.113 CPU SEC.      |         |          |                                             |                               |
| *                           | 70.686 ELAPSED SEC. |         |          |                                             |                               |
| *READY:                     |                     |         |          |                                             |                               |
| -P0V3C=5 RECIPIENTS.        |                     |         |          |                                             |                               |
| *STUDENT-ID-NO 113591707    |                     |         |          |                                             |                               |
| *YEAR-AWARDED 74            |                     |         |          |                                             |                               |
| *TERM-AWARDED FALL          |                     |         |          |                                             |                               |
| *CLASS-LEVEL SENIOR         |                     |         |          |                                             |                               |
| *TYPE-OF-AID TUITION        |                     |         |          |                                             |                               |
| *SOURCE-OF-FUNDS UNIVERSITY |                     |         |          |                                             |                               |

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| *AMOUNT          | 1200.00                                                       |
| *REMARK*         |                                                               |
| *NAME            | JONES WILLIAM                                                 |
| *ETHNICITY       | WHITE                                                         |
| *SEX             | MALE                                                          |
| *BIRTHDATE       | 551019                                                        |
| *CLASS-YEAR      | 77                                                            |
| *STUDENT-ID-NO   | 133680736                                                     |
| *YEAR-AWARDED    | 74                                                            |
| *TERM-AWARDED    | FALL                                                          |
| *CLASS-LEVEL     | JUNIOR                                                        |
| *TYPE-OF-AID     | ROOM-BOARD                                                    |
| *SOURCE-OF-FUNDS | FEDERAL-GOV                                                   |
| *AMOUNT          | 485.00                                                        |
| *REMARK*         | OUTSTANDING SENIOR                                            |
| *NAME            | LEE SUE JEAN                                                  |
| *ETHNICITY       | ORIENTAL                                                      |
| *SEX             | FEMALE                                                        |
| *BIRTHDATE       | 560716                                                        |
| *CLASS-YEAR      | 78                                                            |
| *STUDENT-ID-NO   | 142668782                                                     |
| *YEAR-AWARDED    | 74                                                            |
| *TERM-AWARDED    | FALL                                                          |
| *CLASS-LEVEL     | JUNIOR                                                        |
| *TYPE-OF-AID     | TUITION                                                       |
| *SOURCE-OF-FUNDS | INDUSTRY                                                      |
| *AMOUNT          | 1050.00                                                       |
| *REMARK*         | AWARD FROM INTERNATIONAL BUSINESS SYSTEMS OF ARMONK, NEW YORK |
| *NAME            | COOK CYNTHIA ANNA                                             |
| *ETHNICITY       | BLACK                                                         |
| *SEX             | FEMALE                                                        |
| *BIRTHDATE       | 560908                                                        |
| *CLASS-YEAR      | 78                                                            |
| *STUDENT-ID-NO   | 149514226                                                     |
| *YEAR-AWARDED    | 75                                                            |
| *TERM-AWARDED    | WINTER                                                        |
| *CLASS-LEVEL     | JUNIOR                                                        |
| *TYPE-OF-AID     | CASH                                                          |
| *SOURCE-OF-FUNDS | INDUSTRY                                                      |
| *AMOUNT          | 100.00                                                        |
| *REMARK*         | GRANT FROM THE OILY PETROLEUM CO. TO STUDY OIL SLICKS         |
| *NAME            | SCOTT MARY GRACE                                              |
| *ETHNICITY       | WHITE                                                         |
| *SEX             | FEMALE                                                        |
| *BIRTHDATE       | 561230                                                        |
| *CLASS-YEAR      | 78                                                            |
| *STUDENT-ID-NO   | 173049236                                                     |
| *YEAR-AWARDED    | 75                                                            |
| *TERM-AWARDED    | FALL                                                          |
| *CLASS-LEVEL     | SOPHOMORE                                                     |
| *TYPE-OF-AID     | OTHER                                                         |

\*SOURCE-OF-FUNDS UNIVERSITY  
 \*AMOUNT 100.00  
 \*REMARK\* PART-TIME EMPLOYMENT GRADING PAPERS  
 \*NAME CROSBY JANE LILLIAN  
 \*ETHNICITY WHITE  
 \*SEX FEMALE  
 \*BIRTHDATE 570510  
 \*CLASS-YEAR 79  
 \* \$ .05, \$4.11T  
 \* 0.088 CPU SEC.  
 \* 36.617 ELAPSED SEC.

\*READY:  
 -\* THE NEXT COMMAND ONLY VALIDATES THE RESTRICTANDJOIN COMMAND BY  
 +SHOWING THAT ALL THE RECIPIENTS'S ID NUMBERS ARE INDEED IN FIN-AID.

\* \$ .00, \$4.11T  
 \* 0.004 CPU SEC.  
 \* 1.619 ELAPSED SEC.

\*READY:  
 -RESTRICT IN RECIPIENTS WHERE SIN IS SIN IN FIN-AID.  
 \* 25 RECORDS IN RESULT SET

\* \$ .02, \$4.13T  
 \* 0.042 CPU SEC.  
 \* 2.002 ELAPSED SEC.

\*READY:  
 -\* THE NEXT COMMAND ALSO CREATES COMPOSITE RECORDS BUT NOTICE THAT RECORDS ARE ALSO  
 +CREATED FOR STUDENT AND FINANCIAL AID RECORDS THAT DID NOT MATCH WITH RECORDS  
 +IN THE OTHER SET.

\* \$ .01, \$4.14T  
 \* 0.004 CPU SEC.  
 \* 1.203 ELAPSED SEC.

\*READY:  
 -JOIN STUDENTS BY SIN WITH FIN-AID.  
 \* 94 RECORDS IN RESULT SET

\* \$ .07, \$4.21T  
 \* 0.178 CPU SEC.  
 \* 2.268 ELAPSED SEC.

\*READY:  
 -NAME IT BIGJOIN.

\* \$ .00, \$4.21T  
 \* 0.003 CPU SEC.  
 \* 2.000 ELAPSED SEC.

\*READY:  
-TDOC BIGJOIN.

\*FOR: BIGJOIN  
\*DICTIONARY LOCATION: -MIC#002  
\*DATE: JUN 29, 1977

\*THIS DATASET CANNOT BE DESTROYED.  
\*THIS DATASET CAN BE REPLACED.  
\*THIS DATASET IS SCRAMBLED.  
\*USE COUNT:  
1

\*DATA SET DESCRIPTION:  
\*NO DESCRIPTION EXISTS

| *F(#) | FIELD NAME        | ABBR | VALUE    | TYPE    | LENGTH | DISP | SCALE | FACTOR |
|-------|-------------------|------|----------|---------|--------|------|-------|--------|
| *F(1) | STUDENT-ID-NO     | SIN  | REQUIRED | U       | 4      |      | 0     |        |
| *F(2) | NAME              | NA   | REQUIRED | C       | 24     |      | 4     |        |
| *F(3) | ETHNICITY         | RACE | REQUIRED | UC      | 1      |      | 28    |        |
|       | CATEGORIES (ONLY) |      |          | DEFAULT |        |      |       |        |
|       | UNKNOWN           | UNK  | 0        |         |        |      |       |        |
|       | WHITE             | WH   | 1        |         |        |      |       |        |
|       | BLACK             | BL   | 2        |         |        |      |       |        |
|       | AMER-INDIAN       | AI   | 3        |         |        |      |       |        |
|       | SPANISH           | SP   | 4        |         |        |      |       |        |
|       | ORIENTAL          | OR   | 5        |         |        |      |       |        |
|       | OTHER             | OTH  | 6        |         |        |      |       |        |
| *F(4) | SEX               | SEX  | REQUIRED | CC      | 1      |      | 29    |        |
|       | CATEGORIES (ONLY) |      |          |         |        |      |       |        |
|       | MALE              | MA   | M        |         |        |      |       |        |
|       | FEMALE            | FE   | F        |         |        |      |       |        |
|       | UNKNOWN           | UNK  | U        |         |        |      |       |        |
| *F(5) | BIRTHDATE         | BDAY | REQUIRED | UC      | 3      |      | 30    |        |
|       | CATEGORIES        |      |          |         |        |      |       |        |
|       | MISSING           | MISS | 0        |         |        |      |       |        |
| *F(6) | CLASS-YEAR        | CLYR |          | SC      | 1      |      | 33    |        |
|       | CATEGORIES        |      |          |         |        |      |       |        |
|       | UNKNOWN           | UNK  | -1       |         |        |      |       |        |
|       |                   |      |          | DEFAULT |        |      |       |        |

|       |                   |      |          |    |   |    |
|-------|-------------------|------|----------|----|---|----|
| *F(7) | YEAR-AWARDED      | YEAR | REQUIRED | U  | 1 | 34 |
| *F(8) | TERM-AWARDED      | TERM | REQUIRED | UC | 1 | 35 |
| *     | CATEGORIES (ONLY) |      |          |    |   |    |
| *     | FALL              | F    | 1        |    |   |    |
| *     | WINTER            | W    | 2        |    |   |    |
| *     | SUMMER            | S    | 3        |    |   |    |
| *F(9) | CLASS-LEVEL       | CL   | REQUIRED | UC | 1 | 36 |
| *     | CATEGORIES (ONLY) |      |          |    |   |    |
| *     | FRESHMAN          | FR   | 1        |    |   |    |
| *     | SOPHOMORE         | SO   | 2        |    |   |    |

\*FOR: BIGJOIN

| *F(#)  | FIELD NAME             | ABBR    | VALUE    | TYPE | LENGTH | DISP | SCALE | FACTOR |
|--------|------------------------|---------|----------|------|--------|------|-------|--------|
| *      | CATEGORIES (CONTINUED) |         |          |      |        |      |       |        |
| *      | JUNIOR                 | JR      | 3        |      |        |      |       |        |
| *      | SENIOR                 | SR      | 4        |      |        |      |       |        |
| *F(10) | TYPE-OF-AID            | TYPE    | REQUIRED | UC   | 1      | 37   |       |        |
| *      | CATEGORIES (ONLY)      |         |          |      |        |      |       |        |
| *      | TUITION                | TU      | 1        |      |        |      |       |        |
| *      | ROOM-BOARD             | RB      | 2        |      |        |      |       |        |
| *      | CASH                   | CA      | 3        |      |        |      |       |        |
| *      | LOAN                   | LO      | 4        |      |        |      |       |        |
| *      | WORK-STUDY             | WS      | 5        |      |        |      |       |        |
| *      | OTHER                  | OTH     | 6        |      |        |      |       |        |
| *F(11) | SOURCE-OF-FUNDS        | SOF     | REQUIRED | UC   | 1      | 38   |       |        |
| *      | CATEGORIES (ONLY)      |         |          |      |        |      |       |        |
| *      | UNIVERSITY             | UNIV    | 1        |      |        |      |       |        |
| *      | STATE-GOV              | SGOV    | 2        |      |        |      |       |        |
| *      | FEDERAL-GOV            | FGOV    | 3        |      |        |      |       |        |
| *      | INDUSTRY               | IND     | 4        |      |        |      |       |        |
| *      | FOUNDATION             | FOUN    | 5        |      |        |      |       |        |
| *      | OTHER                  | OTH     | 6        |      |        |      |       |        |
| *F(12) | AMOUNT                 | AMT     | REQUIRED | S    | 4      | 39   | /     | 100    |
| *F(13) | REMARK*                | REMA*   |          | E    | 4      | 43   |       |        |
| *      | FILE NAME              |         |          |      |        |      |       |        |
| *      | ILIR:REMARKS*          |         |          |      |        |      |       |        |
| *      | \$ .05,                | \$4.26T |          |      |        |      |       |        |
| *      | 0.067 CPU SEC.         |         |          |      |        |      |       |        |
| *      | 57.706 ELAPSED SEC.    |         |          |      |        |      |       |        |

\*READY:  
 --\* THE FOLLOWING XTAB SHOWS THAT NUMEROUS RECORDS IN BIGJOIN HAVE ZERO VALUES FOR THOSE STUDENTS WITHOUT  
 +FINANCIAL AID.

\* \$ .01, \$4.27T  
 \* 0.004 CPU SEC.  
 \* 1.181 ELAPSED SEC.

\*READY:  
 -XTAB IN BIGJOIN SOURCE-OF-FUNDS.  
 \* 05 RECORDS IN RESULT SET  
 \* 94 RECORDS REPRESENTED

| * SOURCE-OF-FUNDS    | COUNT | PERCENT |
|----------------------|-------|---------|
| 0                    | 60    | 63.82   |
| UNIVERSITY           | 15    | 15.95   |
| STATE-GOV            | 9     | 9.57    |
| FEDERAL-GOV          | 6     | 6.38    |
| INDUSTRY             | 4     | 4.25    |
| * \$ .04, \$4.31T    |       |         |
| * 0.100 CPU SEC.     |       |         |
| * 7.080 ELAPSED SEC. |       |         |

\*READY:  
 --\* SUPPOSE A STATISTICS PROFESSOR AT INTERCOLLEGIATE U HEARS THAT  
 +THERE HAVE BEEN FIVE MARRIAGES BETWEEN STUDENTS ON CAMPUS. THE  
 +PROF WONDERS HOW MANY POSSIBLE COUPLES THERE ARE ON CAMPUS. HE TYPES:

\* \$ .01, \$4.32T  
 \* 0.004 CPU SEC.  
 \* 1.680 ELAPSED SEC.

\*READY:  
 -FIND IN STUDENTS W SEX IS FEMALE.  
 \* 45 (52.94%) RECORDS FOUND  
 \* 45 RECORDS IN RESULT SET

\* \$ .02, \$4.34T  
 \* 0.044 CPU SEC.  
 \* 2.727 ELAPSED SEC.

\*READY:  
 -NAME IT WOMEN.  
 \* \$ .00, \$4.34T  
 \* 0.003 CPU SEC.  
 \* 1.184 ELAPSED SEC.

\*READY:  
 -FIND IN STUDENTS W SEX IS MALE.  
 \* 39 (45.88%) RECORDS FOUND  
 \* 39 RECORDS IN RESULT SET

\* \$ .02, \$4.36T  
 \* 0.044 CPU SEC.  
 \* 2.854 ELAPSED SEC.

\*READY:  
 -N IT MEN.

\* \$ .00, \$4.37T  
 \* 0.003 CPU SEC.  
 \* 1.143 ELAPSED SEC.

\*READY:  
 -\* THE PROFESSOR COULD GET THE ANSWER BY SIMPLY MULTIPLYING THESE  
 +NUMBERS OF RECORDS TOGETHER BUT HE IS LAZY. SO.....

\* \$ .00, \$4.37T  
 \* 0.004 CPU SEC.  
 \* 1.496 ELAPSED SEC.

\*READY:  
 -JOIN WOMEN WITH MEN.  
 \* 1,755 RECORDS IN RESULT SET

\* \$ .16, \$4.53T  
 \* 0.368 CPU SEC.  
 \* 4.183 ELAPSED SEC.

\*READY:  
 -SET VERTICAL PRINT ON.

\* \$ .00, \$4.54T  
 \* 0.004 CPU SEC.  
 \* 1.247 ELAPSED SEC.

\*READY:  
 -PG=3.

\*STUDENT-ID-No 13222340  
 \*NAME RICHARDSON DEBORAH  
 \*ETHNICITY BLACK  
 \*SEX FEMALE

\*BIRTHDATE 550310  
 \*CLASS-YEAR 77  
 \*STUDENT-ID-NO 95887329  
 \*NAME POND STUART LESLIE  
 \*ETHNICITY WHITE  
 \*SEX MALE  
 \*BIRTHDATE 560528  
 \*CLASS-YEAR 78

\*STUDENT-ID-NO 1322340  
 \*NAME RICHARDSON DEBORAH  
 \*ETHNICITY BLACK  
 \*SEX FEMALE  
 \*BIRTHDATE 550310  
 \*CLASS-YEAR 77  
 \*STUDENT-ID-NO 113591707  
 \*NAME JONES WILLIAM  
 \*ETHNICITY WHITE  
 \*SEX MALE  
 \*BIRTHDATE 551010  
 \*CLASS-YEAR 77

\*STUDENT-ID-NO 1322340  
 \*NAME RICHARDSON DEBORAH  
 \*ETHNICITY BLACK  
 \*SEX FEMALE  
 \*BIRTHDATE 550310  
 \*CLASS-YEAR 77  
 \*STUDENT-ID-NO 12045525  
 \*NAME GRACE KENNETH R  
 \*ETHNICITY WHITE  
 \*SEX MALE  
 \*BIRTHDATE 580418  
 \*CLASS-YEAR 80

\* \$ .03, \$4.57T  
 \* 0.058 CPU SEC.  
 \* 20.343 ELAPSED SEC.

\*READY:  
 -\* USE THE RESTRICT COMMAND TO EXTRACT THOSE RECORDS FROM BIGJOIN FOR THE INDIVIDUALS RECEIVING NO AID.

\* \$ .00, \$4.57T  
 \* 0.004 CPU SEC.  
 \* 1.908 ELAPSED SEC.

\*READY:  
 -RESTRICT IN BIGJOIN WHERE SIN IS SIN IN NO-AID.  
 \* 60 RECORDS IN RESULT SET

\* \$ .03, \$4.61T  
 \* 0.074 CPU SEC.  
 \* 2.331 ELAPSED SEC.



```

READY
-POC=5 IT.

*STUDENT-ID-NO 1322340
*NAME RICHARDSON DEBORAH
*ETHNICITY BLACK
*SEX FEMALE
*BIRTHDATE 550310
*CLASS-YEAR 77
*YEAR-AWARDED 0
*TERM-AWARDED 0
*CLASS-LEVEL 0
*TYPE-OF-AID 0
*SOURCE-OF-FUNDS 0
*AMOUNT .00
REMARK

*STUDENT-ID-NO 9587329
*NAME POND STUART LESLIE
*ETHNICITY WHITE
*SEX MALE
*BIRTHDATE 560528
*CLASS-YEAR 78
*YEAR-AWARDED 0
*TERM-AWARDED 0
*CLASS-LEVEL 0
*TYPE-OF-AID 0
*SOURCE-OF-FUNDS 0
*AMOUNT .00
REMARK

*STUDENT-ID-NO 122045525
*NAME GRACE KENNETH R
*ETHNICITY WHITE
*SEX MALE
*BIRTHDATE 590418
*CLASS-YEAR 30
*YEAR-AWARDED 0
*TERM-AWARDED 0
*CLASS-LEVEL 0
*TYPE-OF-AID 0
*SOURCE-OF-FUNDS 0
*AMOUNT .00
REMARK

*STUDENT-ID-NO 139904799
*NAME BAILEY EDITH
*ETHNICITY BLACK
*SEX FEMALE
*BIRTHDATE 570913
*CLASS-YEAR 77
*YEAR-AWARDED 0
*TERM-AWARDED 0

```

\*CLASS-LEVEL 0  
 \*TYPE-OF-AID 0  
 \*SOURCE-OF-FUNDS 0  
 \*AMOUNT .00  
 \*REMARK\*

\*STUDENT-ID-NO 181703558  
 \*NAME HALE ROBERT SAMUEL  
 \*ETHNICITY WHITE  
 \*SEX MALE  
 \*BIRTHDATE 551011  
 \*CLASS-YEAR 77  
 \*YEAR-AWARDED 0  
 \*TERM-AWARDED 0  
 \*CLASS-LEVEL 0  
 \*TYPE-OF-AID 0  
 \*SOURCE-OF-FUNDS 0  
 \*AMOUNT .00  
 \*REMARK\*

\* \$ .05, \$4.65T  
 \* 0.083 CPU SEC.  
 \* 33.543 ELAPSED SEC.

\*READY:  
 -SET VERTICAL PRINT OFF.

\* \$ .00, \$4.66T  
 \* 0.004 CPU SEC.  
 \* 1.258 ELAPSED SEC.

\*READY:  
 -\* SAVE COMMAND EXAMPLES.

\* \$ .00, \$4.66T  
 \* 0.003 CPU SEC.  
 \* 1.268 ELAPSED SEC.

\*READY:  
 -SAVE BIGJOIN AS JOINED-SET.  
 \*DATA SET "JOINED-SET " IS TO BE SCRAMBLED. PLEASE ENTER KEY.  
 ?test  
 \* [KEY] T E S I  
 \* E3 C5 E2 E3  
 \*DESCRIPTION ?  
 \*join of students and fin-aid.  
 \*CAN THIS DATA SET BE DESTROYED ?  
 \*Y  
 \*CAN THIS DATA SET BE REPLACED ?  
 \*Y

\* \$ .35, \$5.02T  
 \* 0.793 CPU SEC.  
 \* 54.133 ELAPSED SEC.

\*READY:  
-LIST SETS.

| *DATA SET NAME | STATUS |
|----------------|--------|
| *CARS          | DISK   |
| *RESULI        | TEMP*  |
| *MEN           | TEMP*  |
| *WOMEN         | TEMP*  |
| *JOINED-SET    | DISK   |
| *RECIPIENTS    | TEMP*  |
| *GOT-AID       | TEMP*  |
| *NO-AID        | TEMP*  |
| *S             | TEMP*  |
| *EVENYEARS     | TEMP*  |
| *NOT-JR        | TEMP*  |
| *FROSH         | TEMP*  |
| *ART           | DISK   |
| *FIN-AID       | DISK*  |
| *STUDENTS      | DISK*  |
| *PUBLIC.DATA   | DISK   |

\* \$ .01, \$5.03T  
\* 0.018 CPU SEC.  
\* 9.766 ELAPSED SEC.

\*READY:  
-SAVE RECIPIENTS.  
\*DATA SET "RECIPIENTS" " IS TO BE SCRAMBLED. PLEASE ENTER KEY.  
?test  
\*{(KEY) I E S T  
\* DESCRIPTION ?  
\*raj of students and fin-aid.  
\*CAN THIS DATA SET BE DESTROYED ?  
\*Y  
\*CAN THIS DATA SET BE REPLACED ?  
\*Y

\* \$ .21, \$5.24T  
\* 0.453 CPU SEC.  
\* 39.637 ELAPSED SEC.

\*READY:  
-LIST SETS.

| *DATA SET NAME | STATUS |
|----------------|--------|
| *CARS          | DISK   |
| *RESULI        | TEMP*  |
| *MEN           | TEMP*  |
| *WOMEN         | TEMP*  |
| *JOINED-SET    | DISK   |
| *RECIPIENTS    | DISK   |
| *GOT-AID       | TEMP*  |

```

NO-AID TEMP
S TEMP
EVENYEARS TEMP
NOI-JR TEMP
FROSH TEMP
ART DISK
FIN-AID DISK
STUDENTS DISK
*PUBLIC.DATA DISK

* $.01, $5.25T
* 0.018 CPU SEC.
* 9.759 ELAPSED SEC.

```

```

*READY:
-PURGE.

```

```

* $.02, $5.27T
* 0.038 CPU SEC.
* 1.120 ELAPSED SEC.

```

```

*READY:
-LIST SETS.

```

```

*DATA SET NAME STATUS
*CARS DISK
*ART DISK
*FIN-AID DISK
*STUDENTS DISK
*JOINED-SET DISK
*RECIPIENTS DISK

```

```

* $.01, $5.28T
* 0.009 CPU SEC.
* 4.345 ELAPSED SEC.

```

```

*READY:
-* END EXERCISE B-10.

```

```

* $.00, $5.28T
* 0.003 CPU SEC.
* 1.236 ELAPSED SEC.

```

```

*READY:
-*

```

## EXERCISE B-11 - THE STRATIFIED PRINT AND SELECT COMMANDS.

```
* $.00, $5.28T
* 0.003 CPU SEC.
* 3.954 ELAPSED SEC.
```

```
*READY:
-SELECT IN RECIPIENTS YEAR,TYPE,NAME,SOF,AMT.
*DATA SET "RECIPIENTS" IS TO BE UNSCRAMBLED. PLEASE ENTER KEY.
```

```
?test
*[(KEY) T E S I
* E3 C5 E2 E3
* 25 RECORDS IN RESULT SET
```

```
* $.06, $5.34T
* 0.131 CPU SEC.
* 7.028 ELAPSED SEC.
```

```
*READY:
-POST=2.
*
```

| YEAR-AWARDED          | 73              |        |
|-----------------------|-----------------|--------|
| *TYPE-OF-AID          | ROOM-BOARD      |        |
| * NAME                | SOURCE-OF-FUNDS | AMOUNT |
| * FOSTER HAROLD PETER | STATE-GOV       | 125.00 |
| * JONES DOROTHY       | UNIVERSITY      | 500.00 |
| *                     |                 |        |

| YEAR-AWARDED        | 74              |         |
|---------------------|-----------------|---------|
| *TYPE-OF-AID        | TUITION         |         |
| * NAME              | SOURCE-OF-FUNDS | AMOUNT  |
| * BORCHINSKY ETHEL  | UNIVERSITY      | 125.00  |
| * COOK CYNTHIA ANNA | INDUSTRY        | 1050.00 |
| * JONES WILLIAM     | UNIVERSITY      | 1200.00 |
| * SMITH MARY        | UNIVERSITY      | 350.00  |
| * WARD JULIA CAROL  | INDUSTRY        | 150.00  |
| *                   |                 |         |

| YEAR-AWARDED 74 |                 |            |  |
|-----------------|-----------------|------------|--|
| *TYPE-OF-AID    |                 | ROOM-BOARD |  |
| * NAME          | SOURCE-OF-FUNDS | AMOUNT     |  |
| * BRUZA LESLIE  | UNIVERSITY      | 525.00     |  |
| * LEE SUE JEAN  | FEDERAL-GOV     | 485.00     |  |
| *               |                 |            |  |

| YEAR-AWARDED 74  |                 |        |  |
|------------------|-----------------|--------|--|
| *TYPE-OF-AID     |                 | CASH   |  |
| * NAME           | SOURCE-OF-FUNDS | AMOUNT |  |
| * SMOTHERS MAE S | STATE-GOV       | 50.00  |  |
| *                |                 |        |  |

| YEAR-AWARDED 75       |                 |         |  |
|-----------------------|-----------------|---------|--|
| *TYPE-OF-AID          |                 | TUITION |  |
| * NAME                | SOURCE-OF-FUNDS | AMOUNT  |  |
| * FORINO JOSEPH LOUIS | FEDERAL-GOV     | 550.00  |  |
| * TAYLOR VERA M       | UNIVERSITY      | 540.00  |  |
| *                     |                 |         |  |

| YEAR-AWARDED 75    |                 |        |  |
|--------------------|-----------------|--------|--|
| *TYPE-OF-AID       |                 | CASH   |  |
| * NAME             | SOURCE-OF-FUNDS | AMOUNT |  |
| * BORG ELIZABETH M | FEDERAL-GOV     | 129.00 |  |
| * MILLER SUSAN ANN | UNIVERSITY      | 75.00  |  |
| * SCOTT MARY GRACE | INDUSTRY        | 100.00 |  |
| *                  |                 |        |  |

| YEAR-AWARDED       | 75              | WORK-STUDY |
|--------------------|-----------------|------------|
| *TYPE-OF-AID       |                 |            |
| * NAME             | SOURCE-OF-FUNDS | AMOUNT     |
| * CROWFOOT WILLIAM | FEDERAL-GOV     | 400.00     |
| * MENKE SUSAN A    | UNIVERSITY      | 400.00     |
| *                  |                 |            |

| YEAR-AWARDED          | 75              | OTHER  |
|-----------------------|-----------------|--------|
| *TYPE-OF-AID          |                 |        |
| * NAME                | SOURCE-OF-FUNDS | AMOUNT |
| * CROSBY JANE LILLIAN | UNIVERSITY      | 100.00 |
| *                     |                 |        |

| YEAR-AWARDED | 76              | TUITION |
|--------------|-----------------|---------|
| *TYPE-OF-AID |                 |         |
| * NAME       | SOURCE-OF-FUNDS | AMOUNT  |
| * KENT JERRY | STATE-GOV       | 295.00  |
| *            |                 |         |

| YEAR-AWARDED     | 76              | ROOM-BOARD |
|------------------|-----------------|------------|
| *TYPE-OF-AID     |                 |            |
| * NAME           | SOURCE-OF-FUNDS | AMOUNT     |
| * ROSEN LARRY Z  | UNIVERSITY      | 250.00     |
| * WHITE ALBERT L | STATE-GOV       | 280.00     |
| *                |                 |            |

| YEAR-AWARDED     | 76              | WORK-STUDY |
|------------------|-----------------|------------|
| *TYPE-OF-AID     |                 |            |
| * NAME           | SOURCE-OF-FUNDS | AMOUNT     |
| * WONG DAVID LEE | STATE-GOV       | 150.00     |
| *                |                 |            |

| YEAR-AWARDED            | 76              | OTHER  |
|-------------------------|-----------------|--------|
| *TYPE-OF-AID            |                 |        |
| * NAME                  | SOURCE-OF-FUNDS | AMOUNT |
| * HERNANDEZ RAFAEL JOSE | UNIVERSITY      | 200.00 |
| *                       |                 |        |

| YEAR-AWARDED       | 77              | ROOM-BOARD |
|--------------------|-----------------|------------|
| *TYPE-OF-AID       |                 |            |
| * NAME             | SOURCE-OF-FUNDS | AMOUNT     |
| * BATES LESLIE ANN | UNIVERSITY      | 175.00     |
| *                  |                 |            |

| YEAR-AWARDED            | 77              | LOAN   |
|-------------------------|-----------------|--------|
| *TYPE-OF-AID            |                 |        |
| * NAME                  | SOURCE-OF-FUNDS | AMOUNT |
| * NEUMANN CURTIS ARTHUR | UNIVERSITY      | 150.00 |
| *                       |                 |        |
| * \$ .07, \$5.41T       |                 |        |
| * 0.083 CPU SEC.        |                 |        |
| * 93.361 ELAPSED SEC.   |                 |        |

\*READY:  
 --\* END EXERCISE B-11  
 \*\*



EXERCISE 3-12 - FURTHER DEMONSTRATE NAME (RENAME) COMMAND IN ADDITION TO THE TECHDOC COMMAND.

\* \$ .00, \$5.42T  
 \* 0.004 CPU SEC.  
 \* 4.251 ELAPSED SEC.

\*READY:  
 -TECHDOC IN FIN-AID SOF.

| *F(#) | FIELD NAME          | ABBR    | VALUE    | TYPE | LENGTH | DISP | SCALE | FACTOR |
|-------|---------------------|---------|----------|------|--------|------|-------|--------|
| *F(1) | SOURCE-OF-FUNDS     | SOF     | REQUIRED | UC   | 1      | 9    |       |        |
| *     | CATEGORIES (ONLY)   |         |          |      |        |      |       |        |
| *     | UNIVERSITY          | UNIV    | 1        |      |        |      |       |        |
| *     | STATE-GOV           | SGOV    | 2        |      |        |      |       |        |
| *     | FEDERAL-GOV         | FGOV    | 3        |      |        |      |       |        |
| *     | INDUSTRY            | IND     | 4        |      |        |      |       |        |
| *     | FOUNDATION          | FOUN    | 5        |      |        |      |       |        |
| *     | OTHER               | OTH     | 6        |      |        |      |       |        |
| *     | \$ .02,             | \$5.44T |          |      |        |      |       |        |
| *     | 0.034 CPU SEC.      |         |          |      |        |      |       |        |
| *     | 10.365 ELAPSED SEC. |         |          |      |        |      |       |        |

\*READY:  
 -RENAME IN FIN-AID SOURCE-OF-FUNDS TO SOURCE-OF-AID.

\* \$ .00, \$5.44T  
 \* 0.003 CPU SEC.  
 \* 1.483 ELAPSED SEC.

\*READY:  
 -RENAME IN FIN-AID SOF ABBR TO SOA.

\* \$ .00, \$5.44T  
 \* 0.003 CPU SEC.  
 \* 1.348 ELAPSED SEC.

\*READY:  
 -IDOC IN FIN-AID SOA.

| *F(*) | FIELD NAME                      | ABBR         | VALUE    | TYPE | LENGTH | DISP | SCALE | FACTOR |
|-------|---------------------------------|--------------|----------|------|--------|------|-------|--------|
| *F(1) | SOURCE-OF-AID CATEGORIES (ONLY) | SOA          | REQUIRED | UC   | 1      |      |       | 8      |
| *     | UNIVERSITY                      | UNIV         | 1        |      |        |      |       |        |
| *     | STATE-GOV                       | SGOV         | 2        |      |        |      |       |        |
| *     | FEDERAL-GOV                     | FGOV         | 3        |      |        |      |       |        |
| *     | INDUSTRY                        | IND          | 4        |      |        |      |       |        |
| *     | FOUNDATION                      | FOUND        | 5        |      |        |      |       |        |
| *     | OTHER                           | OTH          | 6        |      |        |      |       |        |
| *     |                                 |              |          |      |        |      |       |        |
| *     | \$ .01,                         | \$5.45T      |          |      |        |      |       |        |
| *     | 0.015                           | CP'I SEC.    |          |      |        |      |       |        |
| *     | 10.341                          | ELAPSED SEC. |          |      |        |      |       |        |

\*READY:  
--\* THESE TWO CHANGES AFFECT ONLY THE CORE COPY (BUT NOT THE PERMANENT DISK COPY) OF THE DATA SET.  
\*\* END EXERCISE B-12.

```

* $0.01, $5.46T
* 0.004 CPU SEC.
* 1.233 ELAPSED SEC.

```

\*READY:  
-\*

## EXERCISE B-13 - THE CHANGE, COMPUTE, AND UPDATE COMMANDS.

\* \$ .00, \$5.47I  
 \* 0.003 CPU SEC.  
 \* 3.961 ELAPSED SEC.

\*READY:  
 -CHANGE IN RECIPIENTS WHERE SIN IS 906922815 TO 234567890.  
 \* 01 (4.00%) RECORDS CHANGED  
 \* 25 RECORDS IN RESULT SET

\* \$ .04, \$5.51I  
 \* 0.098 CPU SEC.  
 \* 2.592 ELAPSED SEC.

\*READY:  
 -CHANGE IN IT SO THAT YEAR=77.  
 \* 25 (100.00%) RECORDS CHANGED  
 \* 25 RECORDS IN RESULT SET

\* \$ .02, \$5.53I  
 \* 0.034 CPU SEC.  
 \* 2.375 ELAPSED SEC.

\*READY:  
 -NAME IT 4.

\* \$ .00, \$5.53I  
 \* 0.003 CPU SEC.  
 \* 1.154 ELAPSED SEC.

\*READY:  
 -FIND IN 4 WHERE SIN IS 234567890.  
 \* 01 (4.00%) RECORDS FOUND  
 \* 01 RECORDS IN RESULT SET

\* \$ .02, \$5.55I  
 \* 0.036 CPU SEC.  
 \* 2.375 ELAPSED SEC.

\*READY:  
 -P3V IT.

\*STUDENT-ID-NO 234567890  
 \*YEAR-AWARDED 77  
 \*TERM-AWARDED FALL  
 \*CLASS-LEVEL JUNIOR  
 \*TYPE-OF-AID ROOM-BOARD  
 \*SOURCE-OF-FUNDS STATE-GOV  
 \*AMOUNT 125.00

\*REMARK\*  
 \*NAME FOSTER HAROLD PETER  
 \*ETHNICITY OTHER  
 \*SEX MALE  
 \*BIRTHDATE 561130  
 \*CLASS-YEAR 78  
 \* \$ .02, \$5.57T  
 \* 0.032 CPU SEC.  
 \* 7.873 ELAPSED SEC.

\*READY:  
 -RELEASE 4.

\* \$ .00, \$5.57T  
 \* 0.005 CPU SEC.  
 \* 1.148 ELAPSED SEC.

\*READY:  
 -COMPUTE IN RECIPIENTS ST NEW.AMOUNT=AMOUNT \* 1.05.  
 \* 25 RECORDS IN RESULT SET

\* \$ .07, \$5.65T  
 \* 0.170 CPU SEC.  
 \* 2.667 ELAPSED SEC.

\*READY:  
 -COMPUTE IN IT ST INCREASE=NEW.AMOUNT - AMOUNT.  
 \* 25 RECORDS IN RESULT SET

\* \$ .02, \$5.66T  
 \* 0.035 CPU SEC.  
 \* 1.965 ELAPSED SEC.

\*READY:  
 -COMPUTE IN IT NEWCODE=-447.  
 \* 25 RECORDS IN RESULT SET

\* \$ .01, \$5.67T  
 \* 0.020 CPU SEC.  
 \* 1.812 ELAPSED SEC.

\*READY:  
 -DATESTAMP IT.  
 \* 25 RECORDS IN RESULT SET

\* \$ .02, \$5.69T  
 \* 0.033 CPU SEC.  
 \* 1.696 ELAPSED SEC.

\*READY:  
-PC=5 IN IT AMOUNT, INCREASE, NEW.AMOUNT, NEWCODE, DATESTAMP.

| * | AMOUNT  | INCREASE | NEW.AMOUNT | NEWCODE | DATESTAMP |
|---|---------|----------|------------|---------|-----------|
| * | 1200.00 | 60.0000  | 1260.0000  | -447    | 770629    |
| * | 495.00  | 24.2500  | 509.2500   | -447    | 770629    |
| * | 1050.00 | 52.5000  | 1102.5000  | -447    | 770629    |
| * | 100.00  | 5.0000   | 105.0000   | -447    | 770629    |
| * | 100.00  | 5.0000   | 105.0000   | -447    | 770629    |

\* \$01, \$5.70T  
\* 0.017 CPU SEC.  
\* 7.553 ELAPSED SEC.

\*READY:  
-\* TO SIMPLIFY THE UPDATE EXAMPLE, A UNIQUE KEY WILL BE CREATED  
- THEN THE FIRST 5 RECORDS WILL RECEIVE NEW STUDENT-ID NUMBERS;  
+ THE EFFECT OF ERRONEOUSLY SPECIFYING A KEY VALUE IS ALSO SHOWN.

\* \$01, \$5.71T  
\* 0.004 CPU SEC.  
\* 1.584 ELAPSED SEC.

\*READY:  
-KEY STUDENTS.  
\*DATA SET "STUDENTS " IS TO BE UNSCRAMBLED. PLEASE ENTER KEY.

?test  
\* [KEY] I E S I  
E3 C5 E2 E3  
\* 85 RECORDS IN RESULT SET

\* \$05, \$5.76T  
\* 0.117 CPU SEC.  
\* 8.842 ELAPSED SEC.

\*READY:  
-RESET INPUT.

\* \$00, \$5.77T  
\* 0.003 CPU SEC.  
\* 1.167 ELAPSED SEC.

\*READY:  
-update it.  
-UPDATE IT.  
ENTER NAMES OF FIELDS USED TO IDENTIFY RECORDS.  
?key  
ENTER NAMES OF FIELDS TO BE UPDATED OR "STOP".  
?sin

%ABBREVIATED PROMPTING ?

%no : 1  
%KEY : 111  
%STUDENT-ID-NO : 111

%CORRECTIONS:  
%  
%KEY : 2  
%STUDENT-ID-NO : 222

%CORRECTIONS:  
%  
%KEY : 3  
%STUDENT-ID-NO : 333

%CORRECTIONS:  
%  
%KEY : 4  
%STUDENT-ID-NO : 444

%CORRECTIONS:  
%5

%CONCERNING: 5  
%FIELD MUST BE SET OFF BY "=" OR ":" OR "<" OR ">" OR "<".

%CORRECTIONS:  
%  
%KEY : 5  
%STUDENT-ID-NO : 555

%CORRECTIONS:  
%  
%KEY : 999  
%STUDENT-ID-NO : 999

%CORRECTIONS:  
%stop  
% 6 RECORDS ARE BEING ADDED.

RECORDS WITH THE FOLLOWING IDENTIFYING VALUES COULD NOT BE FOUND.

3 KEY  
3 -----  
3 999  
3 ENTER NAMES OF FIELDS TO BE UPDATED OR "STOP".  
3 stop  
3 85 RECORDS IN RESULT SET

\* \$.28, \$6.05I  
\* 0.525 CPU SEC.  
\* 122.729 ELAPSED SEC.

\*READY:

-- the input is again set to be read from a file.  
 -- THE INPUT IS AGAIN SET TO BE READ FROM A FILE.

\* \$ .02, \$6.07T  
 \* 0.004 CPU SEC.  
 \* 1.380 ELAPSED SEC.

\*READY:

--set input=cmts(381).  
 --SET INPUT=CWDS(381).

\* \$ .01, \$6.08T  
 \* 0.006 CPU SEC.  
 \* 1.114 ELAPSED SEC.

\*READY:

--P@C=5.

| * | KEY | STUDENT-ID-NO | NAME               | ETHNICITY | SEX    | BIRTHDATE | CLASS-YEAR |
|---|-----|---------------|--------------------|-----------|--------|-----------|------------|
| * | 1   | 111           | RICHARDSON DEBORAH | BLACK     | FEMALE | 550310    | 77         |
| * | 2   | 222           | POND STUART LESLIE | WHITE     | MALE   | 560528    | 78         |
| * | 3   | 333           | JONES WILLIAM      | WHITE     | MALE   | 551010    | 77         |
| * | 4   | 444           | GRACE KENNETH R    | WHITE     | MALE   | 580418    | 80         |
| * | 5   | 555           | LEE SUE JEAN       | ORIENTAL  | FEMALE | 560716    | 79         |

\* \$ .01, \$6.10T  
 \* 0.019 CPU SEC.  
 \* 9.412 ELAPSED SEC.

\*READY:

-- END EXERCISE B-13.

\* \$ .01, \$6.10T  
 \* 0.003 CPU SEC.  
 \* 1.232 ELAPSED SEC.

\*READY:

--

EXERCISE B-14 - THE WRITE FOR ANALYSIS COMMAND; ALSO KEY AND SELECT COMMANDS.

```
* $.00, $6.10T
* 0.004 CPU SEC.
* 4.111 ELAPSED SEC.
```

```
*READY:
-KEY RECIPIENTS.
* 25 RECORDS IN RESULT SET

* $.01, $6.12T
* 0.021 CPU SEC.
* 1.710 ELAPSED SEC.
```

```
*READY:
-SEL IN IT ALL BUT NAME AND SIN.
* 25 RECORDS IN RESULT SET

* $.02, $6.14T
* 0.037 CPU SEC.
* 1.828 ELAPSED SEC.
```

```
*READY:
-NAME RESULT MIDAS.

* $.00, $6.14T
* 0.003 CPU SEC.
* 1.201 ELAPSED SEC.
```

```
*READY:
-WRITE FOR ANALYSIS MIDAS.

* $.03, $6.22T
* 0.172 CPU SEC.
* 3.926 ELAPSED SEC.
```

```
*READY:
-* END EXERCISE B-14.

* $.00, $6.22T
* 0.003 CPU SEC.
* 1.235 ELAPSED SEC.
```

```
*READY:
-*
```



EXERCISE B-15 - DEMONSTRATE THE CALL COMMAND. \*

\*\* THE FIRST CALL IS WITHOUT A DATA SET BEING PASSED TO THE SUBROUTINE; PARAMETERS ARE PASSED.

\* \$\$.01, \$6.23T

\* 0.004 CPU SEC.

\* 1.809 ELAPSED SEC.

\*READY:

-CALL TIME WITHOUT DATASET PAR=5.1.

\*JUN 29, 1977

\* \$.08, \$6.31T

\* 0.168 CPU SEC.

\* 2.605 ELAPSED SEC.

\*READY:

-\* THE SECOND CALL CAUSES MICRO TO PASS DATA TO THE SUBROUTINE. THE SOURCE CODE FOR THE SUBROUTINE APPEARS

+ IN APPENDIX D. THE PRECEDING FIND COMMAND RESTRICTS THE NUMBER OF RECORDS PASSED TO THE SUBROUTINE.

\* \$.01, \$6.31T

\* 0.004 CPU SEC.

\* 1.915 ELAPSED SEC.

\*READY:

-FIND IN RECIPIENTS WHERE AMT >= 500.00.

\* 06 (24.00%) RECORDS FOUND

\* 06 RECORDS IN RESULT SET

\* \$.02, \$6.33T

\* 0.037 CPU SEC.

\* 2.525 ELAPSED SEC.

\*READY:

-NAME IT GE500.

\* \$.00, \$6.34T

\* 0.003 CPU SEC.

\* 1.193 ELAPSED SEC.

\*READY:

-CALL CALLEX USING GE500 LIB=ILIR;DEMOLIB KEEP=NO PAR=1,2.05,STRING.

\* IFINFO AND ICINFO TABLES

\*

FIELD NAME = STUDENT-ID-NO  
 CALLEX FIELD NUMBER = 1

INDEX INTO IFINFO = 1  
 INDEX INTO ICINFO = 0  
 NUMBER OF CATEGORIES = 0

NO CATEGORIES FOR THIS FIELD

FIELD NAME = NAME  
 CALLEX FIELD NUMBER = 2

INDEX INTO IFINFO = 9  
 INDEX INTO ICINFO = 0  
 NUMBER OF CATEGORIES = 0

NO CATEGORIES FOR THIS FIELD

FIELD NAME = ETHNICITY  
 CALLEX FIELD NUMBER = 3

INDEX INTO IFINFO = 15  
 INDEX INTO ICINFO = 21  
 NUMBER OF CATEGORIES = 7

| CATEGORIES: | ICINFO INDEX | NAME        | VALUE |
|-------------|--------------|-------------|-------|
|             | 21           | UNKNOWN     | 0     |
|             | 22           | WHITE       | 1     |
|             | 23           | BLACK       | 2     |
|             | 24           | AMER-INDIAN | 3     |
|             | 25           | SPANISH     | 4     |
|             | 26           | ORIENTAL    | 5     |
|             | 27           | OTHER       | 6     |

FIELD NAME = SEX  
 CALLEX FIELD NUMBER = 4

INDEX INTO IFINFO = 16  
 INDEX INTO ICINFO = 28  
 NUMBER OF CATEGORIES = 3

|   |                              |         |  |  |       |
|---|------------------------------|---------|--|--|-------|
| ★ | CATEGORIES:                  |         |  |  | VALUE |
| ★ | ICINFO INDEX                 | NAME    |  |  | M     |
| ★ | 28                           | MALE    |  |  |       |
| ★ | 29                           | FEMALE  |  |  | F     |
| ★ | 30                           | UNKNOWN |  |  | U     |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | FIELD NAME = BIRTHDATE       |         |  |  |       |
| ★ | CALLEX FIELD NUMBER =        | 5       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | INDEX INTO IFINFO =          | 17      |  |  |       |
| ★ | INDEX INTO ICINFO =          | 31      |  |  |       |
| ★ | NUMBER OF CATEGORIES =       | 1       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | CATEGORIES:                  |         |  |  | VALUE |
| ★ | ICINFO INDEX                 | NAME    |  |  | 0     |
| ★ | 31                           | MISSING |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | FIELD NAME = CLASS-YEAR      |         |  |  |       |
| ★ | CALLEX FIELD NUMBER =        | 6       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | INDEX INTO IFINFO =          | 18      |  |  |       |
| ★ | INDEX INTO ICINFO =          | 32      |  |  |       |
| ★ | NUMBER OF CATEGORIES =       | 1       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | CATEGORIES:                  |         |  |  | VALUE |
| ★ | ICINFO INDEX                 | NAME    |  |  | -1    |
| ★ | 32                           | UNKNOWN |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | FIELD NAME = YEAR-AWARDED    |         |  |  |       |
| ★ | CALLEX FIELD NUMBER =        | 7       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | INDEX INTO IFINFO =          | 2       |  |  |       |
| ★ | INDEX INTO ICINFO =          | 0       |  |  |       |
| ★ | NUMBER OF CATEGORIES =       | 0       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | NO CATEGORIES FOR THIS FIELD |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | FIELD NAME = TERM-AWARDED    |         |  |  |       |
| ★ | CALLEX FIELD NUMBER =        | 3       |  |  |       |
| ★ |                              |         |  |  |       |
| ★ | INDEX INTO IFINFO =          | 3       |  |  |       |
| ★ | INDEX INTO ICINFO =          | 1       |  |  |       |
| ★ | NUMBER OF CATEGORIES =       | 13      |  |  |       |

\* \* \* CATEGORIES:  
 \* \* \* ICINFO INDEX NAME VALUE  
 \* \* \* 1 FALL 1  
 \* \* \* 2 WINTER 2  
 \* \* \* 3 SUMMER 3

\* \* \* FIELD NAME = CLASS-LEVEL  
 \* \* \* CALLEX FIELD NUMBER = 9

\* \* \* INDEX INTO IFINFO = 4  
 \* \* \* INDEX INTO ICINFO = 4  
 \* \* \* NUMBER OF CATEGORIES = 4

\* \* \* CATEGORIES:  
 \* \* \* ICINFO INDEX NAME VALUE  
 \* \* \* 4 FRESHMAN 1  
 \* \* \* 5 SOPHOMORE 2  
 \* \* \* 6 JUNIOR 3  
 \* \* \* 7 SENIOR 4

\* \* \* FIELD NAME = TYPE-OF-AID  
 \* \* \* CALLEX FIELD NUMBER = 10

\* \* \* INDEX INTO IFINFO = 5  
 \* \* \* INDEX INTO ICINFO = 8  
 \* \* \* NUMBER OF CATEGORIES = 6

\* \* \* CATEGORIES:  
 \* \* \* ICINFO INDEX NAME VALUE  
 \* \* \* 8 TUITION 1  
 \* \* \* 9 ROOM-BOARD 2  
 \* \* \* 10 CASH 3  
 \* \* \* 11 LOAN 4  
 \* \* \* 12 WORK-STUDY 5  
 \* \* \* 13 OTHER 6

\* \* \* FIELD NAME = SOURCE-OF-FUNDS  
 \* \* \* CALLEX FIELD NUMBER = 11

\* \* \* INDEX INTO IFINFO = 6  
 \* \* \* INDEX INTO ICINFO = 14  
 \* \* \* NUMBER OF CATEGORIES = 6

\* CATEGORIES\*  
 \* ICINFO INDEX NAME VALUE  
 \* 14 UNIVERSITY 1  
 \* 15 STATE-GOV 2  
 \* 16 FEDERAL-GOV 3  
 \* 17 INDUSTRY 4  
 \* 18 FOUNDATION 5  
 \* 19 OTHER 6

\* FIELD NAME = AMOUNT  
 \* CALLEX FIELD NUMBER = 12

\* INDEX INTO IFINFO = 7  
 \* INDEX INTO ICINFO = 0  
 \* NUMBER OF CATEGORIES = 0

\* NO CATEGORIES FOR THIS FIELD

\* FIELD NAME = REMARK\*  
 \* CALLEX FIELD NUMBER = 13

\* INDEX INTO IFINFO = 8  
 \* INDEX INTO ICINFO = 20  
 \* NUMBER OF CATEGORIES = 1

\* EXTERNAL -- FILE = ILIR:REMARKS\*

## IDATA MATRIX

\* RECORD 1  
 \* STUDENT-ID-NO 113591707  
 \* NAME JONES WILLIAM  
 \* ETHNICITY WHITE  
 \* SEX MALE  
 \* BIRTHDATE 551010  
 \* CLASS-YEAR 77  
 \* YEAR-AWARDED 74  
 \* TERM-AWARDED FALL  
 \* CLASS-LEVEL SENIOR  
 \* TYPE-OF-AID TUITION  
 \* SOURCE-OF-FUNDS UNIVERSITY  
 \* AMOUNT 120000  
 \* REMARK\*

| RECORD    | STUDENT-ID-NO | NAME  | ETHNICITY | SEX    | BIRTHDATE | CLASS-YEAR | YEAR-AWARDED | TERM-AWARDED | CLASS-LEVEL | TYPE-OF-AID | SOURCE-OF-FUNDS | AMOUNT | REMARK* |
|-----------|---------------|-------|-----------|--------|-----------|------------|--------------|--------------|-------------|-------------|-----------------|--------|---------|
| 142668782 | COOK CYNTHIA  | ANNA  | BLACK     | FEMALE | 560908    | 78         | FALL         | JUNIOR       | TUITION     | INDUSTRY    | 105000          |        |         |
| 179410240 | TAYLOR VERA   | BLACK | FEMALE    | 501004 | 78        | 75         | SUMMER       | JUNIOR       | TUITION     | UNIVERSITY  | 54000           |        |         |
| 251760359 | JONES DOROTHY | WHITE | FEMALE    | 550314 | 77        | 73         | FALL         | SENIOR       | ROOM-BOARD  | UNIVERSITY  | 50000           |        |         |

```

* RECORD 5
* STUDENT-ID-NO 409142875
* NAME FORINO JOSEPH LOUIS
* ETHNICITY WHITE VALUE = 1
* SEX MALE VALUE = M
* BIRTHDATE 560907
* CLASS-YEAR 78
* YEAR-AWARDED 75
* TERM-AWARDED WINTER VALUE = 2
* CLASS-LEVEL JUNIOR VALUE = 3
* TYPE-OF-AID TUITION VALUE = 1
* SOURCE-OF-FUNDS FEDERAL-GOV VALUE = 3
* AMOUNT 55000
* REMARK*
*

* RECORD 6
* STUDENT-ID-NO 686850331
* NAME BRUZA LESLIE
* ETHNICITY SPANISH VALUE = 4
* SEX FEMALE VALUE = F
* BIRTHDATE 550511
* CLASS-YEAR 77
* YEAR-AWARDED 74
* TERM-AWARDED WINTER VALUE = 2
* CLASS-LEVEL SENIOR VALUE = 4
* TYPE-OF-AID ROOM-BOARD VALUE = 2
* SOURCE-OF-FUNDS UNIVERSITY VALUE = 1
* AMOUNT 52500
* REMARK*
*

* 3 PARAMETERS
* INTGER = 1 FLIPT = 2.05 STRING = STRING
*
* $.19, $6.53T
* 0.321 CPU SEC.
* 158.131 ELAPSED SEC.

*READY:
--* END EXERCISE B-15.

* $.01, $6.54T
* 0.003 CPU SEC.
* 1.229 ELAPSED SEC.

*READY:
--*

```

EXERCISE B-16 - PUTTING A SET IN REVERSE ORDER. \*  
 \*(REVERSE PRINTING WILL MAKE THIS UNNECESSARY FOR MOST APPLICATIONS.)  
 +THE FUNCTION TO BE USED IS: (KEY - (NUMBER OF RECORDS + 1)) \* -1 .

\* \$.01, \$6.54T  
 \* 0.004 CPU SEC.  
 \* 1.622 ELAPSED SEC.

\*READY:  
 -KEY STUDENTS.  
 \* 85 RECORDS IN RESULT SET

\* \$.01, \$6.56T  
 \* 0.026 CPU SEC.  
 \* 1.776 ELAPSED SEC.

\*READY:  
 -COMPUTE IN IT SUCH THAT KEY = KEY - 85 \* -1.  
 \* 85 RECORDS IN RESULT SET

\* \$.03, \$6.58T  
 \* 0.056 CPU SEC.  
 \* 1.949 ELAPSED SEC.

\*READY:  
 -PQC=5 STUDENTS.

| STUDENT-ID-NO | NAME               | ETHNICITY | SEX    | BIRTHDATE | CLASS-YEAR |
|---------------|--------------------|-----------|--------|-----------|------------|
| 13222340      | RICHARDSON DEBORAH | BLACK     | FEMALE | 550310    | 77         |
| 95887328      | POND STUART LESLIE | WHITE     | MALE   | 560528    | 78         |
| 113591707     | JONES WILLIAM      | WHITE     | MALE   | 551010    | 77         |
| 122045525     | GRACE KENNETH R    | WHITE     | MALE   | 580418    | 80         |
| 133680736     | LEE SUE JEAN       | ORIENTAL  | FEMALE | 560716    | 78         |

\* \$.01, \$6.60T  
 \* 0.017 CPU SEC.  
 \* 8.938 ELAPSED SEC.

\*READY:  
 -PQC=5 RESULT.

| KEY | STUDENT-ID-NO | NAME                | ETHNICITY | SEX  | BIRTHDATE | CLASS-YEAR |
|-----|---------------|---------------------|-----------|------|-----------|------------|
| 0   | 967397398     | WENDEZ JOSE ROBERTO | SPANISH   | MALE | 560708    | 78         |
| 1   | 949705539     | WHITE ALBERT L      | WHITE     | MALE | 590911    | 80         |
| 2   | 921293935     | ARRILLAGA PAUL      | OTHER     | MALE | 560225    | 78         |
| 3   | 907175979     | SCHMIDT RAYMOND F   | WHITE     | MALE | 580505    | 80         |
| 4   | 906922815     | FOSTER HAROLD PETER | OTHER     | MALE | 561130    | 78         |

\* \$.01, \$6.61T  
 \* 0.017 CPU SEC.  
 \* 9.521 ELAPSED SEC.



```
*READY:
-* END EXERCISE B-16.
* $00, $6.61T
* 0.003 CPU SEC.
* 1.232 ELAPSED SEC.

*READY:
-*
```

EXERCISE B-17 - ADDING AN EXTRA FIELD TO AN EXISTING SET. \*

\*THE SELECT COMMAND IS USED TO TWICE SELECT A FIELD OF THE DESIRED WIDTH.

\* \$ .01, \$6.62T  
\* 0.004 CPU SEC.  
\* 1.669 ELAPSED SEC.

\*READY:

\*-SELECT IN STUDENTS SIN,NA,RACE,SEX,BDAY,CLYR,BDAY.

\* 35 RECORDS IN RESULT SET

\* \$ .02, \$6.64T  
\* 0.045 CPU SEC.  
\* 1.997 ELAPSED SEC.

\*READY:

\*-\* THE FIRST BIRTHDATE FIELD MUST BE RENAMED IN ORDER TO GAIN ACCESS TO THE SECOND ONE.

\* \$ .00, \$6.65T  
\* 0.003 CPU SEC.  
\* 1.777 ELAPSED SEC.

\*READY:

\*-RENAME IN IT BIRTHDATE TO BIRTHDATE1.

\* \$ .00, \$6.65T  
\* 0.003 CPU SEC.  
\* 1.374 ELAPSED SEC.

\*READY:

\*-\* NOW THE SECOND BIRTHDATE FIELD CAN BE RENAMED AND THE FIRST FIELD CAN BE RESTORED.

\* \$ .00, \$6.65T  
\* 0.003 CPU SEC.  
\* 1.758 ELAPSED SEC.

\*READY:

\*-RENAME IN IT BIRTHDATE TO NEWFIELD.

\* \$ .00, \$6.66T  
\* 0.004 CPU SEC.  
\* 1.360 ELAPSED SEC.

\*READY:

\*-RENAME IN IT NEWFIELD ABBR TO NEW.

\* \$ .00, \$6.66T  
\* 0.004 CPU SEC.  
\* 1.351 ELAPSED SEC.

\*READY:  
-RENAME IN IT BDAY TO BIRTHDATE.

\* \$ .00, \$6.67T  
\* 0.003 CPU SEC.  
\* 1.323 ELAPSED SEC.

\*READY:  
-IDOC IN IT \*.

| *F( # )               | FIELD NAME    | ABBR | VALUE    | TYPE | LENGTH | DISP | SCALE | FACTOR |
|-----------------------|---------------|------|----------|------|--------|------|-------|--------|
| *F(1)                 | STUDENT-ID-NO | SIN  | REQUIRED | U    | 4      | 0    |       |        |
| *F(2)                 | NAME          | NA   | REQUIRED | C    | 24     | 4    |       |        |
| *F(3)                 | ETHNICITY     | RACE | REQUIRED | UC   | 1      | 28   |       |        |
| *F(4)                 | SEX           | SEX  | REQUIRED | CC   | 1      | 29   |       |        |
| *F(5)                 | BIRTHDATE     | BDAY | REQUIRED | UC   | 3      | 30   |       |        |
| *F(6)                 | CLASS-YEAR    | CLYR |          | SC   | 1      | 33   |       |        |
| *F(7)                 | NEWFIELD      | NEW  | REQUIRED | UC   | 3      | 34   |       |        |
| * \$ .01, \$6.68T     |               |      |          |      |        |      |       |        |
| * 0.014 CPU SEC.      |               |      |          |      |        |      |       |        |
| * 12.450 ELAPSED SEC. |               |      |          |      |        |      |       |        |

\*READY:  
-\* THE NEW FIELD IS READY TO BE USED. NOW COMPUTE, CHANGE OR UPDATE CAN  
+BE USED TO GIVE THIS FIELD THE PROPER VALUES. \*  
+END EXERCISE B-17.

\* \$ .01, \$6.69T  
\* 0.004 CPU SEC.  
\* 1.218 ELAPSED SEC.

\*READY:  
-\*

## EXERCISE B-18 - THE PLOT COMMAND.

\* \$.00, \$.69T  
 \* 0.003 CPU SEC.  
 \* 3.765 ELAPSED SEC.

\*READY:

--\* THE PLOT COMMAND DOES NOT YET HANDLE SCALED FIELDS. IT PLOTS THE ACTUAL INTEGER VALUES  
 \* UNSCALED. IN ORDER TO PLOT VALUES OF THE SAME INTEGER SCALE, TWO COMPUTE COMMANDS  
 \* FOLLOW THE XTAB COMMAND WHICH GENERATES THE DATA FOR THE PLOT.

\* \$.01, \$.670T  
 \* 0.004 CPU SEC.  
 \* 1.592 ELAPSED SEC.

\*READY:

-XTAB IN RECIPIENTS AVE.AMOUNT, MIN.AMOUNT, MAX.AMOUNT, YEAR.  
 \* 05 RECORDS IN RESULT SET  
 \* 25 RECORDS REPRESENTED

| * YEAR-AWARDED | AVE.AMOUNT | MIN.AMOUNT | MAX.AMOUNT | COUNT | PERCENT |
|----------------|------------|------------|------------|-------|---------|
| 73             | 312.5000   | 125.00     | 500.00     | 2     | 8.00    |
| 74             | 491.8750   | 50.00      | 1200.00    | 8     | 32.00   |
| 75             | 286.6250   | 75.00      | 550.00     | 8     | 32.00   |
| 76             | 235.0000   | 150.00     | 295.00     | 5     | 20.00   |
| 77             | 162.5000   | 150.00     | 175.00     | 2     | 8.00    |

\* \$.06, \$.676T  
 \* 0.121 CPU SEC.  
 \* 9.306 ELAPSED SEC.

\*READY:

-COMPUTE IN IT MIN.AMOUNT=MIN.AMOUNT \* 1.00.  
 \* 05 RECORDS IN RESULT SET

\* \$.01, \$.677T  
 \* 0.019 CPU SEC.  
 \* 1.938 ELAPSED SEC.

\*READY:

-COMPUTE IN IT MAX.AMOUNT = MAX.AMOUNT \* 1.00.  
 \* 05 RECORDS IN RESULT SET

\* \$.01, \$.678T  
 \* 0.019 CPU SEC.  
 \* 1.963 ELAPSED SEC.

\*READY:

-P.

| * YEAR-AWARDED | AVE.AMOUNT | MIN.AMOUNT | MAX.AMOUNT | COUNT | PERCENT |
|----------------|------------|------------|------------|-------|---------|
| 73             | 312.5000   | 125.0000   | 500.0000   | 2     | 8.00    |
| 74             | 491.8750   | 50.0000    | 1200.0000  | 8     | 32.00   |
| 75             | 286.6250   | 75.0000    | 550.0000   | 8     | 32.00   |
| 76             | 235.0000   | 150.0000   | 295.0000   | 3     | 20.00   |
| 77             | 162.5000   | 150.0000   | 175.0000   | 2     | 8.00    |

\* \$ .01, \$6.79T  
 \* 0.017 CPU SEC.  
 \* 7.858 ELAPSED SEC.

\*READY:  
 -PLOT IN RESULT AVE.AMOUNT, MIN.AMOUNT,MAX.AMOUNT AGAINST YEAR ON PLOTOUT.

\*PDS: PLOT DESCRIPTION GENERATION BEGINS \*\*\*

\* \$ .29, \$7.08T  
 \* 0.616 CPU SEC.  
 \* 6.644 ELAPSED SEC.

\*READY:  
 -\* TO ACTUALLY PRODUCE A CALCOMP PLOT \*CCQUEJE MUST BE \$RUN AFTER  
 +THE MICRO SESSION. THIS WILL BE DONE AFTER THIS DEMONSTRATION SESSION AND THE PLOT GENERATED  
 +WILL FOLLOW.

\* \$ .01, \$7.09T  
 \* 0.004 CPU SEC.  
 \* 1.171 ELAPSED SEC.

\*READY:  
 -\* END EXERCISE B-18.

\* \$ .00, \$7.09T  
 \* 0.003 CPU SEC.  
 \* 1.238 ELAPSED SEC.

\*READY:  
 -\*

## EXERCISE B-19 - THE DESTROY COMMAND.

\* \$ .00, \$7.10T  
 \* 0.003 CPU SEC.  
 \* 3.306 ELAPSED SEC.

\*READY:  
 -SET INPUT ECHO OFF.  
 \* \$ .01, \$7.10T  
 \* 0.003 CPU SEC.  
 \* 1.220 ELAPSED SEC.

\*READY:  
 -destroy recipients.  
 \*DATA SET "RECIPIENTS " IS TO BE DESTROYED. PLEASE CONFIRM.  
 \*OK

\* \$ .07, \$7.17T  
 \* 0.110 CPU SEC.  
 \* 15.139 ELAPSED SEC.

\*READY:  
 -destroy joined-set ok.

\* \$ .08, \$7.25T  
 \* 0.127 CPU SEC.  
 \* 2.588 ELAPSED SEC.

\*READY:  
 -\* end of exercise b-19.

\* \$ .02, \$7.27T  
 \* 0.004 CPU SEC.  
 \* 0.399 ELAPSED SEC.

\*READY:  
 -

```

#T=0.005 DR=0 $0.01, $7.34T
#r *ccqueue par=plotout
#EXECUTION BEGINS
1 PLOTS; PLOTTING REQUIRES 47 SEC. AND 13 IN.; $2.1
OK?
OK
#ILIR:PLOTOUT " HAS BEEN PERMITTED "R PRJNO=W".
PLOT ASSIGNED RECEIPT # 529987.
#EXECUTION TERMINATED
#T=0.176 DR=1 $2.28, $7.63T
#

```

\* EXERCISE B-20 - THE STOP COMMAND.

```

* $0.03, $7.30T
* 0.004 CPU SEC.
* 0.767 ELAPSED SEC.

```

```

*READY:
-stop.

```

```

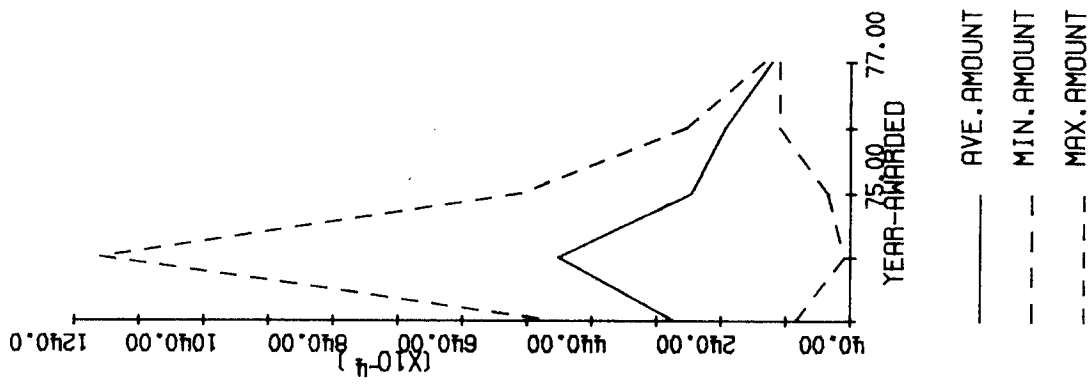
*23:04:00
*JUN 29, 1977
#EXECUTION TERMINATED
#T=14.157 DR=5202 $6.64, $7.33T

```

```

#signoff
#ILIR 22:10:31-23:07:07 WED JUN 29/77
#TERM,LOW,UNIV $1.12
#ELAPSED TIME 56.584 MIN. $4.24
#CPU TIME USED 16.247 SEC. $2.15
#CPU STOR VMI 27.417 PAGE-MIN.
#WAIT STOR VMI 95.553 PAGE-HP.
#DRUM READS 5383
#PLOT TIME .8 MIN. $0.11
#PLOT PAPER 1.13 FEET $0.11
#APPROX. COST OF THIS RUN IS $7.73
#DISK STORAGE 3959 PAGE-HR. $0.30
#APPROX. REMAINING BALANCE: $49.97
p

```





APPENDIX C  
Example Custom Assembly Language Program Listing

01-13-77 22:17:45 PAGE 0001

| LOCATION | INSTRUCTION | ADDR 1 | ADDR 2 | LINE # | SOURCE |
|----------|-------------|--------|--------|--------|--------|
|          |             |        |        | 1.000  | MACRO  |
|          |             |        |        | 2.000  | CVERT  |
|          |             |        |        | 3.000  | LCIA   |
|          |             |        |        | 4.000  | PACK   |
|          |             |        |        | 5.000  | BAL    |
|          |             |        |        | 6.000  | AIF    |
|          |             |        |        | 7.000  | AIF    |
|          |             |        |        | 8.000  | SETA   |
|          |             |        |        | 9.000  | HVC    |
|          |             |        |        | 10.000 | AGO    |
|          |             |        |        | 11.000 | HVC    |
|          |             |        |        | 12.000 | MEND   |

1.000  
 2.000  
 3.000  
 4.000  
 5.000  
 6.000  
 7.000  
 8.000  
 9.000  
 10.000  
 11.000  
 12.000

ENLEN, &VAR, &OUTLEN  
 &C  
 PAK(8), &VAR, (&INLEN)  
 14, CONVERT  
 (&OUTLEN EQ 3) .END  
 (&OUTLEN EQ 4) .JUMP  
 4-&OUTLEN  
 &VAR, #(&OUTLEN), AP+&C  
 .END  
 &VAR, #(&4), AP

01-13-77 PAGE 0002  
22:17:45

# ESTABLISH ADDRESSIBILITY

| LOCATION | INSTRUCTION    | ADDR 1   | ADDR 2   | LINE # | SOURCE | CSECT               | COMMENT                                   |
|----------|----------------|----------|----------|--------|--------|---------------------|-------------------------------------------|
| 01 00000 | 01 00000       |          |          | 2.000  | PROG   | BALR 12,0           |                                           |
| 01 00000 | 05C0           |          |          | 3.000  |        | USING *,12          |                                           |
| 01 00002 | 01 00002       |          |          | 4.000  |        | USING OUTPUT,9      |                                           |
| 02 00000 | 02 00000       |          |          | 5.000  |        |                     |                                           |
| 01 00002 | 989A C1D2      |          | 01 001D4 | 6.000  | *      | LM 9,10,OUTSTATS    |                                           |
| 01 00006 | 58P0 C30A      |          | 01 001D0 | 7.000  |        | L 15,=P'32767,      | DETERMINE BLOCK SIZE                      |
| 01 0000A | 1B2E           |          |          | 8.000  |        | SR 14,14            |                                           |
| 01 0000C | 1DEA C1D8      |          | 01 001DA | 9.000  |        | DR 14,10            |                                           |
| 01 0000F | 4CP0           |          |          | 10.000 |        | MH 15,ORECSIZ+2     |                                           |
| 01 00012 | 18BF           |          |          | 11.000 |        | LR 11,15            |                                           |
| 01 00014 | 06B0           |          |          | 12.000 |        | BCTR 11,0           |                                           |
| 01 00016 | 41B9 B000      |          |          | 13.000 |        | LA 11,0(9,11)       |                                           |
|          |                |          |          | 14.000 |        |                     |                                           |
|          |                |          |          | 15.000 | *      |                     |                                           |
|          |                |          |          | 16.000 | *      |                     | GR9 POINTS TO CURRENT OUTPUT RECORD       |
|          |                |          |          | 17.000 | *      |                     | GR10=OUTPUT RECORD LENGTH                 |
|          |                |          |          | 18.000 | *      |                     | GR11 POINTS TO END-1 OF OUTPUT BLOCK      |
|          |                |          |          | 19.000 | *      |                     |                                           |
|          |                |          |          | 20.000 |        |                     | COUNTERS(12), COUNTERS ZERO PROCESS STATS |
| 01 0001A | D70B C1P2      | 01 001P4 | 01 001P4 | 21.000 | LOOP0  | XC 1,INPARH         |                                           |
| 01 00020 | 4110 C172      |          | 01 00174 | 22.000 |        | L 15,=V(READ)       |                                           |
| 01 00024 | 5AP0 C306      |          | 01 00308 | 23.000 |        | BALR 14,15          |                                           |
| 01 00028 | 05BF           |          |          | 24.000 |        | LTR 15,15           |                                           |
| 01 0002A | 12FF           |          |          | 25.000 |        | BNZ OUT             |                                           |
| 01 0002C | 4770 C100      |          | 01 00102 | 26.000 |        | L 15,INCOUNT        |                                           |
| 01 00030 | 58P0 C1P2      |          | 01 001P4 | 27.000 |        | A 15,=P'1,          |                                           |
| 01 00034 | 5AP0 C302      |          | 01 00304 | 28.000 |        | ST 15,INCOUNT       |                                           |
| 01 00038 | 50P0 C1P2      |          | 01 001P4 | 29.000 | *      |                     |                                           |
|          |                |          |          | 30.000 | *      |                     |                                           |
|          |                |          |          | 31.000 | *      |                     |                                           |
|          |                |          |          | 32.000 |        |                     |                                           |
| 01 0003C | P278 C1P6      | 01 001P8 | 01 00266 | 33.000 | *      | CVERT 9,SIN,4       |                                           |
| 01 0003C | P278 C1P6      | 01 001P8 | 01 00266 | 34.000 | *      | PACK PAK(8),SIN(9)  |                                           |
| 01 00042 | 45P0 C160      | 01 00162 | 01 00162 | 35.000 | *      | BAL 14,CONVERT      |                                           |
| 01 00046 | D203 9000 C1P2 | 02 00000 | 01 001P0 | 36.000 | *      | MVC SIN(4),AP       |                                           |
| 01 0004C | D217 9004 C26D | 02 00004 | 01 0026F | 37.000 | *      | MVC NAME(24),NAME   |                                           |
| 01 00052 | 05P0 C28F      | 01 00201 | 01 000E8 | 38.000 | *      | CLI RACE,X'P0'      |                                           |
| 01 00056 | 4740 C0E6      |          |          | 39.000 | *      | BL ERROR            |                                           |
| 01 0005A | P270 C1P6      | 01 001P8 | 01 00291 | 40.000 | *      | CVERT 1,RACE,1      |                                           |
| 01 0005A | P270 C1P6      | 01 001P8 | 01 00291 | 41.000 | *      | PACK PAK(8),RACE(1) |                                           |
| 01 00060 | 45E0 C160      | 01 00162 | 01 00162 | 42.000 | *      | BAL 14,CONVERT      |                                           |
| 01 00064 | D200 901C C1P1 | 02 0001C | 01 001P3 | 43.000 | *      | MVC RACE(1),AP+3    |                                           |
| 01 0006A | D200 901D C28F | 02 0001D | 01 00290 | 44.000 | *      | MVC SEX(1),SEFX     |                                           |
| 01 00070 | 95D4 C28F      | 01 00290 |          | 45.000 | *      | CLI SEX,C'N'        |                                           |

INCREMENT RECORDS READ

01-13-77 22:17:45  
PAGE 0003

| LOCATION | INSTRUCTION    | ADDR 1   | ADDR 2   | LINE #  | SOURCE | PUT BDAY IN YIMHDD FORMAT |
|----------|----------------|----------|----------|---------|--------|---------------------------|
| 01 00074 | 4780 C086      | 01 00290 | 01 00088 | 42.000  | B2     | SKIP1                     |
| 01 00078 | 95C6 C28E      | 01 00290 | 01 00088 | 43.000  | CLI    | SEX,C'P'                  |
| 01 0007C | 4780 C086      | 01 00290 | 01 00088 | 44.000  | BE     | SKIP1                     |
| 01 00080 | 95B4 C28E      | 01 00290 | 01 00088 | 45.000  | CLI    | SEX,C'P'                  |
| 01 00084 | 4770 C086      | 01 00038 | 01 00038 | 46.000  | BNE    | ERROR                     |
| 01 00088 | D201 C1FE C28B | 01 00200 | 01 0028D | 47.000  | MVC    | DATE(2),BDAY+4            |
| 01 0008E | D203 C200 C287 | 01 00202 | 01 00289 | 48.000  | MVC    | DATE+2(4),BDAY            |
| 01 00094 | 8275 C1FE C1FE | 01 001P8 | 01 00200 | 49.000  | CVERT  | 6,DATE,0                  |
| 01 00094 | 8275 C1FE C1FE | 01 001P8 | 01 00200 | 50.000  | PACK   | PAK(8),DATE(6)            |
| 01 0009A | 45E0 C160      | 01 00162 | 01 00162 | 51.000  | BAL    | 14,CONVERT                |
| 01 0009E | D202 901E C1EF | 02 0001E | 01 001P1 | 52.000  | MVC    | BDAY*(3),AP+1             |
| 01 000A4 | D501 C290 C10E | 01 00292 | 01 00310 | 53.000  | CLC    | CLASSYR(2),=C'-1'         |
| 01 000A8 | 4770 C0B4      | 01 00036 | 01 00036 | 54.000  | BNE    | SKIP2                     |
| 01 000AF | 92FF 9021      | 02 00021 | 01 00036 | 55.000  | MVI    | CLASSYR*,X'FF'            |
| 01 000B2 | 47E0 C0C4      | 01 000C6 | 01 000C6 | 56.000  | B      | ENDREC                    |
| 01 000B6 | P271 C1E6 C290 | 01 001E8 | 01 00292 | 57.000  | CVERT  | 2,CLASSYR,1               |
| 01 000B6 | P271 C1E6 C290 | 01 001E8 | 01 00292 | 58.000  | PACK   | PAK(8),CLASSYR(2)         |
| 01 000BC | 45E0 C160      | 01 00162 | 01 00162 | 59.000  | BAL    | 14,CONVERT                |
| 01 000C0 | D200 9021 C1F1 | 02 00021 | 01 001P3 | 60.000  | MVC    | CLASSYR*(1),AP+3          |
| 01 000C6 | 58P0 C1FA      | 01 001FC | 01 001FC | 61.000  | L      | 15,OUTCOUNT               |
| 01 000CA | 5AP0 C302      | 01 00304 | 01 00304 | 62.000  | A      | 15,=P'1'                  |
| 01 000CE | 50P0 C1FA      | 01 001FC | 01 001FC | 63.000  | ST     | 15,OUTCOUNT               |
| 01 000D2 | 979A C01E      | 01 00020 | 01 00020 | 64.000  | BKLE   | 9,10,LOOP0                |
| 01 000D6 | 4110 C186      | 01 00188 | 01 00188 | 65.000  | LA     | 1,OUTPARM                 |
| 01 000DA | 5AP0 C2FF      | 01 00300 | 01 00300 | 66.000  | L      | 15,=V(WRITE)              |
| 01 000DE | 05EP           | 01 00188 | 01 00188 | 67.000  | BALR   | 14,15                     |
| 01 000E0 | 58P0 C186      | 01 00020 | 01 00020 | 68.000  | L      | 9,OUTPARM                 |
| 01 000E4 | 47P0 C01E      | 01 00020 | 01 00020 | 69.000  | B      | LOOP0                     |
| 01 000E8 | 5AP0 C1FE      | 01 001P8 | 01 001P8 | 70.000  | L      | 15,ERRCOUNT               |
| 01 000EC | 5AP0 C302      | 01 00304 | 01 00304 | 71.000  | A      | 15,=P'1'                  |
| 01 000F0 | 50P0 C1P6      | 01 001P8 | 01 001P8 | 72.000  | ST     | 15,ERRCOUNT               |
| 01 000F4 | 4110 C19A      | 01 0019C | 01 0019C | 73.000  | LA     | 1,ERRPARM                 |
| 01 000F8 | 58P0 C2FE      | 01 00300 | 01 00300 | 74.000  | L      | 15,=V(WRITE)              |
| 01 000FC | 05EP           | 01 00020 | 01 00020 | 75.000  | BALR   | 14,15                     |
| 01 000FE | 47P0 C01P      | 01 00020 | 01 00020 | 76.000  | B      | LOOP0                     |
| 01 00102 | 58P0 C186      | 01 00188 | 01 00188 | 77.000  | L      | 15,ERRCOUNT               |
| 01 00106 | 5AP0 C302      | 01 00304 | 01 00304 | 78.000  | A      | 15,=P'1'                  |
| 01 0010A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 79.000  | ST     | 15,ERRCOUNT               |
| 01 0010E | 4110 C19A      | 01 0019C | 01 0019C | 80.000  | LA     | 1,ERRPARM                 |
| 01 00112 | 58P0 C186      | 01 00188 | 01 00188 | 81.000  | L      | 15,=V(WRITE)              |
| 01 00116 | 5AP0 C302      | 01 00304 | 01 00304 | 82.000  | BALR   | 14,15                     |
| 01 0011A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 83.000  | B      | LOOP0                     |
| 01 0011E | 47P0 C01P      | 01 00020 | 01 00020 | 84.000  | L      | 15,ERRCOUNT               |
| 01 00122 | 58P0 C186      | 01 00188 | 01 00188 | 85.000  | A      | 15,=P'1'                  |
| 01 00126 | 5AP0 C302      | 01 00304 | 01 00304 | 86.000  | ST     | 15,ERRCOUNT               |
| 01 0012A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 87.000  | LA     | 1,ERRPARM                 |
| 01 0012E | 4110 C19A      | 01 0019C | 01 0019C | 88.000  | L      | 15,=V(WRITE)              |
| 01 00132 | 58P0 C186      | 01 00188 | 01 00188 | 89.000  | BALR   | 14,15                     |
| 01 00136 | 5AP0 C302      | 01 00304 | 01 00304 | 90.000  | B      | LOOP0                     |
| 01 0013A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 91.000  | L      | 15,ERRCOUNT               |
| 01 0013E | 47P0 C01P      | 01 00020 | 01 00020 | 92.000  | A      | 15,=P'1'                  |
| 01 00142 | 58P0 C186      | 01 00188 | 01 00188 | 93.000  | ST     | 15,ERRCOUNT               |
| 01 00146 | 5AP0 C302      | 01 00304 | 01 00304 | 94.000  | LA     | 1,ERRPARM                 |
| 01 0014A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 95.000  | L      | 15,=V(WRITE)              |
| 01 0014E | 4110 C19A      | 01 0019C | 01 0019C | 96.000  | BALR   | 14,15                     |
| 01 00152 | 58P0 C186      | 01 00188 | 01 00188 | 97.000  | L      | 9,OUTPARM                 |
| 01 00156 | 5AP0 C302      | 01 00304 | 01 00304 | 98.000  | B      | LOOP0                     |
| 01 0015A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 99.000  | L      | 15,ERRCOUNT               |
| 01 0015E | 47P0 C01P      | 01 00020 | 01 00020 | 100.000 | A      | 15,=P'1'                  |
| 01 00162 | 58P0 C186      | 01 00188 | 01 00188 | 101.000 | ST     | 15,ERRCOUNT               |
| 01 00166 | 5AP0 C302      | 01 00304 | 01 00304 | 102.000 | LA     | 1,ERRPARM                 |
| 01 0016A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 103.000 | L      | 15,=V(WRITE)              |
| 01 0016E | 4110 C19A      | 01 0019C | 01 0019C | 104.000 | BALR   | 14,15                     |
| 01 00172 | 58P0 C186      | 01 00188 | 01 00188 | 105.000 | L      | 9,OUTPARM                 |
| 01 00176 | 5AP0 C302      | 01 00304 | 01 00304 | 106.000 | B      | LOOP0                     |
| 01 0017A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 107.000 | L      | 15,ERRCOUNT               |
| 01 0017E | 47P0 C01P      | 01 00020 | 01 00020 | 108.000 | A      | 15,=P'1'                  |
| 01 00182 | 58P0 C186      | 01 00188 | 01 00188 | 109.000 | ST     | 15,ERRCOUNT               |
| 01 00186 | 5AP0 C302      | 01 00304 | 01 00304 | 110.000 | LA     | 1,ERRPARM                 |
| 01 0018A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 111.000 | L      | 15,=V(WRITE)              |
| 01 0018E | 4110 C19A      | 01 0019C | 01 0019C | 112.000 | BALR   | 14,15                     |
| 01 00192 | 58P0 C186      | 01 00188 | 01 00188 | 113.000 | L      | 9,OUTPARM                 |
| 01 00196 | 5AP0 C302      | 01 00304 | 01 00304 | 114.000 | B      | LOOP0                     |
| 01 0019A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 115.000 | L      | 15,ERRCOUNT               |
| 01 0019E | 47P0 C01P      | 01 00020 | 01 00020 | 116.000 | A      | 15,=P'1'                  |
| 01 001A2 | 58P0 C186      | 01 00188 | 01 00188 | 117.000 | ST     | 15,ERRCOUNT               |
| 01 001A6 | 5AP0 C302      | 01 00304 | 01 00304 | 118.000 | LA     | 1,ERRPARM                 |
| 01 001AA | 50P0 C1P6      | 01 001P8 | 01 001P8 | 119.000 | L      | 15,=V(WRITE)              |
| 01 001AE | 4110 C19A      | 01 0019C | 01 0019C | 120.000 | BALR   | 14,15                     |
| 01 001B2 | 58P0 C186      | 01 00188 | 01 00188 | 121.000 | L      | 9,OUTPARM                 |
| 01 001B6 | 5AP0 C302      | 01 00304 | 01 00304 | 122.000 | B      | LOOP0                     |
| 01 001BA | 50P0 C1P6      | 01 001P8 | 01 001P8 | 123.000 | L      | 15,ERRCOUNT               |
| 01 001BE | 47P0 C01P      | 01 00020 | 01 00020 | 124.000 | A      | 15,=P'1'                  |
| 01 001C2 | 58P0 C186      | 01 00188 | 01 00188 | 125.000 | ST     | 15,ERRCOUNT               |
| 01 001C6 | 5AP0 C302      | 01 00304 | 01 00304 | 126.000 | LA     | 1,ERRPARM                 |
| 01 001CA | 50P0 C1P6      | 01 001P8 | 01 001P8 | 127.000 | L      | 15,=V(WRITE)              |
| 01 001CE | 4110 C19A      | 01 0019C | 01 0019C | 128.000 | BALR   | 14,15                     |
| 01 001D2 | 58P0 C186      | 01 00188 | 01 00188 | 129.000 | L      | 9,OUTPARM                 |
| 01 001D6 | 5AP0 C302      | 01 00304 | 01 00304 | 130.000 | B      | LOOP0                     |
| 01 001DA | 50P0 C1P6      | 01 001P8 | 01 001P8 | 131.000 | L      | 15,ERRCOUNT               |
| 01 001DE | 47P0 C01P      | 01 00020 | 01 00020 | 132.000 | A      | 15,=P'1'                  |
| 01 001E2 | 58P0 C186      | 01 00188 | 01 00188 | 133.000 | ST     | 15,ERRCOUNT               |
| 01 001E6 | 5AP0 C302      | 01 00304 | 01 00304 | 134.000 | LA     | 1,ERRPARM                 |
| 01 001EA | 50P0 C1P6      | 01 001P8 | 01 001P8 | 135.000 | L      | 15,=V(WRITE)              |
| 01 001EE | 4110 C19A      | 01 0019C | 01 0019C | 136.000 | BALR   | 14,15                     |
| 01 001F2 | 58P0 C186      | 01 00188 | 01 00188 | 137.000 | L      | 9,OUTPARM                 |
| 01 001F6 | 5AP0 C302      | 01 00304 | 01 00304 | 138.000 | B      | LOOP0                     |
| 01 001FA | 50P0 C1P6      | 01 001P8 | 01 001P8 | 139.000 | L      | 15,ERRCOUNT               |
| 01 001FE | 47P0 C01P      | 01 00020 | 01 00020 | 140.000 | A      | 15,=P'1'                  |
| 01 00102 | 58P0 C186      | 01 00188 | 01 00188 | 141.000 | ST     | 15,ERRCOUNT               |
| 01 00106 | 5AP0 C302      | 01 00304 | 01 00304 | 142.000 | LA     | 1,ERRPARM                 |
| 01 0010A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 143.000 | L      | 15,=V(WRITE)              |
| 01 0010E | 4110 C19A      | 01 0019C | 01 0019C | 144.000 | BALR   | 14,15                     |
| 01 00112 | 58P0 C186      | 01 00188 | 01 00188 | 145.000 | L      | 9,OUTPARM                 |
| 01 00116 | 5AP0 C302      | 01 00304 | 01 00304 | 146.000 | B      | LOOP0                     |
| 01 0011A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 147.000 | L      | 15,ERRCOUNT               |
| 01 0011E | 47P0 C01P      | 01 00020 | 01 00020 | 148.000 | A      | 15,=P'1'                  |
| 01 00122 | 58P0 C186      | 01 00188 | 01 00188 | 149.000 | ST     | 15,ERRCOUNT               |
| 01 00126 | 5AP0 C302      | 01 00304 | 01 00304 | 150.000 | LA     | 1,ERRPARM                 |
| 01 0012A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 151.000 | L      | 15,=V(WRITE)              |
| 01 0012E | 4110 C19A      | 01 0019C | 01 0019C | 152.000 | BALR   | 14,15                     |
| 01 00132 | 58P0 C186      | 01 00188 | 01 00188 | 153.000 | L      | 9,OUTPARM                 |
| 01 00136 | 5AP0 C302      | 01 00304 | 01 00304 | 154.000 | B      | LOOP0                     |
| 01 0013A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 155.000 | L      | 15,ERRCOUNT               |
| 01 0013E | 47P0 C01P      | 01 00020 | 01 00020 | 156.000 | A      | 15,=P'1'                  |
| 01 00142 | 58P0 C186      | 01 00188 | 01 00188 | 157.000 | ST     | 15,ERRCOUNT               |
| 01 00146 | 5AP0 C302      | 01 00304 | 01 00304 | 158.000 | LA     | 1,ERRPARM                 |
| 01 0014A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 159.000 | L      | 15,=V(WRITE)              |
| 01 0014E | 4110 C19A      | 01 0019C | 01 0019C | 160.000 | BALR   | 14,15                     |
| 01 00152 | 58P0 C186      | 01 00188 | 01 00188 | 161.000 | L      | 9,OUTPARM                 |
| 01 00156 | 5AP0 C302      | 01 00304 | 01 00304 | 162.000 | B      | LOOP0                     |
| 01 0015A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 163.000 | L      | 15,ERRCOUNT               |
| 01 0015E | 47P0 C01P      | 01 00020 | 01 00020 | 164.000 | A      | 15,=P'1'                  |
| 01 00162 | 58P0 C186      | 01 00188 | 01 00188 | 165.000 | ST     | 15,ERRCOUNT               |
| 01 00166 | 5AP0 C302      | 01 00304 | 01 00304 | 166.000 | LA     | 1,ERRPARM                 |
| 01 0016A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 167.000 | L      | 15,=V(WRITE)              |
| 01 0016E | 4110 C19A      | 01 0019C | 01 0019C | 168.000 | BALR   | 14,15                     |
| 01 00172 | 58P0 C186      | 01 00188 | 01 00188 | 169.000 | L      | 9,OUTPARM                 |
| 01 00176 | 5AP0 C302      | 01 00304 | 01 00304 | 170.000 | B      | LOOP0                     |
| 01 0017A | 50P0 C1P6      | 01 001P8 | 01 001P8 | 171.000 | L      | 15,ERRCOUNT               |
| 01 0017E | 47P0 C01P      | 01 00020 | 01 00020 | 172.000 | A      | 15,                       |

NOPE, SKIP AHEAD  
YUP, OUTPUT IT

OUTPUT RUN STATISTICS

| LOCATION | INSTRUCTION      | ADDR 1   | ADDR 2   | LINE #    | SOURCE                    | BZ          | STOP                               |
|----------|------------------|----------|----------|-----------|---------------------------|-------------|------------------------------------|
| 01 00105 | 4780 C116        | 01 00118 | 01 00118 | 83.000    |                           | STR         | 9,OUTLEN                           |
| 01 0010A | 4090 C1DC        | 01 001DE | 01 001DE | 84.000    |                           | LA          | 1,OUTPARM                          |
| 01 0010P | 4110 C186        | 01 00188 | 01 00188 | 85.000    |                           | L           | 15,=V(WRITE)                       |
| 01 00112 | 58P0 C2PE        | 01 00300 | 01 00300 | 86.000    |                           | BALR        | 14,15                              |
| 01 0011K | 05EP             |          |          | 87.000    |                           |             |                                    |
|          |                  |          |          | 88.000 *  |                           |             |                                    |
|          |                  |          |          | 89.000 *  |                           |             |                                    |
| 01 00118 | 58P0 C1P2        | 01 001P4 | 01 001P4 | 90.000    | STOP                      | L           | 15,INCOUNT                         |
| 01 0011C | 4110 C254        | 01 00256 | 01 00256 | 91.000    |                           | LA          | 1,SOURCE                           |
| 01 00120 | 4580 C148        | 01 0014A | 01 0014A | 92.000    |                           | BAL         | 14,PACKIT                          |
| 01 00124 | 58P0 C1PA        | 01 001PC | 01 001PC | 93.000    |                           | L           | 15,OUTCOUNT                        |
| 01 00128 | 4580 C148        | 01 0014A | 01 0014A | 94.000    |                           | BAL         | 14,PACKIT                          |
| 01 0012C | 58P0 C1P6        | 01 001P8 | 01 001P8 | 95.000    |                           | L           | 15,ERRCOUNT                        |
| 01 00130 | 45P0 C148        | 01 0014A | 01 0014A | 96.000    |                           | BAL         | 14,PACKIT                          |
|          |                  |          |          | 97.000 *  |                           |             |                                    |
| 01 00134 | 0P3E C204        | 01 00206 | 01 00256 | 98.000    |                           | ED          | PATTERN(63),SOURCE                 |
| 01 0013A | 4110 C1AE        | 01 001B0 | 01 001B0 | 99.000    |                           | LA          | 1,RETPARM                          |
| 01 0013P | 58P0 C2PA        | 01 002PC | 01 002PC | 100.000   |                           | L           | 15,=V(SPRINT)                      |
| 01 00142 | 05EP             |          |          | 101.000   |                           | BALR        | 14,15                              |
| 01 00144 | 58P0 C2P6        | 01 002P8 | 01 002P8 | 102.000   |                           | L           | 15,=V(SYSTEM)                      |
| 01 00148 | 07EP             |          |          | 103.000   |                           | BR          | 15                                 |
|          |                  |          |          | 104.000 * |                           |             |                                    |
| 01 0014A | 4EP0 C1E6        | 01 001E8 | 01 001E8 | 105.000   | PACKIT                    | CVD         | 15,PAK                             |
| 01 0014E | 960P C1E0        | 01 001EP | 01 001EP | 106.000   |                           | CI          | PAK+7,K'OP'                        |
| 01 00152 | 94PD C1ED        | 01 001EP | 01 001EP | 107.000   |                           | HI          | PAK+7,K'PD'                        |
| 01 00156 | D204 1000 C1E9   | 01 001EB | 01 001EB | 108.000   |                           | MVC         | 0(5,1),PAK+3                       |
| 01 0015C | 4110 1005        |          |          | 109.000   |                           | LA          | 1,5(,1)                            |
| 01 00160 | 07EP             |          |          | 110.000   |                           | BR          | 14                                 |
|          |                  |          |          | 111.000 * |                           |             |                                    |
|          |                  |          |          | 112.000 * |                           |             |                                    |
|          |                  |          |          | 113.000 * | CONTENTS GR15 ARE ALTERED |             |                                    |
| 01 00162 | 960P C1ED        | 01 001EP | 01 001EP | 114.000   | CONVERT CI                | PAK+7,K'OP' | MAKE SIGN LEGAL                    |
| 01 00166 | 4EP0 C1E6        | 01 001E8 | 01 001E8 | 115.000   | CVB                       | 15,PAK      | IN CASE CVB BLOWS UP               |
| 01 0016A | 4704 C0E6        | 01 000E8 | 01 000E8 | 116.000   | NOP                       | ERROR(4)    |                                    |
| 01 0016P | 50P0 C1EP        | 01 001P0 | 01 001P0 | 117.000   | ST                        | 15,AP       |                                    |
| 01 00172 | 07EP             |          |          | 118.000   | BR                        | 14          |                                    |
|          |                  |          |          | 119.000 * |                           |             |                                    |
|          |                  |          |          | 120.000 * |                           |             |                                    |
| 01 00174 | 00000265000001DC |          |          | 121.000   | INPARM                    | DC          | A (INPUT,INLEN,MOD,LNUM,INUNIT)    |
| 01 00198 | 00000312000001DE |          |          | 122.000   | OUTPARM                   | DC          | A (OUTREG,OUTLEN,MOD,LNUM,OUTUNIT) |
| 01 0019C | 00000265000001DC |          |          | 123.000   | ERRPARM                   | DC          | A (INPUT,INLEN,MOD,LNUM,ERRUNIT)   |
| 01 00180 | 0000020600000130 |          |          | 124.000   | RETPARM                   | DC          | A (PATTERN,REPLEN,MOD,LNUM)        |
|          |                  |          |          | 125.000 * |                           |             |                                    |
| 01 001C0 | 00000000         |          |          | 126.000   | MOD                       | DC          | P'0'                               |
| 01 001C4 | 00000000         |          |          | 127.000   | LNUM                      | DC          | P'0'                               |
|          |                  |          |          | 128.000 * |                           |             |                                    |

| LOCATION | INSTRUCTION | ADDR 1 | ADDR 2 | LINE #  | SOURCE       | P.5' | P.6' | P.7' |
|----------|-------------|--------|--------|---------|--------------|------|------|------|
| 01 00108 | 00000005    |        |        | 129.000 | INUNIT DC    |      |      |      |
| 01 00109 | 00000006    |        |        | 130.000 | OUTUNIT DC   |      |      |      |
| 01 00110 | 00000007    |        |        | 131.000 | ERRUNIT DC   |      |      |      |
| 01 00111 | 00000008    |        |        | 132.000 | *            |      |      |      |
| 01 00112 | 00000009    |        |        | 133.000 | OUTSTATS DC  |      |      |      |
| 01 00113 | 00000010    |        |        | 134.000 | ORECSIZ DC   |      |      |      |
| 01 00114 | 00000011    |        |        | 135.000 | *            |      |      |      |
| 01 00115 | 00000012    |        |        | 136.000 | *            |      |      |      |
| 01 00116 | 00000013    |        |        | 137.000 | *            |      |      |      |
| 01 00117 | 00000014    |        |        | 138.000 | *            |      |      |      |
| 01 00118 | 00000015    |        |        | 139.000 | INLEN DC     |      |      |      |
| 01 00119 | 00000016    |        |        | 140.000 | OUTLEN DC    |      |      |      |
| 01 00120 | 00000017    |        |        | 141.000 | REPTLEN DC   |      |      |      |
| 01 00121 | 00000018    |        |        | 142.000 | *            |      |      |      |
| 01 00122 | 00000019    |        |        | 143.000 | DS           |      |      |      |
| 01 00123 | 00000020    |        |        | 144.000 | PAK DC       |      |      |      |
| 01 00124 | 00000021    |        |        | 145.000 | AP DS        |      |      |      |
| 01 00125 | 00000022    |        |        | 146.000 | *            |      |      |      |
| 01 00126 | 00000023    |        |        | 147.000 | COUNTERS EQU |      |      |      |
| 01 00127 | 00000024    |        |        | 148.000 | INCOUNT DS   |      |      |      |
| 01 00128 | 00000025    |        |        | 149.000 | ERRCOUNT DS  |      |      |      |
| 01 00129 | 00000026    |        |        | 150.000 | OUTCOUNT DS  |      |      |      |
| 01 00130 | 00000027    |        |        | 151.000 | *            |      |      |      |
| 01 00131 | 00000028    |        |        | 152.000 | DATE DS      |      |      |      |
| 01 00132 | 00000029    |        |        | 153.000 | *            |      |      |      |
| 01 00133 | 00000030    |        |        | 154.000 | PATTERN DC   |      |      |      |
| 01 00134 | 00000031    |        |        | 155.000 | DC           |      |      |      |
| 01 00135 | 00000032    |        |        | 156.000 | DC           |      |      |      |
| 01 00136 | 00000033    |        |        | 157.000 | DC           |      |      |      |
| 01 00137 | 00000034    |        |        | 158.000 | SOURCE DS    |      |      |      |
| 01 00138 | 00000035    |        |        | 159.000 | *            |      |      |      |
| 01 00139 | 00000036    |        |        | 160.000 | *            |      |      |      |
| 01 00140 | 00000037    |        |        | 161.000 | *            |      |      |      |
| 01 00141 | 00000038    |        |        | 162.000 | INPUT EQU    |      |      |      |
| 01 00142 | 00000039    |        |        | 163.000 | STATUS DS    |      |      |      |
| 01 00143 | 00000040    |        |        | 164.000 | SIN DS       |      |      |      |
| 01 00144 | 00000041    |        |        | 165.000 | NAME DS      |      |      |      |
| 01 00145 | 00000042    |        |        | 166.000 | STATE DS     |      |      |      |
| 01 00146 | 00000043    |        |        | 167.000 | BDAY DS      |      |      |      |
| 01 00147 | 00000044    |        |        | 168.000 | CITIZEN DS   |      |      |      |
| 01 00148 | 00000045    |        |        | 169.000 | SEX DS       |      |      |      |
| 01 00149 | 00000046    |        |        | 170.000 | RACE DS      |      |      |      |
| 01 00150 | 00000047    |        |        | 171.000 | CLASSYR DS   |      |      |      |
| 01 00151 | 00000048    |        |        | 172.000 | *            |      |      |      |

\*\*\*MICRO RECORD SIZE\*\*\*

ORECSIZ MUST BE SET TO THE MICRO DATASET WIDTH AS DEFINED IN THE DICTIONARY (TOTAL NUMBER OF BYTES PER RECORD)

NUMBER OF RECORDS READ  
NUMBER OF RECORDS REJECTED  
NUMBER OF RECORDS PROCESSED

X'402020202020202121',C' RECORDS READ/'

X'222020202020202121'

C' RECORDS PROCESSED/,X'222020202020202121'

C' RECORDS REJECTED'

KL15

KL1

KL9

KL24

KL2

KL6

KL1

KL1

KL1

KL1

KL2

PAGE 0006  
01-13-77 22:17:45  
IN CASE INPUT IS LONG

| LOCATION | INSTRUCTION | ADDR 1 | ADDR 2 | LINE #    | SOURCE   | DS    | XL100      |
|----------|-------------|--------|--------|-----------|----------|-------|------------|
| 01 00294 |             |        |        | 173.000   | SAFETY   | DS    |            |
|          |             |        |        | 174.000 * |          |       |            |
|          |             |        |        | 175.000 * |          |       |            |
|          |             |        |        | 176.000   |          |       |            |
|          | 01 002F8    |        |        |           |          | LTOR3 |            |
| 01 002P8 | 00000000    |        |        |           |          |       | =V(SYSTEM) |
| 01 002PC | 00000000    |        |        |           |          |       | =V(SCRIPT) |
| 01 00300 | 00000000    |        |        |           |          |       | =V(WRITE)  |
| 01 00304 | 00000001    |        |        |           |          |       | =P.1.      |
| 01 00308 | 00000000    |        |        |           |          |       | =V(READ)   |
| 01 0030C | 00007FFF    |        |        |           |          |       | =P.32767.  |
| 01 00310 | 60P1        |        |        |           |          |       | =C'-1.     |
|          |             |        |        | 177.000 * |          |       |            |
|          |             |        |        | 178.000 * |          |       |            |
| 01 00312 |             |        |        | 179.000   | OUTREG   | DS    | XL32767    |
|          |             |        |        | 180.000 * |          |       |            |
|          |             |        |        | 181.000 * |          |       |            |
|          | 02 00000    |        |        | 182.000   | OUTPUT   | DSECT |            |
| 02 00000 |             |        |        | 183.000   | SIN#     | DS    | XL4        |
| 02 00004 |             |        |        | 184.000   | NAME#    | DS    | XL24       |
| 02 0001C |             |        |        | 185.000   | RACE#    | DS    | XL1        |
| 02 0001D |             |        |        | 186.000   | SEX#     | DS    | XL1        |
| 02 0001P |             |        |        | 187.000   | BDAY#    | DS    | XL3        |
| 02 00021 |             |        |        | 188.000   | CLASSYR# | DS    | XL1        |
|          |             |        |        | 189.000 * |          |       |            |
|          |             |        |        | 190.000 * |          |       |            |
|          |             |        |        | 191.000   |          |       |            |
|          |             |        |        |           |          |       | END        |

NO ERRORS IN THIS ASSEMBLY

## CROSS REFERENCE LISTING

| SYMBOL   | TYPE | LNK   | LOCATION | REFERENCES                             | REFERENCES                   |
|----------|------|-------|----------|----------------------------------------|------------------------------|
| AP       | A    | 00004 | 01 0010  | 01 00046, 01 00093, 01 00000, 01 00162 | 01 00064, 01 00082, 01 00082 |
| RDAY     | X    | 00006 | 01 00289 | 01 00088, 01 00095                     |                              |
| BDAY     | X    | 00003 | 02 00012 |                                        |                              |
| CITIZEN  | X    | 00001 | 01 00287 |                                        |                              |
| CLASSYP  | Y    | 00002 | 01 00292 | 01 00044, 01 00066                     |                              |
| CLASSYP  | X    | 00001 | 02 00021 | 01 00046, 01 00060                     | 01 0008C                     |
| CONVERT  | I    | 00004 | 01 00162 | 01 00042, 01 0001A, 01 00083, 01 00094 |                              |
| COUNTERS | U    | 00001 | 01 00174 | 01 00088, 01 00082, 01 00084, 01 0012C |                              |
| DATE     | C    | 00006 | 01 00200 | 01 00088, 01 00082, 01 00084, 01 0016A |                              |
| ENDREC   | I    | 00004 | 01 00066 | 01 00082, 01 00084, 01 0012C           |                              |
| ERRCOUNT | A    | 00004 | 01 00178 | 01 00088, 01 00084, 01 0016A           |                              |
| ERRPROP  | I    | 00004 | 01 00088 | 01 00088, 01 00084, 01 0016A           |                              |
| ERRPARM  | A    | 00004 | 01 0019C | 01 00088, 01 00084, 01 0016A           |                              |
| ERRUNIT  | P    | 00004 | 01 00100 | 01 00088, 01 00084, 01 0016A           |                              |
| INCOUNT  | A    | 00004 | 01 00174 | 01 00088, 01 00084, 01 0016A           |                              |
| INLEN    | H    | 00002 | 01 001DC | 01 00088, 01 00084, 01 0016A           |                              |
| INPARM   | A    | 00004 | 01 00174 | 01 00088, 01 00084, 01 0016A           |                              |
| INPUT    | U    | 00001 | 01 00255 | 01 00088, 01 00084, 01 0016A           |                              |
| INPUT    | P    | 00004 | 01 001C8 | 01 00088, 01 00084, 01 0016A           |                              |
| INUM     | P    | 00004 | 01 001C4 | 01 00088, 01 00084, 01 0016A           |                              |
| LOOP     | I    | 00004 | 01 00020 | 01 00088, 01 00084, 01 0016A           |                              |
| LOOP     | P    | 00004 | 01 001C0 | 01 00088, 01 00084, 01 0016A           |                              |
| MOD      | X    | 00004 | 01 0026P | 01 00088, 01 00084, 01 0016A           |                              |
| NAME     | X    | 00018 | 02 00004 | 01 00088, 01 00084, 01 0016A           |                              |
| NAME     | X    | 00018 | 02 00004 | 01 00088, 01 00084, 01 0016A           |                              |
| OPCSIZ   | A    | 00004 | 01 001D8 | 01 00088, 01 00084, 01 0016A           |                              |
| OUT      | I    | 00004 | 01 00102 | 01 00088, 01 00084, 01 0016A           |                              |
| OUTCOUNT | A    | 00004 | 01 001PC | 01 00088, 01 00084, 01 0016A           |                              |
| OUTLEN   | H    | 00002 | 01 001DE | 01 00088, 01 00084, 01 0016A           |                              |
| OUTPARM  | A    | 00004 | 01 001B8 | 01 00088, 01 00084, 01 0016A           |                              |
| OUTPUT   | I    | 00001 | 02 00000 | 01 00088, 01 00084, 01 0016A           |                              |
| OUTPUT   | X    | 07PPP | 01 00312 | 01 00088, 01 00084, 01 0016A           |                              |
| OUTREG   | A    | 00004 | 01 001D4 | 01 00088, 01 00084, 01 0016A           |                              |
| OUTSTATS | A    | 00004 | 01 001CC | 01 00088, 01 00084, 01 0016A           |                              |
| OUTUNIT  | P    | 00004 | 01 0014A | 01 00088, 01 00084, 01 0016A           |                              |
| PACKET   | I    | 00004 | 01 0014A | 01 00088, 01 00084, 01 0016A           |                              |
| PAK      | X    | 00008 | 01 001ER | 01 00088, 01 00084, 01 0016A           |                              |
| PATTERN  | X    | 0000A | 01 00206 | 01 00088, 01 00084, 01 0016A           |                              |
| PROG     | I    | 00001 | 01 00000 | 01 00088, 01 00084, 01 0016A           |                              |
| RACE     | X    | 00001 | 01 00291 | 01 00088, 01 00084, 01 0016A           |                              |
| PACR#    | X    | 00001 | 02 0001C | 01 00088, 01 00084, 01 0016A           |                              |
| REPTLEN  | R    | 00002 | 01 001F0 | 01 00088, 01 00084, 01 0016A           |                              |
| REPTEPAR | A    | 00004 | 01 001B0 | 01 00088, 01 00084, 01 0016A           |                              |
| SAFETY   | X    | 00064 | 01 00294 | 01 00088, 01 00084, 01 0016A           |                              |
| SEX      | X    | 00001 | 01 00290 | 01 00088, 01 00084, 01 0016A           |                              |
| SEX#     | X    | 00001 | 02 0001D | 01 00088, 01 00084, 01 0016A           |                              |

CROSS REFERENCE LISTING

| SYMBOL | TYPE | ING   | LOCATION | REFERENCES         |
|--------|------|-------|----------|--------------------|
| SIN    | X    | 00009 | 01 00266 | 01 0003C           |
| SIN#   | X    | 00004 | 02 00000 | 01 00046           |
| SKIP1  | I    | 00006 | 01 00088 | 01 00074, 01 0007C |
| SKIP2  | I    | 00006 | 01 00086 | 01 000AA           |
| SOURCE | X    | 0000F | 01 00256 | 01 0011C, 01 00134 |
| STATE  | X    | 00002 | 01 00287 |                    |
| STATUS | X    | 00001 | 01 00265 |                    |
| STOP   | T    | 00004 | 01 00118 | 01 00106           |

01-13-77 22:17:45

EXTERNAL SYMBOL LISTING

| SYMBOL | PROG | SYSTEM | SPRINT | WRITE | READ |
|--------|------|--------|--------|-------|------|
| SD     | 01   | 000000 |        |       |      |
| ER     | 02   |        |        |       |      |
| ER     | 03   |        |        |       |      |
| ER     | 04   |        |        |       |      |
| ER     | 05   |        |        |       |      |



## APPENDIX D

## Example Subroutine for CALL Command

05/06/77 PAGE 1 CALLEX

```

* OVERDRIVE VERSION 770504S
*
* CALLEX
*

SUBROUTINE CALLEX (NWORDS, IPINFO, NCATS, ICINFO, NRECS, IDATA,
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
)
 INTEGER, PLTPT, STRING
 !
 ! Example subroutine called by the MICRO CALL command:
 !
 ! CALL CALLEX USING RANDATA LIB=ccid:CALLOBJ PAR=5,1.2,STRING.
 !
 ! (see the CALL command description)
 !
 IMPLICIT INTEGER (A-Z)
 INTEGER*4 IPINFO(6,NWORDS), ICINFO(4,NCATS), IDATA(NWORDS,NRECS),
 !
 ! P(13), CATNAM(3), BUFFER(50), BUFP(75),
 !
 ! FILE(5) /5* ' ', STRING(2)
 !
 INTEGER*2 LPN
 REAL*8 FIELDS(2,13), NOPLD(6), PLPPT
 EXTERNAL GETPD
 DATA NOPLD / '0*** IN', 'CORRECT ', 'FIELD NA', 'NE -- ',
 !
 ! 2* ' ', /
 !
 ! CALLEX is written in IBM FORTRAN with OVERDRIVE, a pre-
 !
 ! processor which lends ALGOL-like control structures to
 !
 ! PJOPTAN. The following MTS command was used to compile
 !
 ! CALLEX:
 !
 ! $RUN *PTN SCARDS=CALLEX SPRINT=CALL.L PAR=OVER TEST LOAD=CALL.J

```

```

36. *
37. *
38. *
39. *
40. *
41. *
42. *
43. *
44. *
45. *
46. *
47. *
48. *
49. *
50. *
51. *
52. *
53. *
54. *
55. *
56. *
57. *
58. *
59. *
60. *
61. *
62. *
63. *
64. *
65. *
66. *
67. *
68. *
69. *
70. *
71. *
72. *
73. *
74. *
75. *
76. *
77. *
78. *
79. *
80. *
81. *
82. *
83. *
84. *
85. *
86. *

```

LOAD=CALL.02MP

The load module was then readied by the MTS command:

```

$COPY CALL.O+XWES:INDOBJ TO CCID:CALLOBJ

```

INDOBJ contains the object module for the INDEX routine used in CALLEX (INDEX source code follows the CALLEX listing).

Note: several MTS library routines are used in CALLEX for character manipulation, file handling, etc. These include INVC, BTD, RCALL, ADROP and READ (see MTS vol 3). If a MICRO subroutine call uses a data set, the subroutine must "know" something about the fields in the data set. This information may be obtained using the TECHDOC command.

Set up an array containing the exact names of those fields to be used by the subroutine. CALLEX uses these 13 fields in the following order:

```

DATA FIELDS / 'STUDENT-', 'ID-NO',
'NAME', 'ETHNICITY',
'SEX', 'BIRTHDAY', 'E',
'CLASS-YE', 'AR', 'YEAR-AWA', 'RDED',
'TERM-AWA', 'RDED', 'CLASS-LE', 'VEL',
'TYPE-OP-', 'AID', 'SOURCE-C', 'P-FUNDS',
'AMOUNT', 'REMARK',

```

The following code demonstrates the use of the INDEX routine. Given IPINFO, NWORDS and the field name to be INDEXED (from FIELDS), INDEX returns an index (or pointer) into the IPINFO and IDATA arrays for that field.

Note: the number of fields (13 in CALLEX) used by the subroutine need not equal NWORDS. Also, only those fields used by the subroutine (for which an index is needed) need be specified in FIELDS and INDEXED.

```

FLAG = 0
DO FOR (I = 1, 13)
 P(I) = INDEX(IPINFO, NWORDS, FIELDS(I, I))
 IF (P(I) .EQ. 0)
 * If the field name is not found, INDEX returns a zero and

```

```

14 * CALLEX issues an error message to *MSINK* for each missing
15 * field.
16 * LEN = 48
17 * CALL IMVC(16, NOFLD, 30, FIELDS(1,I), 0)
18 * CALL SERCOM(NOFLD, LEN, 0)
19 * FLAG = 1
20 * END IF
21 * END FOR
22
23 * Abort if any fields were missing.
24 * IP (FLAG.EQ. 1) RETURN
25
26 * The use of INDEX enables a level of indirection in
27 * referencing fields. A subroutine can establish its own field
28 * order in FIELDS and use it independent of the actual field
29 * ordering in the data set (ie. in IDATA).
30 * IPINPO and ICINPO tables
31
32 * The IPINPO and ICINPO arrays are support tables for IDATA.
33 * In the following section of code the contents of these tables
34 * are printed out.
35
36 WRITE(6,101)
37
38 DO FOR (I = 1,13)
39
40 * Get the index into IPINPO
41 * FIELD = F(I)
42
43 * Each entry in IPINPO contains:
44 *
45 * IPINPO(J,FIELD),J=1,4 field name (16 characters)
46 * IPINPO(5,FIELD) number of categories
47 * IPINPO(6,FIELD) pointer to the categories
48 * in ICINPO
49
50 * FIRST = IPINPO(6,FIELD)
51 * LENGTH = IPINPO(5,FIELD)
52
53 * WRITE(6,102) (IPINPO(J,FIELD),J=1,4), I, FIELD, FIRST, LENGTH
54
55 * IP (LENGTH.EQ. 0)
56
57 *
58 * The field has no categories
59 * WRITE(6,103)
60
61 * ELSE IF (I.LE. 12)
62
63 *
64 * The field has regular categories.
65 *
66 * Each entry in ICINPO contains
67

```

```

139. * ICINPO(J,CAT),J=1,3 category name (12 characters)
140. * ICINPO(4,CAT) value
141. *
142. WRITE(6,104)
143. LAST = FIRST + LENGTH - 1
144.
145. DO FOR (CAT = FIRST, LAST)
146. .
147. . VALTE = ICINPO(4,CAT)
148. .
149. . IF (I.EQ. 4)
150. . . * For the SEX field, the value is character data
151. . . . WRITE(6,105) CAT, (ICINPO(J,CAT),J=1,3), VALUE
152. . . . ELSE
153. . . . * All other fields have integer values
154. WRITE(6,106) CAT, (ICINPO(J,CAT),J=1,3), VALUE
155. END IF
156. . . . END FOR
157. . ELSE
158. . * The REMARKS* field is external. All four words of the
159. . * ICINPO entry contain the 16 character file name.
160. .
161. . WRITE(6,107) (ICINPO(J,FIRST),J=1,4)
162. .
163. . END IF
164. .
165. . END FOR
166. .
167.
168.
169.
170.
171.
172. * For an external field, the actual data is in a file.
173. * To access the data, a PDUB must be obtained for the file.
174.
175. CALL IMVC(16, FILE, 0, ICINPO(1,FIRST), 0)
176. CALL PCALL(GETPD, 2, 0, ADROP(FILE), 1, PDUB)
177.
178. IDATA matrix
179.
180. * The remainder of the CALLEX routine prints out the data in
181. * IDATA. Each record (there are NRECS of them) has 13 fields
182. * in this example. The data for each field is printed in a
183. * format determined by the type of the field. Here again the
184. * subroutine must have some information about the fields in the
185. * data set.

```

PROC 35TCATEGORY

50 185.

```

186.
187.
188.
189.
190.
191.
192.
193.
194.
195.
196.
197.
198.
199.
200.
201.
202.
203.
204.
205.
206.
207.
208.
209.
210.
211.
212.
213.
214.
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
229.
230.
231.
232.
233.
234.
235.
236.

* Given the field index in FIELD and the current record
* number in RECORD, the category name is returned in CATNAM.
* If not found, the value is converted to character form &
* returned.

FIRST = IPINPO(6, FIELD)
LAST = FIRST + IPINPO(5, FIELD) - 1
VALUE = IDATA(FIELD, RECORD)
K = -1

DO FOR (J = FIRST, LAST)
 IF (VALUE .EQ. ICINPO(4, J))
 . K = J
 . EXIT DO
 . END IF
END FOR

IF (K .LT. 0)
 . * Category not found - return the value
 . CALL IMVC(12, CATNAM, 0, BLANKS, 0)
 . CALL BTD (VALUE, CATNAM, 6, MD, ' ')
 . ELSE
 . * Category found
 . CALL IMVC(12, CATNAM, 0, ICINPO(1, K), 0)
 . END IF
END PROC

WRITE(6, 108)

* Width of the REMARKS* "window" (multiple of 4)
SIZE = 60
NPLWD = SIZE/4
SIZE = NPLWD * 4
DO FOR (RECORD = 1, NRECS)
 . WRITE(6, 115) RECORD
 . DO FOR (I = 1, 13)
 . . * Get the field index
 . . FIELD = F(I)
 . . * Print the data according to its field type
 . . DO CASE (I)
 . .

```

```

237.
238.
239. . * Types S & U -- signed & unsigned integer values
240. . * AMOUNT, STUDENT-ID-NO & YEAR-AWARDED
241.
242. . WRITE(6,109) (FIELDS(J,I),J=1,2), IDATA(FIELD,RECORD)
243.
244.
245.
246.
247. . * Type UC -- unsigned integer value with categories required
248. . * ETHNICITY, TERM-AWARDED, CLASS-LEVEL,
249. . * TYPE-OF-AID & SOURCE-OF-FUNDS
250.
251. . INVOKE GETCATEGORY
252.
253. . WRITE(6,110) (FIELDS(J,I),J=1,2), (CATNAM(J),J=1,3), VALUE
254.
255.
256.
257. . * Types UC & SC -- signed & unsigned integer values with
258. . * one default category
259. . * BIRTHDATE & CLASS-YEAR
260.
261. . VALUE = IDATA(FIELD,RECORD)
262.
263. . IF (VALUE .LE. 0)
264. . . INVOKE GETCATEGORY
265. . . WRITE(6,110) (FIELDS(J,I),J=1,2), (CATNAM(J),J=1,3), VALUE
266. . . ELSE
267. . . WRITE(6,109) (FIELDS(J,I),J=1,2), VALUE
268. . . END IF
269.
270.
271.
272.
273. . * Type C -- character data
274. . * NAME (24 characters)
275.
276. . . LAST = FIELD + 5
277.
278. . . WRITE(6,111) (FIELDS(J,I),J=1,2), (IDATA(J,RECORD),J=FIELD, LAST)
279.
280.
281.
282.
283. . * Type CC -- character data with categories
284. . * SEX (1 character)
285.
286. . . INVOKE GETCATEGORY

```

```

287. WRITE(6,112) (FIELDS(J,I),J=1,2), (CAINAM(J),J=1,3), VALUE
288.
289.
290. CASE (13)
291. * Type F -- external data
292. * REMARKS*
293.
294. WRITE(6,113) (FIELDS(J,I),J=1,2)
295.
296. * If there is no data the value is zero, otherwise it is an
297. * internal line number to use in an indexed read from the file.
298. * LINE = IDATA(FIELD,RECORD)
299.
300. IF (LINE .NE. 0)
301.
302. * CALL READ(BUFFER, LEN, 2, LINE, FDOUB)
303.
304. * The WINDOW routine is MICRO resident and is used to set
305. * the width of the "window" for the longer external data. In
306. * this example the width is set to 50 columns. The line just
307. * read is in BUFFER & the width is passed in SIZE. The
308. * windowed line is returned in BUFP and the number of lines
309. * needed to print it is returned in NINES.
310.
311. * CALL WINDOW(BUFFER, LEN, BUFP, SIZE, NINES)
312.
313. * IF (NINES .GT. 0)
314. * DO FOR (K = 1, NINES)
315. * * FINISH = K * NPLWD
316. * * START = FINISH - (NPLWD-1)
317. * * WRITE(6,114) (BUFP(J),J=START,FINISH)
318. * * END FOR
319. * * END IF
320. * * END IF
321. * * END IF
322. * * END CASE
323. * * END FOR
324. * * END FOR
325. * * END FOR
326. * * There are 3 parameters passed in addition to the data set:
327. * * INTEGER, FLTPF & STRING. Print these out.
328.
329. WRITE(6,116) INTEGER, FLTPF, (STRING(J),J=1,2)
330.
331. RETURN
332. 101 FORMAT('1',10X, 'IPINFO AND ICINFO TABLES')
333.
334. 102 FORMAT('0' / '0' /
335. | '0', 4X, 'FIELD NAME = ', 4A4/

```

```

120 336. 5X, 'CALLEX FIELD NUMBER = ', I2 /
120 337. '0', 4X, 'INDEX INTO IFINPO = ', I2 /
120 337.5 5X, 'INDEX INTO ICINPO = ', I2 /
120 338. 5X, 'NUMBER OF CATEGORIES = ', I2)
121 340.
121 341. 103 FORMAT('0', 5X, 'NO CATEGORIES FOR THIS FIELD')
122 342.
122 343. 104 FORMAT('0', 4X, 'CATEGORIES:' /
122 344. 6X, 'ICINPO INDEX', 3X, 'NAME', 7X, 'VALUE')
123 345.
123 346. 105 FORMAT(10X, I3, 5X, 3A4, 2X, A4)
124 347.
124 348. 106 FORMAT(10X, I3, 5X, 3A4, 2X, I4)
125 349.
125 350. 107 FORMAT('0', 4X, 'EXTERNAL -- FILE = ', A4)
126 351.
126 352. 108 FORMAT('1', 20X, 'IDATA MATRIX')
127 353.
127 354. 109 FORMAT(5X, 2A8, 1X, I10)
128 355.
128 356. 110 FORMAT(5X, 2A8, 1X, 3A4, 1X, 'VALUE = ', I4)
129 357.
129 358. 111 FORMAT(5X, 2A8, 1X, 6A4)
130 359.
130 360. 112 FORMAT(5X, 2A8, 1X, 3A4, 1X, 'VALUE = ', A4)
131 361.
131 362. 113 FORMAT(5X, 2A8)
132 363.
132 364. 114 FORMAT(10X, 15A4)
133 365.
133 366. 115 FORMAT('0' / '0 RECORD', I3)
134 367.
134 368. 116 FORMAT('1 3 PARAMETERS' /
134 369. '0 INTEGER = ', I4, 5X, 'PLTPT = ', P5.2, 5X,
134 370. 'STRING = ', 2A4)
135 371.
135 372. END

```

## TABLE OF CONTENTS FOR CALLEX

PAGE TITLE/SUBTITLE/PROC

C EXAMPLE SUBROUTINE FOR MICRO CALL COMMAND

5 SPEC CATEGORY

NORMAL TERMINATION IN STATEMENT 97

NUMBER OF STATEMENTS EXECUTED 26594



|                                 |       |
|---------------------------------|-------|
| EXECUTION TIME (SECONDS)        | 1.588 |
| MSEC/STATEMENT                  | 0.059 |
| NUMBER OF STORAGE REGENERATIONS | 4     |

PAGE 0001

13:28:18

05-06-77

CALLEX

TEST

G(21.9

FORTRAN

SYSTEM

2.000  
2.001  
16.000  
18.000

DATA, IDATA,

NRECS, NRECS, IDATA,

ICINPO, ICINPO, NRECS, IDATA,

INTEGER, FLTP, STRING)

INTEGER (A-2)

INTEGER\*4 IPINP(6,NWOPDS),

ICINPO(4,NCATS), IDATA(NWOPDS,NRECS),

ICINPO(4,NWOPDS),

ICINPO(4,NWOPDS),

ICINPO(4,NWOPDS),

ICINPO(4,NWOPDS),

ICINPO(4,NWOPDS),

0001

0002

0003

THE FOLLOWING IS A LISTING OF THE INDEX SUBROUTINE CALLED BY CALLEX.

The calling sequence for this function is as follows:

L=INDEX(IPINPO, NWORDS, 'field name')  
 where IPINPO and NWORDS are passed by MICRO to the called subroutine  
 and 'field name' is the location of a sixteen byte string containing  
 the name of the field which is the object of the search.  
 The field name in the sixteen byte string must be left-justified.

RTJ is an entry point for a right justification function which is not used by CALLEX.

05-03-77 15:23:13

| LOCATION | INSTRUCTION | ADDR 1   | ADDR 2 | LINE #   | SOURCE                                                         |
|----------|-------------|----------|--------|----------|----------------------------------------------------------------|
|          | 01 00000    |          |        | 1.000    | INDEX CSECT                                                    |
|          |             |          |        | 2.000 *  |                                                                |
|          |             |          |        | 3.000 *  | This is the INDEX routine. It is used to obtain a field's      |
|          |             |          |        | 4.000 *  | index into the IPINPO and IDATA matrices which are passed to a |
|          |             |          |        | 5.000 *  | subroutine by the MICRO CALL command.                          |
|          |             |          |        | 6.000 *  |                                                                |
|          |             |          |        | 7.000 *  | Source -- IWES:INDEX                                           |
|          |             |          |        | 8.000 *  | Object -- IWES:INDOBJ                                          |
|          |             |          |        | 9.000 *  |                                                                |
|          |             |          |        | 10.000 * |                                                                |
|          |             |          |        | 11.000 * | INDEX is a FORTRAN-callable function routine. It is called     |
|          |             |          |        | 12.000 * | by a FORTRAN statement such as:                                |
|          |             |          |        | 13.000 * | FINDEX = INDEX (IPINPO, NWORDS, PNAME)                         |
|          |             |          |        | 14.000 * |                                                                |
|          |             |          |        | 15.000 * | where IPINPO and NWORDS are as described in the CALL command.  |
|          |             |          |        | 16.000 * | PNAME is a region containing the full 16 character field name  |
|          |             |          |        | 17.000 * | that is to be indexed. The field's index will be assigned to   |
|          |             |          |        | 18.000 * | FINDEX.                                                        |
|          |             |          |        | 19.000 * |                                                                |
|          |             |          |        | 20.000 * |                                                                |
|          |             |          |        | 21.000   | ENTRY RTJ                                                      |
|          |             |          |        | 22.000   | STM 14,12,12(13)                                               |
|          |             |          |        | 23.000   | BAL 15,80(,15)                                                 |
|          |             |          |        | 24.000   | USING *,13                                                     |
|          |             |          |        | 25.000   | DC 18A(0)                                                      |
|          |             |          |        | 26.000   | ST 15,8(,13)                                                   |
|          |             |          |        | 27.000   | ST 13,4(,15)                                                   |
| 01 00000 | 90EC 000C   |          |        |          |                                                                |
| 01 00004 | 45F0 F050   |          |        |          |                                                                |
| 01 00008 | 01 00008    |          |        |          |                                                                |
| 01 0000B | 00000000    |          |        |          |                                                                |
| 01 00050 | 50F0 0008   | 01 00010 |        |          |                                                                |
| 01 00054 | 50D0 F004   |          |        |          |                                                                |

05-09-77 15:23:13

LOCATION INSTRUCTION ADDR 1 ADDR 2 LINF # SOURCE

|          |                |          |  |          |      |               |                          |
|----------|----------------|----------|--|----------|------|---------------|--------------------------|
| 01 00058 | 180P           |          |  | 28.000   | LR   | 13,15         |                          |
| 01 0005A | 9835 1000      |          |  | 29.000   | LM   | 3,5,0(1)      | LD PARS                  |
| 01 0005P | 5870 4000      |          |  | 30.000   | L    | 7,0(,4)       | LD # OF FIELDS           |
| 01 00062 | 4110 500P      |          |  | 31.000   | LA   | 1,15(,5)      |                          |
| 01 00066 | 000P 5000 000P | 01 00106 |  | 32.000   | TRT  | 0(16,5),BLANK | FIND BLANK AT END        |
| 01 0006C | 1815           |          |  | 33.000   | SR   | 1,5           |                          |
| 01 0006P | 4160 0001      |          |  | 34.000   | LA   | 6,1           |                          |
| 01 00072 | 1826           |          |  | 35.000   | LR   | 2,5           | CK FIRST CHARACTER       |
| 01 00074 | 0500 3000 5000 |          |  | 36.000   | CLC  | 0(1,3),0(5)   |                          |
| 01 0007A | 4780 0084      | 01 0008C |  | 37.000   | BE   | NAT1          | POINT TO NEXT FIELD NAME |
| 01 0007E | 4130 3018      |          |  | 38.000   | LA   | 3,24(,3)      |                          |
| 01 00082 | 9726 006C      | 01 00074 |  | 39.000   | BXLE | 2,6,CLOOP     |                          |
| 01 00086 | 1800           |          |  | 40.000   | SR   | 0,0           |                          |
| 01 00088 | 47P0 008E      | 01 00096 |  | 41.000   | B    | PINI          |                          |
| 01 0008C | 4410 00F8      | 01 00100 |  | 42.000   | EX   | 1,FULLCOMP    | CLC 0(0,3),0(5)          |
| 01 00090 | 4770 0076      | 01 0007E |  | 43.000   | BNE  | NOTYET        |                          |
| 01 00094 | 1802           |          |  | 44.000   | LR   | 0,2           |                          |
| 01 00096 | 58D0 0004      |          |  | 45.000   | L    | 13,4(,13)     |                          |
| 01 0009A | 98EP 000C      | 01 00014 |  | 46.000   | LM   | 14,15,12(13)  |                          |
| 01 0009E | 981C 0018      | 01 00020 |  | 47.000   | LM   | 1,12,24(13)   |                          |
| 01 000A2 | 18P            |          |  | 48.000   | SR   | 15,15         |                          |
| 01 000A4 | 07P            |          |  | 49.000   | BR   | 14            |                          |
|          |                |          |  | 51.000   | DROP | 13            |                          |
|          |                |          |  | 52.000   | RTJ  | EQU *         |                          |
| 01 000A6 |                |          |  | 53.000 * |      |               |                          |
|          |                |          |  | 54.000 * |      |               |                          |
|          |                |          |  | 55.000 * |      |               |                          |
|          |                |          |  | 56.000 * |      |               |                          |
|          |                |          |  | 57.000 * |      |               |                          |
|          |                |          |  | 58.000 * |      |               |                          |
|          |                |          |  | 59.000 * |      |               |                          |
|          |                |          |  | 60.000 * |      |               |                          |
|          |                |          |  | 61.000 * |      |               |                          |
|          |                |          |  | 62.000 * |      |               |                          |
|          |                |          |  | 63.000   |      |               |                          |
| 01 000A6 |                |          |  | 64.000   |      |               |                          |
| 01 000A6 | 0004 P03A      | 01 000E0 |  | 65.000   |      |               |                          |
| 01 000AA | 9834 1000      |          |  | 66.000   |      |               |                          |
| 01 000AF | 4110 300B      |          |  | 67.000   |      |               |                          |
| 01 000B2 | 000B 3000 P060 | 01 00106 |  | 68.000   |      |               |                          |
| 01 000B8 | 1813           |          |  | 69.000   |      |               |                          |
| 01 000BA | D20B 4000 P04P | 01 000P4 |  | 70.000   |      |               |                          |
| 01 000C0 | 4120 000C      |          |  | 71.000   |      |               |                          |
| 01 000C4 | 1821           |          |  | 72.000   |      |               |                          |
| 01 000C6 | 1A24           |          |  |          |      |               |                          |

The RTJ routine is used in MICRO to right justify category names. It is called by a PORTMAN CALL statement such as:

CALL RTJ (INBUP, OUTBUP)

where INBUP and OUTBUP are 12 byte regions. RTJ takes the left-justified string in INBUP and returns it right-justified in OUTBUP. INBUP remains unaffected.

USING \*,15  
 STM 0,4,SA  
 LM 3,4,0(1)  
 LA 1,11(,3)  
 TRT 0(12,3),BLANK  
 SR 1,3  
 MVC 0(12,4),BLKS  
 LA 2,12  
 SR 2,1  
 AR 2,4

05-09-77 15:23:13

| LOCATION                | INSTRUCTION | ADDR 1   | ADDR 2 | LINE # | SOURCE   |               |
|-------------------------|-------------|----------|--------|--------|----------|---------------|
| 01 000C8 1211           |             |          |        | 73.000 | LTR      | 1,1           |
| 01 000CA 4780 P02P      |             |          |        | 74.000 | BZ       | DONE          |
| 01 000CP 0510           |             | 01 000D4 |        | 75.000 | BCTR     | 1,0           |
| 01 000C0 4410 P034      |             | 01 000DA |        | 76.000 | EX       | 1,MVC         |
| 01 000D4 9804 P03A      |             | 01 000E0 |        | 77.000 | LH       | 0,4,SA        |
| 01 000D8 07P2           |             |          |        | 78.000 | BR       | 14            |
| MVC 0(0,2),0(3)         |             |          |        |        |          |               |
| 01 000DA D200 2000 3000 |             |          |        | 80.000 | MVC      | 0(0,2),0(3)   |
| 01 000E0 00000000       |             |          |        | 81.000 | DC       | 5A(0)         |
| 01 000F4 40             |             |          |        | 82.000 | DC       | 12C'          |
| TRI TABLE FOR BLANKS    |             |          |        |        |          |               |
| 01 00100 D500 3000 5000 |             |          |        | 84.000 | FULLCOMP | CLC           |
| 01 00106 00             |             |          |        | 85.000 | DC       | 0(0,3),0(5)   |
| 01 00146 PP             |             |          |        | 86.000 | DC       | 64X'00'       |
| 01 00147 00             |             |          |        |        | DC       | X'PP',15X'00' |
| 01 00156 00             |             |          |        | 87.000 | DC       | 176X'00'      |
|                         |             |          |        | 88.000 | END      |               |

NO ERRORS IN THIS ASSEMBLY

## CROSS REFERENCE LISTING

| SYMBOL   | TYPE | LNQ      | LOCATION | REFERENCES         |
|----------|------|----------|----------|--------------------|
| BLANK    | X    | 00001    | 01 00106 | 01 00065, 01 000B2 |
| BLKS     | C    | 00001    | 01 000F4 | 01 000BA           |
| CLOOP    | I    | 00006    | 01 00074 | 01 00082           |
| DONE     | I    | 00004    | 01 000D4 | 01 000CA           |
| FINI     | T    | 00004    | 01 00096 | 01 00088           |
| FULLCOMP | I    | 00006    | 01 00100 | 01 0008C           |
| INDEX    | J    | 00001    | 01 00000 |                    |
| MAT1     | I    | 00004    | 01 0008C | 01 0007A           |
| MVC      | I    | 00006    | 01 000DA | 01 000D0           |
| NOTYPE   | I    | 00004    | 01 0007E | 01 00090           |
| RTJ      | T    | 00001    | 01 000A6 |                    |
| SA       | A    | 00004    | 01 000P0 | 01 000A6, 01 000D4 |
| INDEX    | SD   | 01 00000 | 000206   |                    |
| RTJ      | LD   | 000A6    | 01       |                    |



## APPENDIX F

Error Messages

General error messages - those which may be produced by more than one MICRO command.

'COMMAND CANCELLED.'

Usually preceded by another error message; indicates that no action was taken by the command.

'COMMAND MAY NOT EXCEED 500 CHARACTERS.'

Indicates that more than 500 characters were entered for a MICRO command.

'DATA SET DOES NOT EXIST OR IS NOT AVAILABLE.'

A request was made for a data set that either does not exist or for which the user does not have proper MTS permit access.

'ERROR: DATA AND DICTIONARY ARE NOT COMPATIBLE.'

Indicates that the dictionary specifies a longer record than exists in the data.

'ERROR: DATA FILE DOES NOT EXIST.'

Indicates that the data set specified refers to a data file which does not exist.

'ERROR: DATA SET IS NOT A PERMANENT SET.'

Indicates that a request for a permanent operation was made on a temporary set.

'ERROR: DICTIONARY FILE DOES NOT EXIST.'

Indicates that the data set refers to a dictionary file which does not exist.

'ERROR IN READING DIRECTORY.'

Indicates that an unspecified I/O error occurred while reading the specified directory.

'ERROR IN SYNTAX.'

Indicates that an unspecified illegal formulation of a command was entered.

'ERROR: INVALID DICTIONARY FILE.'

The specified dictionary file does not contain a valid MICRO dictionary.

'ERROR: NO DATA SET SPECIFIED.'

Usually indicates that the phrase "IN <DSN>" has been omitted.

'ERROR: NO FIELDS WERE SPECIFIED.'

Indicates that the command was incomplete, in that no fields were supplied where they were required.

'ERROR: READ ACCESS TO DATA FILE NOT ALLOWED.'

Indicates that the user does not have MTS read access to the data file.

'ERROR: READ ACCESS TO DICTIONARY NOT ALLOWED.'

Indicates that the user does not have MTS read access to the dictionary file.

'ERROR: SOFTWARE OR HARDWARE ERROR.'

Indicates that while reading a data or dictionary file a software or hardware error was encountered.

'ERROR: UNABLE TO READ DICTICNABY.'

Indicates that an unspecified I/O error occurred while reading the specified dictionary.

'ERROR: UNABLE TO READ DIRECTCRY.'

Indicates that the specified directory does not exist or that the user does not have MTS read access to it.

'FILE NCT AVAILAEIE.'

Indicates that the MTS file specified does not exist or that the user lacks proper MTS access.

'INVALID COMMAND GENERATED FROM A MACRO - COMMAND IGNORED.'

Indicates that a macro generated an invalid MICRO command.

'MICRO PROGRAM INTERRUPT'

'PSW = ????????? ???? Etc.'

'PROCEED AT YOUR CWN RISK!!!!'

Indicates that a program interrupt has occurred. The output should be sent to ILIR to determine the cause of the problem. MICRO will return the user to command mode to enable the user to attempt to SAVE any temporary sets that are of value. Depending upon what has caused the program interrupt, it may not be possible to SAVE anything. Furthermore, it is highly recommended that a minimum of processing be done before rerunning MICRO.

'MTS RUN COMMAND IS NOT ALLOWED!'

An MTS run command was given and disallowed.

'NULL CCMAND.'

'NUMBER OF SYMBOLS EXCEEDS MAX ALLOWED.'



All MICRO commands are limited to a maximum of 100 symbols.

'SERIOUS SYSTEM ERROR - PLEASE DISCONTINUE PROCESSING.'

Indicates that a bug occurred in the system and all output should be sent to the Institute of Labor and Industrial Relations.

'TAPE CANNOT BE MOUNTED.'

The specified MICRC data set is stored on tape and the tape cannot be mounted.

'THAT FIELD HAS NO CATEGORIES.'

Indicates that a request for categorical information was requested for a field that has no categories.

'UNBALANCED PRIMES - YOU NEED TWO PRIMES (' ').'

'YOU HAVE NO DATA SETS.'

The user has given a command which requires inclusion of a data set and he has none.



## INDEX

| SYMBOL 138  
 & SYMBOL 138  
 \$ FACILITY 279  
 \* 153  
 ; SYMBOL 138  
 % FACILITY 281

@% 161  
 @ABBREVIATION 162,237,238,240,245  
 @COUNT 162,237,240,241,246  
 @DATE 162,238,240,245  
 @DESCRIPTION 162,237,238,240,241  
 @FIRST 162,237,240,241  
 @FORWARD 162,238,240,241  
 @GF 217,270  
 @HEXADECIMAL 162,238,240,245  
 @HORIZONTAL 162,238,240,242  
 @INCREMENT 162,238,240,241  
 @INDIRECTION 162,238,240,245  
 @KC 155  
 @NO% 161  
 @NOPRINT 162  
 @NSC 155  
 @NV 167  
 @PAGE 162,238,240,247  
 @PC 162,238,240,247  
 @PD 162,238,240,247  
 @PRINT 161  
 @REVERSE 162,238,240,241  
 @SSN 162,238,240,245  
 @STRATIFICATION 162,238,240,243  
 @VALUE 162,238,240,245  
 @VERTICAL 162,238,240,242  
 @WINDOW 162,238,240,245  
 @X 162,238,240,245

ABSTRACIION OF DATA COLLECTICNS 31  
 ABSTRACTION OF DATA ELEMENTS (FIELDS) 32  
 ABSIRACTION OF DATA 31  
 ACKNOWLEDGEMENTS 15  
 ADVANCED DESIGN CONCEPTS 52  
 ALSO WHERE SYMBOL 138  
 ALSO WHERE 204  
 ALTER 147  
 AND SYMBOL 138  
 AND 204

APPEND (MICRO.DIR) 105  
 APPEND COMMAND (MICRC.DEF) 73  
 APPLICABLE COMMAND MODIFIERS SECTION OF MICRO  
 COMMAND DESCRIPTIONS 140  
 ATTENTION INTERRUPTS 134  
 AVE 161,163  
 AVE. 163  
 AVERAGE 161,163  
  
 BATCH FIXED-FORMAT INPUT 112  
 BATCH FREE-FORMAT DATA ENTRY 110  
 BEFORE 203  
 BUILDING AND USING MICRO DATA SETS 23  
  
 CALL 143  
 CARD NUMBER (FIXED-FORMAT DATA ENTRY) 117  
 CAT SYMBOL 138  
 CATEGORICAL INFORMATION 50  
 CATEGORIES SYMBOL 138  
 CATEGORIES 68,70  
 CATEGORY ABBREVIATION 69  
 CATEGORY DESCRIPTION 52,70  
 CATEGORY NAME AND ABBREVIATION 50,69  
 CATEGORY REQUIRED STATUS 50,68  
 CATEGORY SYMBOL 138  
 CATEGORY TYPE 51,69  
 CATEGORY VALUE 51,69  
 CHANGE COMMAND (MICRC.DEF) 74  
 CHANGE 147  
 CHANGING THE DATA SET DEFINITION - CORRECTION MODE 71  
 CHARACTER DATA 46  
 CHARACTER FIELD LENGTH 67  
 CHARACTER STRING FIELDS WITH CATEGORIES 47  
 CHARACTER STRING FIELDS WITHOUT CATEGORIES 47  
 CHECKPOINT AND RECOVERY PROCEDURE 135  
 CLOCK FACILITY 263,279  
 CLOSE (MICRO.FORM) 121  
 CLOSE COMMAND (MICRC.DEF) 76  
 CNT FACILITY 279  
 CODIFICATION OF DATA VALUES 49  
 COMBINE 151  
 COMMAND AND FIELD MODIFIERS 141  
 COMMAND DESCRIPTIONS - FORMAT 139  
 COMMAND SYMBOL 138  
 COMMAND SYNONYMS SECTION OF MICRO COMMAND DESCRIPTIONS 139  
 COMMAND TIME FACILITY 279  
 COMMAND TIME LIMITS 134  
 COMMAND TRACE FACILITY 28  
 COMMANDS SYMBOL 138  
 COMMENT 153  
 COMMENTS IN THE MACRO DEFINITION 311

COMMENTS SECTION OF MICRO COMMAND DESCRIPTIONS 140  
 COMPOSITION STYLE 16  
 COMPUTE 155  
 CONCEPTUALIZATION OF DATA MANAGEMENT NEEDS 26,31  
 CONDITIONAL MACRO PROCESSOR 28  
 CONSTRAINTS ON FIXED-LENGTH CHARACTER STRING FIELDS AND SOLUTIONS 48  
 CONVERSION FILE NAME 41,64  
 CONVERSION MODE - FIXED-FORMAT 128  
 CONVERT (MICRO.FORM) 122  
 CORRECTION MODE - FIXED-FORMAT 120  
 COST FACILITY 263,279  
 COUNT FIELD 23  
 CPU TIME FACILITY 279  
 CREATING DICTIONARIES FOR THE EXAMPLE SETS 83  
 CROSSTABULATE 161  
 CROSSTABULATED DATA SET INDICATOR 41,65  
 CROSSTABULATED DATA SETS 22  
 CUSTOM PROGRAMMING DATA ENTRY 129

DATA ENTRY PROCESS 26  
 DATA ENTRY 109  
 DATA FILE NAME 40,64  
 DATA FILE 27  
 DATA FILES, DICTIONARIES AND DIRECTORIES 19  
 DATA REQUIRED STATUS 68  
 DATA SET DEFINITION 59  
 DATA SET DESCRIPTION 40,64  
 DATA SET DESIGN 26  
 DATA SET DESTRUCTION STATUS 40,64  
 DATA SET DOCUMENTATION 27  
 DATA SET NAME 39  
 DATA SET REPLACEMENT STATUS 40,64  
 DATA SET REQUIREMENTS 39  
 DATA SET SCRAMBLE STATUS 65  
 DATA SET SYMBOL 138  
 DATA SETS SYMBOL 138  
 DATA SETS, RECORDS, FIELDS, VALUES AND CATEGORIES 19  
 DATA 26  
 DATASET SYMBOL 138  
 DATASETS SYMBOL 138  
 DATESTAMP 167  
 DECIMAL NUMBERS 45  
 DEF 313  
 DEFAULT VALUES AND REQUIRED DATA 52  
 DEFINE 313  
 DEFINING EACH FIELD 42  
 DEFINING A DATA SET 59  
 DEFINITION OF THE FIN-AID DATA SET 84  
 DEFINITION OF THE STUDENTS DATA SET 96  
 DELETE (MICRO.DIR) 105  
 DELETE (MICRO.FORM) 124

DELETE COMMAND (MICRO.DEF) 77  
 DELIMITER LIST 314  
 DESCRIBE 169  
 DESCRIPTIONS AND NAMES 20  
 DESTROY 171  
 DICTIONARY FILE NAME 41,62  
 DICTIONARY USECOUNT 42,65  
 DICTIONARY 26  
 DIFFERENCES BETWEEN SIGNED AND UNSIGNED INTEGERS 45  
 DIRECTORY ENTRY CREATION AND MODIFICATION 105  
 DIRECTORY 26  
 DISPLAY 173  
 DOCUMENT (MICRO.FORM) 123  
 DOCUMENT COMMAND (MICRO.DEF) 78  
 DOCUMENT 175  
  
 ECHO FACILITY 263,279  
 EFFICIENCY OF THE CHANGE AND UPDATE COMMANDS 131  
 END DEFINITION 315  
 END 179  
 ENTERDATA 181  
 EQUAL 147,203  
 ERROR MESSAGES SECTION OF MICRO COMMAND DESCRIPTIONS 140  
 EXAMPLE OF CROSSTABULATED DATA SET 22  
 EXAMPLES SECTION OF MICRO CCMAND DESCRIPTIONS 140  
 EXPLAIN 197  
 EXTERNAL DATA FIELDS 48  
 EXTERNAL DATA FILE NAME 67  
  
 FDOC 211  
 FIELD ABBREVIATION 42,66  
 FIELD DESCRIPTION 43,68  
 FIELD NAME 42,66  
 FIELD SIZE - EXTERNAL DATA FIELDS 49  
 FIELD SIZE OF CHARACTER STRING FIELDS 47  
 FIELD SIZE OF INTEGER FIELDS 44  
 FIELD SYMBOL 138  
 FIELD TYPE 43,66  
 FIELDS SYMBOL 138  
 FIELDS 65  
 FIND 199  
 FIXED LENGTH CHARACTER FIELDS 47  
 FIXED-FORMAT DATA ENTRY 114  
 FULLDOC 211  
  
 GB FACILITY 279  
 GBLOCK FACILITY 238,240,246 263,279  
 GENERAL DATA SET INFORMATION 61  
 GET 215  
 GOTO 316  
 GREATER THAN OR EQUAL TO 203

GREATER THAN 203

HCC FACILITY 238,240,247,263,280  
 HIRCC FACILITY 238,240,246,263,280  
 HISTORY 14  
 HORIZONTAL PRINT FACILITY 238,240,242,263,280  
 HP FACILITY 279

I FACILITY 280

IF 317

IMPLICIT EFFECT ON THE RESULT SET SECTION OF MICRO  
 COMMAND DESCRIPTIONS 140

INCLUDING OR EXCLUDING CERTAIN RECORDS 115

INFORMATION PERTAINING TO DATA IN EACH FIELD - FIXED-FORMAT 116

INITIALIZATION OF DATA SETS 33

INPUT ECHO FACILITY 280

INPUT FACILITY 263,279

INPUT TYPE (FIXED-FORMAT DATA ENTRY) 117

INSERT COMMAND (MICRC.DEF) 79

INTEGRATION OF DATA SETS 35,52

INTERACTIVE PROMPTING DATA ENTRY 109

IS 147,203

IT SYMBOL 138

JOIN 217

KEY 221

LENGTH (FIXED-FORMAT DATA ENTRY) 118

LESS THAN OR EQUAL TO 204

LESS THAN 203

LIST 223

LOCAL COMMAND TRACE FILE NAME 63

LOGICAL DESIGN 26

LPP FACILITY 238,240,247,263,280

MACRO COMMAND SYNONYMS 311

MACRO COMMANDS 310

MACRO COMMAND 318

MACRO DEFINITIONS 309

MACRO ECHO FACILITY 263,281

MACRO EXAMPLE 321

MACRO LIBRARIES 309

MACRO PROCESSOR 309

MACRO VARIABLE SYMBOLS 310

MAINTENANCE OF DATA INTEGRITY 135

MANUAL USE 15

MASTER COMMAND TRACE FILE NAME 63

MASTER ID FACILITY 281

MASTER SIGNON-ID 28,62

MATCHES 147,200,204

MAX 161,163  
 MAX. 163  
 MAXIMUM NUMBER OF CARDS IN ONE RECORD 114  
 MAXIMUM TOTAL NUMBER OF COLUMNS ON ALL CARDS 114  
 MAXIMUM VALUE OF A FIELD 67  
 MAXIMUM 161,163  
 MEAN 161,163  
 MESSAGE 227  
 MICRO COMMAND DESCRIPTIONS 139  
 MICRO COMMAND LANGUAGE 137  
 MICRO COMMAND PROCESSOR 27  
 MICRO COMMANDS 27  
 MICRO DATA SET NAME 63  
 MICRO PROGRAMS 27  
 MICRO 28,133  
 MICRO.ADD 29,110,187,189  
 MICRO.CNVRT 128  
 MICRO.DEF PROGRAM 26,29,59  
 MICRO.DIR 29,105  
 MICRO.DOC 29,106,176,212,290  
 MICRO.FCRM 111  
 MICRO.UP 29,297  
 MIN 161,163  
 MIN. 163  
 MINIMUM 161,163  
 MISSING DATA VALUE (FIXED-FORMAT DATA ENTRY) 118  
 MODEL 319  
 MODIFY COMMAND (MICRO.DEF) 80  
 MODIFYING A DATA SET 34,131  
 MODIFYING VIA THE CHANGE COMMAND 131  
 MODIFYING VIA THE COMPUTE COMMAND 132  
 MODIFYING VIA THE UPDATE COMMAND 131  
 MTS 229  
  
 NAME 231  
 NATURAL AND CODIFIED DATA 32  
 NOP 320  
 NOT EQUAL 203  
 NOTATION USED IN THIS MANUAL 137  
 NOTEWORTHY USAGE STRATEGIES SECTION OF MICRO  
     COMMAND DESCRIPTIONS 140  
 NUMBER 221  
 NUMERICAL AND CHARACTER DATA 33  
 NUMERICAL DATA 43  
  
 OPERATING ENVIRONMENT 13  
 OR SYMBOL 138  
 OR WHERE SYMBOL 138  
 OR 204  
 ORDER 287  
 ORDERING OF RECORDS IN A DATA SET 21



ORL FACILITY 238,240,241,242,281  
 OUTPUT FACILITY 238,240,241,263,281  
 OVERLAP COLUMN 67  
 OVERLAP LENGTH 67  
 OVERLAPPING FIELDS 55

PC FACILITY 238,240,247,263,281  
 PCC FACILITY 238,240,246,247,263,281  
 PDATE FACILITY 238,240,247,281  
 PERCENT FACILITY 161,263,281  
 PERCENT FIELD 23  
 PERMANENT AND TEMPORARY SETS AND THE RESULT SET 20  
 PHYSICAL DESIGN PROCESS 26  
 PLOT 235  
 PRECISION FACILITY 283  
 PREFIX CHARACTERS 133  
 PRINT 237  
 PRINT DIRECTION FACILITY 282  
 PRINTDIRECTION FACILITY 238,240,241,282  
 PROTOTYPE SECTION OF MICRO COMMAND DESCRIPTIONS 139  
 PURGE 251  
 PURPOSE SECTION OF MICRO COMMAND DESCRIPTIONS 139  
 PVALUE FACILITY 238,240,245,282

R 255  
 RAJ 269  
 RAM 269  
 READ 253  
 READY FACILITY 263,282  
 RECODE (MICRO.FORM) 125  
 RECODE TYPE (FIXED-FORMAT DATA ENTRY) 118  
 RECOMMENDED READING 16  
 REDEFINE COMMAND (MICRO.DEF) 81  
 REDO (MICRO.DEF) 81  
 REDO (MICRO.DIR) 106  
 REDO (MICRO.FORM) 126  
 RELEASE 255  
 REMOVE 257  
 RENAME 231  
 REPEATING GROUPS 33,54  
 REPLACE 259  
 REPORTS AND ANSWERS 27  
 REPRESENTATION OF LARGE NUMBERS 46  
 REPRESENTING CHARACTER STRING DATA AS CATEGORIES 49  
 REQUIRED CATEGORIES 50,68  
 RESET 263  
 RESTRICT 265  
 RESTRICTANDJOIN 269  
 RESULT SYMBOL 138  
 RETRIEVAL OF INFORMATION FROM DATA SETS 34  
 RUNFILE FACILITY 282

RUNFILE NAME 63  
 RUNFILE 28  
 RUNNING MICRO ON MTS 133  
  
 S.D. 163  
 SAMPLING STRING 116  
 SAVE 273  
 SCALING BY ADDITION AND SUBTRACTION 46  
 SCALING BY DIVISION 45  
 SCALING BY MULTIPLICATION 46  
 SCALING CONSTANT 68  
 SCALING FUNCTION 67  
 SCALING OVERFLOW 46  
 SCALING 45  
 SCRAMBLED DATA SET INDICATOR 41  
 SD 161,163  
 SECURITY FEATURES AND DATA INTEGRITY 134  
 SELECT 275  
 SET 279  
 SHARED CATEGORIES 52  
 SHOW (MICRO.DEF) 78  
 SHOW (MICRO.FORM) 123  
 SIGNED INTEGER FIELDS WITH CATEGORIES 44  
 SIGNED INTEGER FIELDS WITHOUT CATEGORIES 44  
 SIGNOFF 285  
 SINCE 203  
 SO THAT SYMBOL 139  
 SORT 287  
 SPECIAL DESIGN CONSIDERATIONS SECTION OF MICRO  
     COMMAND DESCRIPTIONS 140  
 SPECIFYING A COMMAND 137  
 ST SYMBOL 139  
 STARTING COLUMN (CR SUBROUTINE FILE) (FIXED-FORMAT DATA ENTRY) 116  
 STD 161,163  
 STD.DEV 161,163  
 STOP (MICRO.DIR) 106  
 STOP (MICRO.FORM) 127  
 STOP COMMAND (MICRO.DEF) 82  
 STOP 179  
 SUCH THAT SYMBOL 139  
 SUM 161,163  
 SYNONYMS ACCEPTABLE IN MICRO 138  
 SYSTEM 229  
  
 TABULATE 161  
 TAPE DATA SETS 107  
 TCC FACILITY 238,240,247,263,264,282  
 TDOC 289  
 TECHDOC 289  
 TITLE FACILITY 238,240,247,264,282  
 TOT 161,163

TOT. 163  
TOTAL 161,163  
TRANSFORMING THE LOGICAL DEFINITION INTO A PHYSICAL DESIGN 39  
TURN 279

UNSIGNED INTEGER FIELDS WITH CATEGORIES 44  
UNSIGNED INTEGER FIELDS WITHOUT CATEGORIES 43  
UNSIGNED INTEGERS 45  
UNTIL 204  
UPDATE 291  
USE 305  
USECOUNT 42,65  
USER DECISION PROCESS 27  
USER-SPECIFIED BATCH FIXED-FORMAT DATA ENTRY 111

VARYING LENGTH CHARACTER FIELDS 48  
VARYING NUMBER OF CARDS PER RECORD 114  
VERTICAL PRINT FACILITY 239,240,242,264,283  
VIRCC FACILITY 239,240,246,247,263,264,283  
VP FACILITY 283

W SYMBOL 138  
WHAT HAPPENS SECTION OF MICRC COMMAND DESCRIPTIONS 140  
WHERE SYMBOL 138  
WINDOW FACILITY 239,240,245,264,283  
WORKSHEET FOR THE FIN-AID DATA SET 57  
WORKSHEET FOR THE STUDENTS DATA SET 58  
WRITE 307  
WRITING MACRO-DESCRIPTIONS/COMMANDS 311

XTAB (%) PRECISION 161,264,283  
XTAB PERCENT FACILITY 282  
XTAB PRINT FACILITY 161,162,264,283  
XTAB 161  
XTAB% FACILITY 281  
XTABPRINT FACILITY 283