REALITY® by Microdata

English™ Programming Manual

REALITY®

ENGLISH®
Programming Manual

791073

Series 3.0
Release 3.2

July 1979

For all Series 3.0 Reality systems thru Release 3.2

Printed in U.S.A.
Price: 15.00

Microdata Corporation
17481 Red Hill Avenue, Irvine, California 92714
Post Office Box 19501, Irvine, California 92713
Telephone: 714/540-6730 • TWX: 910-595-1764

FOREWORD

This book was produced by "Baby", one of our Reality® computer systems using RUNOFF™, Microdata's word processing program. Printer output pages were used as printing masters.

RUNOFF includes elements of the publishing style, e.g., page size, justified right and left margins, headings, italics, pagination, centering, tabular illustrations, etc. The contents and index are automatically generated each time the document is printed, reflecting any changes made. Line art and photos have been added in spaces, or "windows", left for this purpose by RUNOFF.

Because the complete book is stored on disc, it is easy to update the document. The stored text is corrected using the EDITOR, and a new printout is made of only the changed pages. The corrected pages are then substituted for the old pages in the printing masters.

A NOTE REGARDING REALITY DOCUMENTATION

Because these Series 3.0, Level 3.2 publications represent a new step for Microdata, we are using a "fresh start" approach. This document is considered as the starting point for future updates. When necessary, supplements to the manuals will be available and will usually coincide with software releases. Supplements will be printed on different colored stock to enable easy identification and use.

Users upgrading from a Series 2.0-2.5 software operating system are advised to thoroughly reread the Reality manuals since most sections have been updated in some way. This reeducation becomes increasingly important as time goes by, since future system enhancements will add capabilities and features that differ significantly from Series 2.0 systems. Microdata is committed, however, to ensuring users of an easy upgrade path that does not require major modifications to application software.

CONTENTS

1 INTRODUCTION

1.1 Reality® Computer System

Reality® is a generalized, data base management computer system. It is a complete system providing multiple users with the capability to instantly update and/or retrieve information stored in on-line data files. Users communicate via local or remote terminals with computer files that may be private, common, or security-controlled. Each terminal user's vocabulary can be individually tailored to specific application jargon.

Reality is built of field-proven Microdata computers and peripherals, utilizing microprograms to provide users with unrivaled performance and reliability in the medium-sized computer market.

The Reality computer system includes the powerful, yet simple to use, ENGLISH® inquiry language and the DATA/BASIC™ and PROC high-level languages, file maintenance tools, and EDITOR processor, complete programming development facilities, and a host of other user amenities. Reality runs in an on-line, multiuser environment with all system resources and data files efficiently managed by a miroprogrammed virtual memory operating system.

Reality has advantages from every angle: system capability, multiuser performance, file management languages, ease of programming, data structure, and architectural features. Reality's high performance and fast response time are made possible by extensive use of high-speed microprocessors which greatly reduce program execution time and system overhead. The entire Reality computer system is unique -- one of a kind.

Microprogrammed firmware contains:

- Vitual memory manager
- Multiuser operating system
- Special data management instructions
- Input/output processors

System software includes:

- ENGLISH, DATA/BASIC, PROC, EDITOR, and Assembly languages
- Selectable/automatic report formatting
- Dynamic file/memory management
- RUNOFF™ word processing
- New SCREENPRO™ language -- an easy way to set up terminal displays
- Optional BISYNC communications

The file structure provides:

- Variable length files/records/fields
- Multivalues (and subvalues) in a field
- Efficient storage utilization
- Fast access to data items
- Selectable degrees of data security
- File size limited only by size of disc
- Record size up to 32,267 bytes

1.2 The Flexible Family of Computer Systems

The expanded Reality family of high-performance data base management processors ranges from an economical system for first-time users with limited data processing requirements and/or experience, to the high capacity systems used by some of the largest companies in the United States.

Besides superb performance, the entire Reality line offers unmatched growth advantages. As a user company grows, it can add Reality equipment to meet its increased data processing needs without the costly replacement and conversion charges usually associated with updating computer facilities. All Reality systems are both hardware and software compatible. Start with Reality. Grow with Reality.

A typical basic system has:

- Central processing unit (CPU and cabinet)
- Mass storage disc drive
- Magnetic tape drive
- PRISM™ cathode ray tube (CRT) data terminal (up to 32)
- System printer

All Reality systems operate in Microdata's easy-to-use ENGLISH retrieval language, as well as the more advanced DATA/BASIC and PROC, and are fully compatible with other Reality data processing systems.

There is a high performance Reality system designed for the small to medium-sized company just entering computerized data base management. This system is a low-cost, efficient way to start. The computer and all peripherals are totally compatible with Microdata's complete Reality line. This system has a special extended performance feature for future expansion.

At the top of the Reality line is Microdata's most advanced microprocessing technology. Greater load capacity. Still faster data processing. More applications. All without overloading the central processing unit or degrading the speed of terminal response. The advanced system's exceptional power and adaptability provides up to 32 separate users with fingertip access to voluminous business data and any other business, technical, or scientific applications that utilize data base management techniques.

- Complete small business computer capabilities
- Microprogrammed virtual memory operating system
- Up to 32 users and 600 million characters of file storage
- On-line file update/retrieval
- ENGLISH retrieval language
- Fast terminal response
- Printer spooling
- Optional communication capabilities
- Special data management processors
- High-speed generalized sort capability
- Small computer price
- Big computer performance
- Computer/peripheral compatibility from top to bottom of the entire Reality family

Figure A. Reality System Advantages

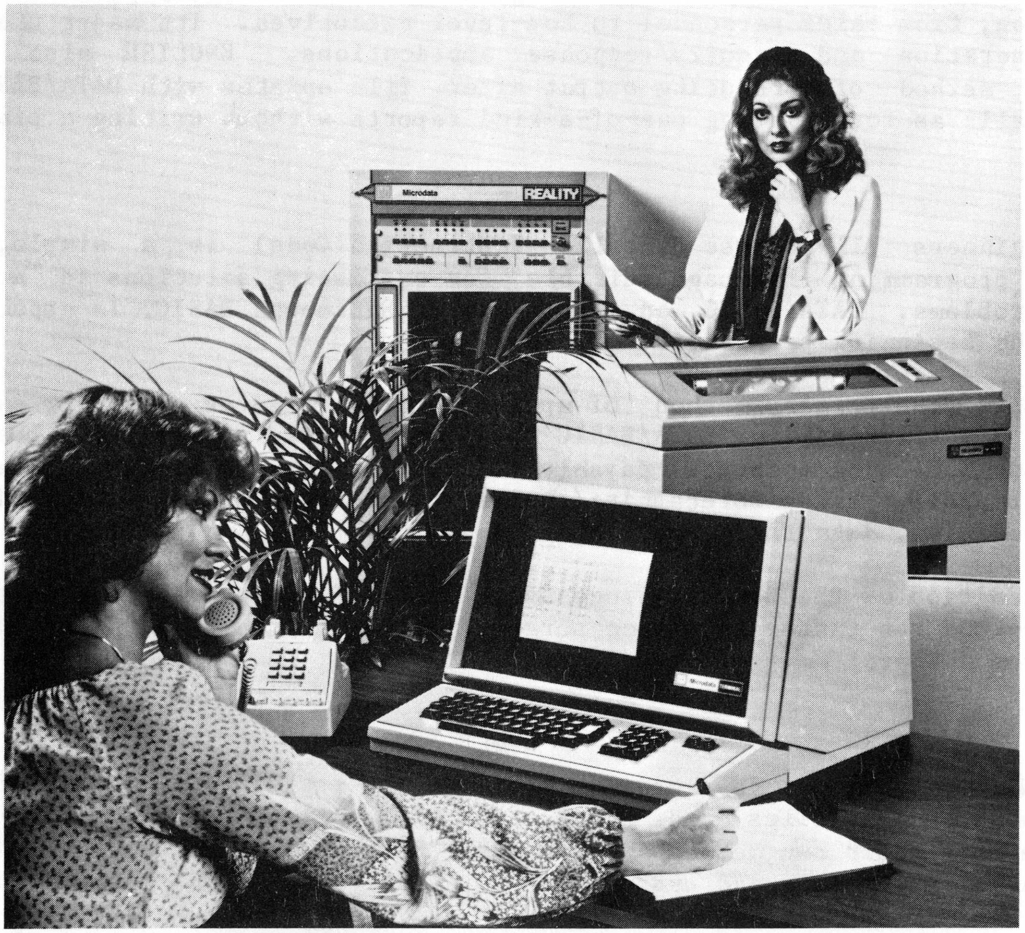


Figure B. Typical Microdata Reality System

1.3 Reality Software

Processors available on the Reality computer system comprise the most extensive data base management software available on any minicomputer. Overviews of the software processors and their typical uses follow.

ENGLISH Language

ENGLISH is a generalized data retrieval/report generator language. A typical ENGLISH inquiry consists of a relatively free-form sentence containing appropriate verbs, file-names, data selection criteria, and control modifiers. An easy-to-use, dictionary-based language that employs simplified prose statements, ENGLISH permits the user to produce original reports rapidly and efficiently.

ENGLISH applications are limitless because of the ease with which output can be accessed from user files. Since nonprogrammers can master the process quickly, ENGLISH is a valuable information management tool for many people in an organization, from sales personnel to top-level executives. Its major uses are report generation and inquiry/response applications. ENGLISH also is a convenient method of producing output after file updates with DATA/BASIC or PROC, as well as for printing one-of-a-kind reports without writing a program.

DATA/BASIC

BASIC (Beginners All-purpose Symbolic Instruction Code) is a simple, yet versatile, programming language suitable for expressing solutions to a wide range of problems. DATA/BASIC, an extension of Dartmouth BASIC, is especially easy for the beginning programmer to learn.

DATA/BASIC is the primary method of updating user files on a Reality system. Because of its flexibility, DATA/BASIC is used for a variety of business applications including accounts payable/receivable, general ledger, inventory control, payroll, sales forecasting/analysis, order processing, invoicing, claims processing, data entry, and other projects.

With the addition of the Screen Processor, DATA/BASIC programs are even easier to write -- and run faster -- since screen handling and data validation can be removed from the program.

SCREENPRO

The SCREENPRO program was developed to minimize the software gap between the establishment of data files and the creation of reports. No longer do users have to develop their own methods of creating and processing screens to display text, inputs, validations and updates.

Because SCREENPRO requires fewer program statements, it greatly simplifies program maintenance while increasing operator and programmer efficiency. Data throughput is accelerated. A screen can be designed, displayed, tested and changed without affecting the program.

PROC

The PROC (stored procedure) processor enables the user to prestore a complex sequence of operations which can then be invoked by a single word command. Any sequence of operations that can be executed from the terminal can be prestored in a PROC.

PROC is similar to the Job Control Language (JCL) used in larger computer systems, but PROC is less cryptic and has far greater capabilities including interactive (optionally formatted) terminal prompting, input validation, printer formatting, and file input/output.

PROCs are typically used to create special user-defined functions by combining execution of DATA/BASIC programs, ENGLISH data retrieval operations, and PROC argument passing.

EDITOR

The EDITOR permits on-line interactive modification of any item in the data base.

Primarily, the EDITOR is used to create and/or modify DATA/BASIC or PROC programs. The EDITOR enters and updates text processed by RUNOFF. Particularly useful in word processing is the EDITOR's global search and replace capabilities. Performing one-of-a-kind modifications to items in user files is another EDITOR function.

RUNOFF

RUNOFF is a word processing facility offering many special features. RUNOFF processes text entered and modified with the EDITOR. RUNOFF numbers pages automatically and can print text headings and footnotes.

Another RUNOFF feature is chapter and section numbering. New chapters and/or sections may be added to a document, and the subsequent updated publication, with changes and additions, will be completely renumbered automatically. RUNOFF assembles and prints a table of contents covering all subjects, including corrected/updated copy.

RUNOFF also automatically assembles a publication index based on specified words and phrases. RUNOFF supplies index page numbers. If new pages are added, the index is automatically updated.

RUNOFF also performs tabulations, centering, selective left/right justifications, underlining, and boldface printing.

This and all Reality user manuals were produced by RUNOFF on "BABY", a Reality computer system.

1.4 How to Use the Reality Manuals

This manual is written in modular format with each pair of facing pages presenting a single topic.

The approach taken in this and other Reality manuals differs substantially from the typical reference manual format. Here, each pair of pages discusses an individual topic. Generally the left-hand page is devoted to text, while the right-hand page presents figures referred to by the text. A pair of titles, the first naming the chapter and the second naming the topic, are at the head of each text page. Immediately below these titles is a brief summary of the material covered in the topic.

The advantage of this format will become readily apparent as the reader begins to use this manual. First of all, the figures referred to in the text are always conveniently right in front of the reader at the point where the reference is made. Secondly, there is a psychological advantage to the reader knowing that when he has completed reading a topic and turns the page he is done with one idea and is ready for a new one.

Documentation for the Reality system includes the following manuals:

- Introduction to Reality
- Programmer's Reference Manual
- EDITOR Programming Manual
- ENGLISH Programming Manual
- DATA/BASIC Programming Manual
- PROC Programming Manual
- SCREENPRO Programming Manual
- ASSEMBLY Language Programming Manual
- BISYNC Programming Manual

IMPORTANT NOTE: the user should thoroughly read the manual titled INTRODUCTION TO REALITY prior to referencing this manual!

In presenting general command formats and examples throughout this and other Reality manuals, certain conventions apply. Conventions used in presenting general command formats are listed in Figure A, while conventions used in examples are listed in Figure B.

Marginal change bars will be used in future manuals and supplements for the convenience of present Reality users and will indicate significant additions or changes from prior Reality publications.

Convention	Meaning
UPPER CASE	*Characters or words printed in upper case are required and must appear exactly as shown.*
lower case	*Characters or words printed in lower case are parameters to be supplied by the user (e.g., file-name, item-id, data, etc.).*
{ }	*Braces surrounding a word and/or a parameter indicates that the word and/or parameter is optional and may be included or omitted at the user's option.*
{ }...	*If an elipses (i.e., three dots) follows the terminating bracket, then the enclosed word and/or parameter may be omitted or repeated an arbitrary number of times.*
item-list*	*An asterisk following an item-list indicates that the list of item-ids may be omitted if supplied by a previous SELECT, SSELECT, GET-LIST, or FORM-LIST command.*

Figure A. Conventions Used in General Command Formats

Convention	Meaning
TEXT	*Shaded text represents the user's input.*
TEXT	*All other text represents output printed by the system.*
TEXT	*Italicized text is used for comments and notes which help explain or describe the example.*
<cr>	*This symbol represents a carriage return.*
<lf>	*This symbol representsa line feed.*
<c>	*This symbol specifies that the following character is a control character generated by depressing the <CTRL> key while typing the character. Also depress the <SHIFT> key if the character appears on the upper half of a key top.*
_	*This is the ASCII underline character represented as a backarrow (-) on some terminals.*

Figure B. Conventions Used in Examples

1.5 ENGLISH Language

ENGLISH is a user oriented data retrieval language for accessing files within the Reality computer system.

ENGLISH is a generalized information management and data retrieval language. A typical ENGLISH inquiry (called an ENGLISH input sentence) consists of a relative free-form sentence containing appropriate verbs, file names, data selection criteria, and control modifiers. Each user's vocabulary can be individually tailored to his application jargon.

ENGLISH is a dictionary-driven language in that the vocabulary used in composing an ENGLISH sentence is contained in dictionaries. Verbs and file names are located in each user's Master Dictionary (M/DICT). User-files consist of a data section and a dictionary section. The dictionary section contains a structural definition of the data section. ENGLISH references the dictionary section for data attribute descriptions. These descriptions specify attribute fields, functional calculations, interfile retrieval operations, display format, and more.

ENGLISH offers these advantages:

- Limited freedom of word order and syntax for inqiries
- Generation of user-specified formatted output
- Sorting capability on variable number of descending or ascending sort keys
- Generation of statistical information concerning files
- Selection and sorting of items for use by subsequent TCL-II processors
- Relational and logical operations
- Support of 48-bit signed arithmetic (number range is -2^{47} through $2^{47} - 1$)

Is is assumed that the user has read the INTRODUCTION TO REALITY manual prior to referencing this publication.

As a general introduction to the ENGLISH language, Figure A illustrates a typical user inquiry (shaded text), as well as the formatted output produced by ENGLISH (nonshaded text).

```
:SORT THE ACCOUNT FILE WITH ITEMS BEFORE '23020' NAME ADDRESS <cr>


PAGE 1                                   14:43:38   12 FEB 1979

ACCOUNT...  NAME................  ADDRESS..........

11000       M H KEENER            100 ANCHOR PL
11015       L K HARMAN            118 ANCHOR PL
11020       J T O'BRIEN           124 ANCHOR PL
11025       P R BAGLEY            130 ANCHOR PL
11030       F E CARBON            101 BEGONIA
11035       R S MARCUS            107 BEGONIA
11040       E G MCCARTHY          113 BEGONIA
11045       F R DRESCH            119 BEGONIA
11050       J R MARSHECK          125 BEGONIA
11055       W H KOONS             131 BEGONIA
11060       F T NATORI            131 BAY STREET
11065       C V RANDALL           125 BAY STREET
11070       A A ALTHOFF           119 BAY STREET
11075       T F LINDSEY           113 BAY STREET
11080       E M AWAD              107 BAY STREET
11085       A B SEGUR             101 BAY STREET
11090       J W JENKINS           130 AVOCADO
11095       J B STEINER           124 AVOCADO
11100       E F CHALMERS          118 AVOCADO
11105       C C GREEN             112 AVOCADO
11110       D L WEISBROD          106 AVOCADO
11115       D R MASTERS           100 AVOCADO
21780       E W AWAD              107 BAY STREET
23000       H T LEE               200 BAY STREET
23005       W B THOMPSON          206 BAY STREET
23010       W E MCCOY             212 BAY STREET
23015       R M COOPER            218 BAY STREET

27 ITEMS LISTED.
```

Figure A. Sample ENGLISH Inquiry

2 ELEMENTS OF ENGLISH LAGUAGE

2.1 Forming ENGLISH Input Sentences

> The user forms ENGLISH input sentences which specify desired data retrieval functions. The ENGLISH retrieval language is a limited form of natural English. Formats for input sentences are simple yet very general. The ENGLISH processors, used with dictionaries, permit inputs to be stated directly in the technical terminology natural to each application area.

The ENGLISH language uses the lineal format natural to prose text. ENGLISH accepts any number of variable length words and permits a limited freedom of word order and syntax. The user constructs an ENGLISH input sentence terminated by a carriage return. This sentence then directs the appropriate ENGLISH processor to perform the specified data retrieval function. The ENGLISH input sentence contains several elements as shown in Figure A.

A verb and a file-name are required; all other elements are optional. Thus, the minimum ENGLISH sentence consists of a verb followed by a file-name. The item-list specifies items eligible for consideration (the absence of an item-list implies all items). An item-list consists of specifically enumerated item-ids, each enclosed within single quotes, additionally constrained by relational operators and logical connectives. Selection-criteria further limit items for output to those meeting the specified conditions. Output-specifications enumerate attributes (fields) desired for output. Figure B illustrates a sample ENGLISH input sentence.

The following general rules apply to the use of ENGLISH input sentences:

1. ENGLISH input sentences are entered at the TCL level, i.e., when the system prompts with a colon (:).

2. The first word of any ENGLISH input sentence must be an ENGLISH verb defined in the Master Dictionary (M/DICT).

3. A sentence is terminated by a <cr>. A sentence longer than 140 characters (roughly 1-3/4 lines on a Prism screen) may be continued to a second line by ending the first line with a segment mark (<c>_, X'FF') followed by a <cr>.

4. Exactly one file-name must appear in each sentence. File-names may consist of any sequence of nonblank characters and must be unique within the M/DICT and within all file dictionaries. The modifier "DICT" may be included anywhere in the sentence (normally just preceding the file-name) to specify operation on the file dictionary rather than the data file.

5. Any number of attribute names may be used in a sentence. Attribute names may consist of any sequence of nonblank characters and must be contained in the dictionary of the referenced file.

6. Any number of modifiers, connectives, and relational operators may be used which have been predefined in the M/DICT.

7. Verbs, file-names, attribute names, modifiers, connectives, and relational operators must be immediately followed by a blank or language delimiter (i.e., single quote, double quote, relational operator or carriage return).

8. Specified item-ids are enclosed within single quotes (e.g., 'XYZ') and may appear anywhere within the sentence.

9. Specified values are enclosed within double quotes (e.g., "ABC") and apply to the previous attribute name.

10. An option specification may appear in parenthesis.

A set of verbs, modifiers, connectives, and relational operators have been supplied. These special words are defined as items in the M/DICT and, to that extent, are reserved words. However, a user may define any number of synonyms for these words (and even remove the system defined entries) thereby creating his own semantics for the language. Synonyms may be created by copying the definition of the standard reserved word into an M/DICT item with the desired synonym name as the item-id (refer to the Reality Programmer's Reference Manual).

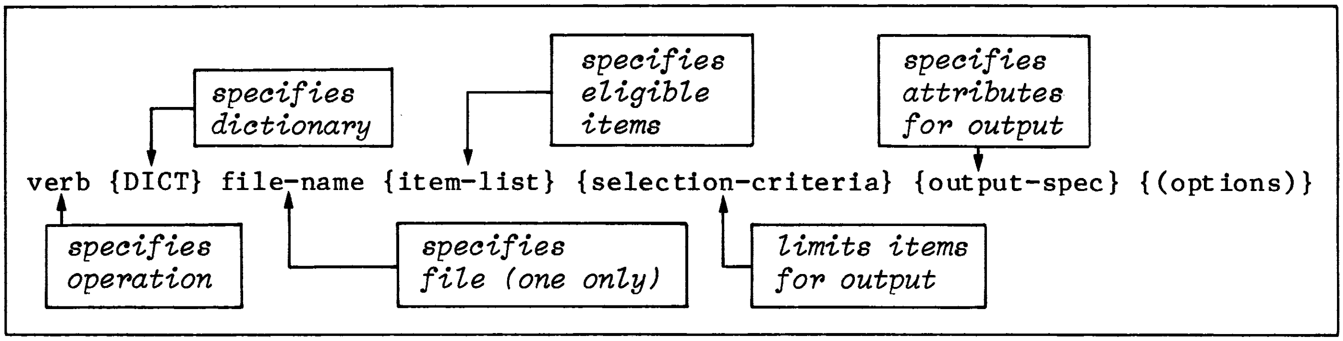

specifies dictionary
specifies eligible items
specifies attributes for output
verb {DICT} file-name {item-list} {selection-criteria} {output-spec} {(options)}
specifies operation
specifies file (one only)
limits items for output

Figure A. General Form of ENGLISH Input Sentence

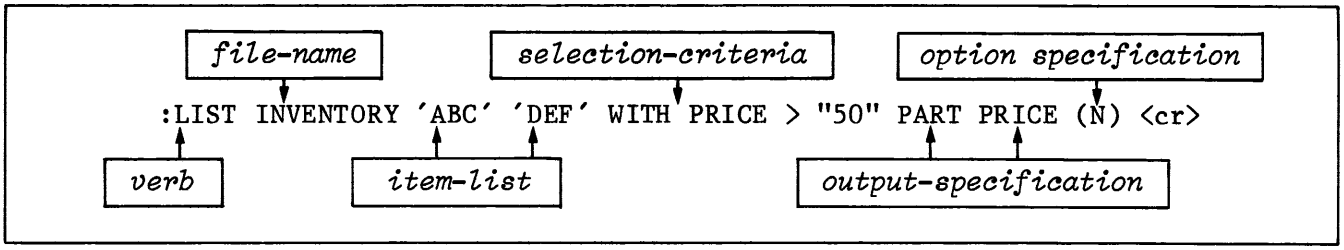


Figure B. Sample ENGLISH Input Sentence

2.2 Overview of ENGLISH Verbs

Each ENGLISH sentence must begin with one (and only one) ENGLISH verb. A set of ENGLISH verbs is provided.

ENGLISH verbs are action oriented words which invoke specific ENGLISH processors. Common ENGLISH verbs are listed in Figure A and are briefly discussed below. A separate chapter in this manual presents a complete description of these ENGLISH verbs. Figure B illustrates sample usage of the verbs.

LIST and SORT; LIST-LABEL and SORT-LABEL

The LIST and SORT verbs are used to generate formatted output. LIST simply lists the selected output, while SORT orders the output in some specified sorted order, either ascending or descending. It can also perform exploding sorts on multivalued attributes. A sorted order will also be invoked without the SORT verb if a BY clause is included. Generated output is formatted into a columnar output if possible. LIST-LABEL and SORT-LABEL are analogous to LIST and SORT but allow formatting for printing labels and other formatted output.

COUNT

The COUNT verb counts the number of items meeting the conditions specified by the combination of item-list and selection-criteria. The output generated by this verb is simply the number of items counted.

SUM and STAT

The SUM and STAT verbs provide a facility for summing one specified attribute. The output generated by these verbs is the derived statistics.

SELECT and SSELECT

The SELECT verb provides a facility to select a set of items using the item-list and selection-criteria. These selected items are then available one at a time to TCL-II processors. The SSELECT verb combines the SORT capability with the SELECT capability.

SAVE-LIST, GET-LIST, EDIT-LIST, COPY-LIST, and DELETE-LIST

The SAVE-LIST, GET-LIST, EDIT-LIST, COPY-LIST, and DELETE-LIST verbs are used to save, retrieve, edit, copy and delete item-lists created by SELECT and SSELECT statements.

FORM-LIST

The FORM-LIST verb forms an item-list from a set of item-ids stored in a file item that was created by the user.

T-DUMP, I-DUMP, and T-LOAD

The T-DUMP and I-DUMP verbs allow the user to selectively dump his dictionaries and data files to magnetic tape or the terminal, respectively. The T-LOAD verb allows a user to selectively load his dictionaries and data files from magnetic tape.

ISTAT

The ISTAT verb provides a file hashing histogram.

COUNT
DELETE-LIST
EDIT-LIST
FORM-LIST
GET-LIST
I-DUMP
ISTAT
LIST
LIST-LABEL
SAVE-LIST
SELECT
SORT
SORT-LABEL
SSELECT
STAT
SUM
T-DUMP
T-LOAD

Figure A. ENGLISH Verbs

This figure illustrates sample ENGLISH input sentences. Any dialogue and generated output are not shown.

```
:LIST ACCOUNT NAME CURR-BALNC WITH CURR-BALNC <cr>

:SORT ACCOUNT > '10000' WITH CURR-BALNC <cr>

:LIST-LABEL ACCOUNT NAME ADDRESS <cr>

:SORT-LABEL ACCOUNT NAME ADDRESS BY BILL RATE <cr>

:COUNT INV WITH PRICE <".30" <cr>

:SUM FILE4 QUAN <cr>

:SELECT DICT MD WITH D/CODE "D" <cr>

:SSELECT ACCOUNT WITH BILL-RATE = "10.03" <cr>

:SAVE-LIST TEMP <cr>

:GET-LIST TEMP <cr>

:EDIT-LIST TEMP <cr>

:FORM-LIST DICT INV BACK-ORDERED <cr>

:T-DUMP XYZ WITH VALUE1 <cr>

:T-LOAD XYZ < '505' <cr>
```

Figure B. Sample ENGLISH Input Sentences

2.3 Using Relational Operators and Logical Connectives

Relational operators and logical connectives may be used to form complex item-lists and selection-criteria.

The relational operators are listed in Figure A. Relational operators may be used in an item-list to constrain items eligible for processing (refer to the topic FORMING ITEM-LISTS), or may be used in selection-criteria to limit items to those whose attributes meet the specified conditions (refer to the topic FORMING SELECTION-CRITERIA). Relational operators apply to the item-id or value immediately following the operator. The absence of a relational operator implies an equality operator.

To resolve a relational condition, every item-id (or attribute value) is compared to the item-id (or value) specified in the item-list (or selection-criteria) of the ENGLISH input sentence. Character pairs (one from the specified item-id or value and one from the item-id or attribute currently being compared) are compared one at a time from leftmost characters to rightmost. If no unequal character pairs are found, then the item-ids or values are considered to be "equal". If an unequal pair of characters is found, the characters are ranked according to their numeric ASCII code equivalents (refer to the list of ASCII codes in the Appendix in this manual). The item-id or value contributing the higher numeric ASCII code equivalent is considered to be "greater" than the other. (If attributes are right-justified, a numeric comparison is attempted first. If either or both of the item-ids (values) are nonnumeric, the item-id (value) with more characters is considered "greater". If both item-ids (values) are of equal length, character pair comparison, as for left-justified attributes, is used.)

Logical connectives are listed in Figure B. Logical connectives bind together sets of item-ids into item-lists, sets of values into value-lists, and sets of selection-criteria into selection-criterion lists. The AND connective specifies that both connected parts must be true, while the OR connective specifies that either (or both) connected parts must be true. In all cases where neither AND nor OR are specified, OR will be assumed.

An ASCII up-arrow (^) may be used as an ignore character in any left-justified value. All comparisons made against the value then ignore the characters in the corresponding positions. The ignore character may be used to compare against item-ids if it is used with an attribute that is synonymous with the item-id (i.e., one whose A/AMC is zero).

Figure C exemplifies the use of relational operators and logical connectives. The user should note that these are partial examples and therefore do not illustrate complete ENGLISH sentences. They are presented at this point to give the user a general feel for these operators. Complete ENGLISH sentences using the above constructions are presented throughout the remainder of the manual.

Note that the precedence of different operators is different for selection criteria than for item-list criteria.

Symbol	Operation
= or EQ	*equal to*
> or GT or AFTER	*greater than*
< or LT or BEFORE	*less than*
>= or GE	*greater than or equal to*
# or NE or NOT or NO	*not equal to or null attribute value*
	If a relational operator is not given, EQ is assumed.

Figure A. Relational Operators

Symbol	Operation
AND	*Both connected parts must be true.*
OR	*Either connected part must be true.*
	If a logical connective is not given, OR is assumed.

Figure B. Logical Connectives

Example	Explanation
= 'ABC' OR > 'DEF'	*Item-list which selects item 'ABC' as well as all items with item-ID's greater than 'DEF'.*
WITH A > "5" AND < "9"	*Selection-criterion which selects all items having a value for attribute A which is between 5 and 9 (exclusive).*
WITH A1 ="X" AND WITH A2 ="^Z"	*Selection-criteria which selects all items having a value of "X" for attribute A1, and a value for A2 which consists of any character followed by a "Z".*
LT '100' GT '200'	*Item-list which selects all items with item-ID's either less than '100" or greater than '200'.*
WITH NO CURR-BLANC	*Selection-criteria which selects items having a null value for attribute CURR-BALNC.*

Figure C. Sample Usage of Relational Operators and Logical Connectives

2.4 Forming Item-Lists

An item-list specifies items eligible for consideration by the specified operation, and consists of specifically enumerated item-ids optionally constrained by relational operators and logical connectives.

An item-list defines items desired for processing. Absence of an item-list implies all items on the file. A simple item-list consists of any number of specified item-ids surrounded by single quotes (e.g., 'XYZ'). These item-ids may be interspersed at will throughout the ENGLISH input sentence. The general form of a simple item-list is shown in Figure A.

Complex item-lists may be constructed using relational operators and logical connectives. For example, consider the following item-list:

```
'ABC' OR >= 'DEF' AND < 'GHI'
```

This item-list selects item 'ABC' as well as all items with item-ids both greater than or equal to 'DEF' and also less than 'GHI'. The general form of a complex item-list is shown in Figure B.

Use of the complex item-list causes all items in the file to be accessed for examination as does absence of an item-list. If a simple item-list is used, only those items will be accessed, and processng will be faster.

The hierarchy (precedence) of the logical connectives in an item-list is left to right. For example, consider this item-list:

```
< 'A' OR > 'B' AND < 'C' OR > 'D' AND < 'E'
```

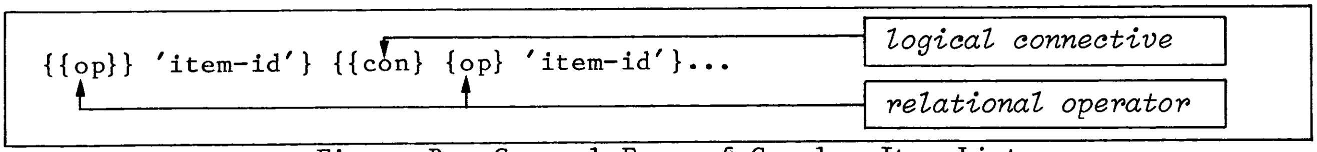
This item-list selects all items wth item-ids less than 'A', or with item-ids greater then 'B' but less than 'C', or with item-ids greater than 'D' but less than 'E'. Since the OR connective is always implied (and may therefore be omitted), the above item-list may have been equivalently specified:

```
< 'A' > 'B' AND < 'C' > 'D' AND < 'E'
```

Further examples of item-lists are illustrated in Figure C. Here the SORT verb is used to select and sequence the item-ids in file TEST. (TEST contains 10 items, with item-ids '10' through '19'). The word ONLY used in these examples specifies that only the item-ids are to be listed.

```
'item-id' {'item-id'}...
```

Figure A. General Form of Simple Item-List



```
{{op}} 'item-id'} {{con} {op} 'item-id'}...
```

Figure B. General Form of Complex Item-List

```
:SORT ONLY TEST > '13' and < '17' <cr>

PAGE 1                                                15:32:19  12 FEB 1979

TEST......

14
15
16

3 ITEMS LISTED.

:SORT ONLY TEST >= '13' AND <= '16' OR >='18' AND < '19' <cr>

PAGE 1                                                15:33:01  12 FEB 1979

TEST......

13
14
15
16
18

5 ITEMS LISTED.

:SORT ONLY TEST NOT '13' AND NOT '15' AND NOT '17' AND NOT '19' <cr>

PAGE 1                                                15:33:31  12 FEB 1979

TEST......

10
11
12
14
16
18

6 ITEMS LISTED.

:SORT ONLY TEST BEFORE '13' <cr>

PAGE 1                                                15:34:24  12 FEB 1979

TEST......

10
11
12

3 ITEMS LISTED.
```

Figure C. Sample Usage of Item-List

2.5 Forming Selection-Criteria

Selection-criteria specify a set of conditions which must be met by an item before it is eligible for output. Selection-criteria are made up of one criterion or several.

The general form of a selection is shown in Figure A. Each selection-criterion must begin with the word WITH or IF followed by a single attribute name. (WITH and IF are synonymous). The attribute name may then be followed by a value-list. Rules for forming value-lists are identical to those for forming item-lists (refer to the topic FORMING ITEM-LISTS), except that double quotes must surround the actual values. For example, the following selection-criterion is met by items which have at least one value for the attribute DESC which is either equal to "ABC" or is both greater than "DEF" and less than "GHI":

```
WITH DESC "ABC" OR > "DEF" AND < "GHI"
```

If a selection-criterion does not include a value-list, then it is true for all items which have at least one value for the specified attribute name. The selection-criterion may be further modified by using the modifier EVERY immediately following the WITH. The modifier EVERY requires that every value for the attribute meet the specified condition, i.e., if the attribute has multivalues, then each value must meet the condition. (The modifier EACH is a synonym for EVERY). The modifier NO may immediately follow the WITH and test for the lack of an attribute value (or null value). In this case a value is not included (e.g., with NO ZIP.CODE).

Several selection-criteria may be bound together by logical connectives to form the complete selection-criteria. When used in this fashion, the AND connective has a higher precedence than the OR connective. A selection-criterion may consist of up to nine "AND clauses". An AND clause is made up of any number of selection-criteria bound by AND connectives. The AND clause is terminated when an OR connective is found in the left to right scan. (Note: the absense of an AND connective implies an OR connective.) For an item to pass the selection-criteria, the conditions speciified by any one of the AND clauses must be met. An example of the logical hierarchy of AND clauses is shown in the selection-criteria below (the parentheses have been included for clarity but do not appear in the actual ENGLISH sentence):

```
(WITH DESC  "ABC" AND WITH  VALUE "1000") OR  (WITH DESC "ABC" AND WITH  NO
VALUE)
```

It is important to note that every attribute name must be preceded by the word WITH (or IF) when combining multiple attributes with the AND or OR connectives. For example, correct usage might be:

```
WITH QTY < "10" OR > "1000" OR WITH PRICE < "10.50"
```

whereas

```
WITH QTY < "10" OR > "1000" OR PRICE < "10.50"
```

would not produce the desired results. Since the ORs are optional (and discarded) the phrase PRICE < "10.50" would be intrepreted as a multivalue print limiter (refer to the topic OUTPUT CRITERIA: MULTIVALVE PRINT LIMITING).

Figure B illustrates further examples of selection-criteria.

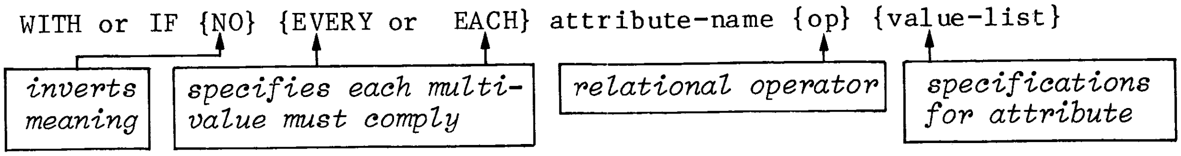

WITH or IF {NO} {EVERY or EACH} attribute-name {op} {value-list}
inverts meaning
specifies each multi-value must comply
relational operator
specifications for attribute

Figure A. General Form of ENGLISH Selection-Criteria

```
:LIST ACCOUNT NAME WITH AVG-USAGE "20" OR "25" AND <c>_<cr>
:WITH SEWER-ASMT "1.50" OR WITH AVG-USAGE "20" OR "30" <c>_<cr>
:AND WITH BILL-RATE > ".30" AVG-USAGE SEWER-ASMT BILL-RATE <cr>

PAGE 1                                             17:36:04  12 FEB 1979

ACCOUNT... NAME............... AVG-USAGE SEWER-ASMT... BILL-..
                                                        RATE

23100      G J PACE                  30                 10.30
35035      M J LANZENDORPHER         30                  0.35
23080      J W YOUNG                 20          1.50    8.40
11045      F R DRESCH                30                 10.03

4 ITEMS LISTED.

:COUNT ACCOUNT WITH CURR-BALNC > "100" AND WITH SEWER-ASMT  <c>_<cr>
:OR WITH BILL-RATE = "30" <cr>

7 ITEMS COUNTED.

:LIST ACCOUNT TRNS-DATE WITH EVERY TRNS-DATE BEFORE "3/18/78" <cr>

PAGE 1                                             17:40:57  12 FEB 1979

ACCOUNT... TRNS-DATE...

11075      17 MAR 1978
           17 MAR 1978
           17 MAR 1978
           13 MAR 1978
           15 JAN 1978
           14 JAN 1978
           10 JAN 1978

END OF LIST
```

Figure B. Sample Usage of Selection-Criteria

2.6 Selection-Criteria: String Searching and Item Size

Selection-criteria may additionally be used to search an attribute for a string of characters, and to use the size of an item as a criterion.

String Searching

ENGLISH has the ability to search an attribute value for any string of characters. The left bracket ([) and the right bracket (]) may be used within double quotes in a selection-criteria. A left bracket indicates that there may be any (or no) characters to the left of the string. A right bracket indicates that there may be any (or no) characters to the right of the string. Used separately, the left bracket specifies that the value must end with the character string, while a right bracket specifies that the value must begin with the character string. If both brackets are used, the character string may appear anywhere in the attribute value. Figure A illustrates the use of this feature.

Note: String searching does not function on item-ids unless an operator precedes the item-list values (i.e. the equality operator is not assumed in this case). If an operator is not specified, ENGLISH will look for item-ids containing bracket(s).

Item Size

The size of items may be used as a selection-criterion. This will cause the size of the item (as specified in the count-field of the item) to be retrieved. Thus the user may LIST or SORT items conditionally on their size. To use this feature, the user must create an attribute definition item in the dictionary of the file with an A/AMC of 9999. For example:

```
    SIZE
001 A
002 9999
003
004
005
006
007 MDO,
008
009 R
010 6
```

SIZE could then be used as a selection-criterion as shown in Figure B.

```
:LIST ACCOUNT WITH NAME "[MAN" NAME <cr>

PAGE 1                                                18:13:27  12 FEB 1979

ACCOUNT... NAME................

23025      D C BINGAMAN
23055      S M NEWMAN
11015      L K HARMAN

3 ITEMS LISTED.
:LIST ACCOUNT WITH NAME "A A A]" NAME <cr>

PAGE 1                                                18:14:09  12 FEB 1979

ACCOUNT... NAME................

11070      A A ALTHOFF

END OF LIST
:LIST ACCOUNT WITH NAME "[INE]" NAME <cr>

PAGE 1                                                18:16:56  12 FEB 1979

ACCOUNT... NAME................

11095      J B STEINER
35065      L J RUFFINE

2 ITEMS LISTED.
```

Figure A. Sample Usage of String Searching Selection-Criteria

```
:SORT ACCOUNT SIZE WITH SIZE > "300" <cr>

PAGE 1                                                18:20:52  12 FEB 1979

ACCOUNT... SIZE..

23060          596
23075          317
23080          318
35085          404

4 ITEMS LISTED.
```

Figure B. Sample Usage of SIZE Selection-Criteria

2.7 Forming Output-Specifications: Columnar vs. Non-Columnar Output

Output-specifications enumerate attributes to be listed.

All attribute names in an ENGLISH sentence which are not part of a selection-criterion (i.e., those not preceded by the modifiers WITH or IF or not modified by certain control modifiers*) are considered as part of the output-specification. These attribute names specify the attribute values which are to be printed as a result of the specified operation. However, only those attribute values from items which pass both the item-list and the selection-criteria will be output. For example:

```
    LIST INV > '500' SIZE QUAN
```

This ENGLISH sentence causes attribute values for attributes SIZE and QUAN in all items with item-ids greater than 500 (in file INV) to be listed.

Selected attributes will be displayed in an automatically generated system format. This format will include a heading line displaying the date, time and page number (unless suppressed*) at the beginning of each new page. The page size is set through the use of the TERM command (refer to the Reality Programmer's Reference Manual). The LIST and SORT verbs will attempt to format the output into a columnar format with each specified attribute name as a column heading (using as a column width either the attribute max-size from the dictionary, the attribute name, or the S/NAME heading, whichever is larger). If the sum of the column widths (adding one blank separator for each specified attribute name) does not exceed the page width as set by the TERM command, then a columnar format will be generated. In a columnar format, the specified attribute names (or S/NAME fields) are displayed as column headings across the top of the page. The values for each of the items are then displayed in their respective columns. The column headings are repeated at the top of each new page.

If the requested output exceeds the page width, then the attribute names are listed down the side of the output with their respective values immediately to the right. A significant difference between the two formats is that for the columnar format all headings are listed only once for each page, whether or not values exist for the columns; while in the non-columnar format, headings are displayed for each item only if there are values for the associated attributes.

The general form of the output-specification is shown in Figure A. Examples of the output-specification are illustrated in Figure B and C. Figure B shows a columnar output format, while Figure C shows a noncolumnar output format.

*Refer to the topic USING MODIFIERS

```
attribute-name {attribute-name}...
```

Figure A. General Form of Output-Specification

```
:SORT ACCOUNT WITH CURR-BALNC > "100000" NAME ADDRESS CURR-BALNC <cr>

PAGE 1                                                  09:09:19  12 FEB 1979

ACCOUNT... NAME................ ADDRESS............. CURR-BALANCE...

11020      J T O'BRIEN          124 ANCHOR PL        $    306,755.54
11055      W H KOONS            131 BEGONIA          $    958,343.75
23040      P B SCIPMA           213 CARNATION        $    123,423.22
35080      G A BUCKLES          307 DOCK WAY         $    447,765.48

4 ITEMS LISTED.
```

Figure B. Columnar Output Format

```
:LIST ACCOUNT '35060' NAME ADDRESS CURR-BALNC BILL-RATE AVG-USAGE <cr>

PAGE                                                    09:11:53  12 FEB 1979

ACCOUNT:  35060
NAME    J A SCHWARTA
ADDRESS    331 DOCK WAY
CURR-BALNC     $ 33,822.34
BILL-RATE  0.02
AVG-USAGE     31

END OF LIST
```

Figure C. Noncolumnar Output Format

2.8 Output-Criteria: Multivalue Print Limiting

Selection-criteria may also be used to limit printing to specific values from multivalued and sub-multivalued attributes.

Limiting output to specific values of multivalued and sub-multivalued attributes can be accomplished by following the attribute name with a print limiting clause using relational and logical operators and values enclosed in double quotes. See Figure A for the general form. If the attribute is an associative attribute (D1), then the corresponding values from the D2 attributes (if specified) will also be returned. However, all items will be listed unless a selection-criterion (WITH clause) is also used to select only items with the desired value(s). For further information regarding associative attributes, refer to the topic DEFINING ASSOCIATIVE ATTRIBUTES: D1 AND D2.

The example in Figure B lists all the items in the INV file. In the example in Figure C, the TRAN-DATE < "12 FEB 78" portion of the ENGLISH sentence indicates to the ENGLISH processor that detail will be listed only when the date is less than "12 FEB 78". Note that there is no WITH preceding the attribute name TRAN-DATE, which invokes print limiting. Also note that all three items were listed because there was no selection-criterion.

If print limiting on both multivalues and sub-multivalues at the same time, then the positional relationship will be maintained by "blanking out" values that do not match the print limiting clause.

The TOTAL modifier may be used to total print limited fields. Figure D illustrates the use of both selection-criteria and print limiting on both multivalued and sub-multivalued fields along with the TOTAL modifier.

```
attribute-name {op} "value" {AND/OR {op} "value"}...
```

Figure A. General Form of Print Limiting Clause

```
:LIST INV TRAN-DATE TRAN-TYPE TRAN-QTY <cr>

PAGE 1                                        11:39:47  12 FEB 1979

INV....... TRAN-DATE TRAN-TYPE TRAN-QTY
                     *         *
1242-22    11 FEB    I              100
                     R               48
                     X               31
           12 FEB    I              144
                     S               43
1242-11    11 FEB    I               19
                     X              122
           13 FEB    R               97
1242-33    16 FEB    I               11
                     C              122
           17 FEB    C               68
                     R               71
```

Figure B. Display of the INV File

```
:LIST INV TRAN-DATE BEFORE "12 FEB 78" TRAN-TYPE TRAN-QTY <cr>

PAGE 1                                              11:41:17  12 FEB 1979

INV....... TRAN-DATE TRAN-TYPE TRAN-QTY
                         *         *
1242-22    11 FEB        I              100
                         R               48
                         X               31
1242-11    11 FEB        I               19
                         X              122
1242-33

3 ITEMS LISTED.
```

Figure C. Sample Usage of Print Limiting

```
:LIST INV WITH TRAN-DATE BEFORE "12 FEB 78" <c> <cr>
:TRAN-DATE < "12 FEB 78" TRAN-TYPE TOTAL TRAN-QTY <= "50" <cr>

PAGE 1                                              11:42:52  12 FEB 1979

INV....... TRAN-DATE TRAN-TYPE TRAN-QTY
                         *         *
1242-22    11 FEB        I
                         R               48
                         X               31
1242-11    11 FEB        I               19
                         X

***                                      98

2 ITEMS LISTED.
```

Figure D. Sample Usage of Selection-Criteria and Print Limiting Totals

2.9 Omission of Output-Specification

If no output-specifications appear, attributes defined by default attribute definition items are selected. This special feature is outlined below; however, for a complete description of attribute definition items and their use, refer to the Reality Programmer's Reference Manual and the SCREENPRO Programming Manual.

If all output-specifications are omitted, then default attributes defined in the dictionary via attribute definition items (i.e., with D/CODEs of A, S or X) will be assumed as the output specification. Default attribute definition items are those with item-ids which are numeric and sequential (i.e., 1, 2, 3, 4,...). Attributes with D/CODEs of A or S are listed; attributes with D/CODEs of X are not listed (i.e., they are only used to maintain the required sequential order). Attribute definition items have a special format (see Figure A).

Item-ids are always included in the output listing unless the modifier ID-SUPP is used. For an output listing only the item-ids, the modifier ONLY must precede the file-name to inhibit the listing of default attributes defined by attribute definition items (item-ids 1,2,3...etc.).

Figure A summarizes the various dictionary attributes as they apply to the formatting of output produced by an ENGLISH operation. For further details regarding attribute definition items, refer to the Reality Programmer's Reference Manual. Figure B shows a sample statement with the output-specifications omitted.

Name	A/AMC	Value	Meaning
D/CODE	1	A or S X	Attribute definition item. Special code to maintain order (but defined attribute is not output by ENGLISH.)
A/AMC	2	attr-num	Defines attribute number.
S/NAME	3	text-name	For A-code and S-code attributes; defines attribute heading to be output by ENGLISH. These names may be padded with blanks to align noncolumnar output. Multiple line column headings may be specified by separating strings with a value mark (<c>]).
S/AMC	4		Not used--reserved.
V/TYP	9	L	For columnar output only; specifies left justification. If value size is greater than column width, value is folded.

Figure A. Attribute Definition Item Summary

Name	A/AMC	Value	Meaning
(cont.) V/TYP	9	R	For columnar output only; specifies right justification. If value size is greater than column width, value overlays previous columns.
		T	For columnar output only; specifies left justification of textual data. If value size is greater than column width, string will be "folded" at blanks.
		U	For columnar output only; specifies left justification. If value size is greater than column width, entire value is printed on the line ignoring column boundaries. Overprinting may occur in other columns.
V/MAX	10	n	For columnar output only; specifies number of characters to reserve for the column width. Column width will be increased if attribute name or text-name heading is larger than V/MAX.
	11-20		Reserved for use by Screen Processor.

Figure A. Attribute Definition Item Summary (Continued)

```
:LIST ACCOUNT '35095' <cr>

PAGE 1                                              18:24:04  12 FEB 1979

ACCOUNT  :  35095
NEXT-ACCT      35100
CSTMR-NAME     A W FEVERSTEIN
SERVC-ADDR     324 CARNATION
MAIL-ADDR      19401 DORAL
MAIL-CITY.     ANOTHER CITY
MAIL-STATE     CA
ZIP-CODE..     19252
DEPOSIT-=      10.00
START-DATE     01 JAN 1968
BILL-RATE.     0.35
AVG-USAGE.     32
CURR-BALNC     19.25
60-DAYS..      9.80

END OF LIST
```

Figure B. Sample Omission of the Output-Specification

2.10 Using Modifiers and Options

Modifiers and the option specification may be used to further modify the meaning of ENGLISH sentences.

Modifiers which may be used in an ENGLISH sentence are listed in alphabetical order below.

Modifier — Description

BREAK-ON
Defines control-breaks (see the topic BREAKING ON ATTRIBUTE VALUES.)

BY
Designates the attribute name immediately following as a sort key for the SORT operation. Sequencing is in ascending order comparing ASCII values (see the topic THE SORT VERB).

BY-DSND
Like BY, except sort is in descending order.

BY-EXP
Designates the attribute name immediately following as an exploding sort key (on multivalues) for the SORT operation. Sequencing is in ascending order comparing ASCII values (see the topics EXPLODING SORT: MULTIVALUED ATTRIBUTES, and THE SORT VERB).

BY-EXP-DSND
Like BY-EXP, except sort is in descending order.

COL-HDR-SUPP
Suppress the output of the page number and time/date heading, the column headings, and the "XX ITEMS LISTED" message.

DBL-SPC
Causes output to be double-spaced.

DET-SUPP
Suppresses detail output when used with TOTAL or BREAK-ON modifiers (see the topic GENERATING SUBTOTALS USING CONTROL-BREAKS).

DICT
Modifies the file-name so that the ENGLISH sentence references the file dictionary instead of the file (see the topic FORMING ENGLISH INPUT SENTENCES).

EVERY or EACH
Modifies a selection-criterion so that every value for a multivalued attribute must meet the specified condition for the criterion to be true. This modifier must immediately follow the modifier WITH (see the topic FORMING SELECTION-CRITERIA).

GRAND-TOTAL
Specifies a label for the grand-total line.

HDR-SUPP or SUPP
Suppresses the output of the page number and time/date heading, and the "XX ITEMS LISTED" message.

Modifier Description

ID-SUPP Suppresses the display of item-ids for LIST and SORT operations.

LPTR Routes output to the printer.

NOPAGE When output is to the terminal, this modifier will suppress the automatic paging of outputs; i.e., pages will be output to the terminal one after the other without pausing for the user to enter a carriage return.

ONLY Inhibits the appending of the special default synonym attributes when a null output-specification is encountered (see the topic OMISSION OF THE OUTPUT-SPECIFICATION); when used, must precede the file-name.

PAGE (Optional) - When output is to the terminal, the PAGE mode halts output at the end of each page; output of the next page resumes when the user enters a carraige return. PAGE mode is automatically in effect unless the NOPAGE modifier or 'N' option is in effect.

TAPE Indicates that retrieval is from the tape file positioned on the tape drive rather than from a disc file. Attribute definitions will be found in the dictionary of the file specified in the sentence. If a dictionary file is specified, then attribute definitions will be retrieved from the user's M/DICT. The TAPE modifier is only valid with LIST, SELECT, COUNT, SUM, STAT, I-STAT, and LIST-LABEL verbs.

TOTAL Causes totals to be accumulated for the attribute which follows (see the topic GENERATING TOTALS).

WITH or IF Designates selection-criteria (see the topic FORMING SELECTION-CRITERIA).

WITHIN Specifies that the file-name immediately following is a sublist file (see the topic SUBLISTS: THE WITHIN CONNECTIVE).

The next topic, USING THROWAWAY CONNECTIVES, contains examples illustrating the use of modifiers.

An option specification may be included in ENGLISH input sentences to modify the meaning. Options consist of single letters separated by commas. The option specification is surrounded by parentheses. For example, the option specification (N) eliminates paging (NOPAGE).

Different verbs permit different options, but, in general, the following options apply to most ENGLISH sentences:

I	List item-ids numbered sequentially
N	NOPAGE
P	Route output to the spooler for printing

2.11 Using Throwaway Connectives

Throwaway connectives do not affect the meaning of ENGLISH sentences. They may be used anywhere in the sentence and are included to provide a degree of naturalness to the language.

Throwaway connectives which may be used in an ENGLISH sentence are listed in alphabetical order below.

Throwaway Connective	Description
A	Adjective, e.g., WITH A PRICE GT "500"
AN	Adjective, e.g., WITH AN AMOUNT LT "1"
ARE	Connector, e.g., ITEMS ARE GT "40"
ANY	Adjective, e.g., LIST ANY NAME
FILE	Noun, e.g., LIST THE INV FILE
FOR	Connector, e.g., FOR ITEMS > '35000'
IN	Connector, e.g., LIST ITEMS IN ACCOUNT FILE
ITEMS	Noun, e.g., ITEMS NE "40"
OF	Connector, e.g., NAMES OF DELEGATES
OR	Logical connector, e.g., COUNT EQ "10" OR LT "20"
THE	Adjective, e.g., LIST THE NAME

The user may create his own throwaway connectives by copying any of these items into the desired item in the M/DICT.

Figure A illustrates the use of modifiers and throwaway connectives.

```
:LIST THE ACCOUNT FILE SUPP DBL-SPC <cr>
```

This sentence causes the ACCOUNT file to be listed. Listing will be double-spaced, and the page number and time/date heading and "XX ITEMS LISTED" messages will be suppressed.

```
:SORT INVENTORY WITH PRICE GT "500" SUPP (P) <cr>
```

This sentence causes items in the INVENTORY file with PRICE greater than 500 to be sorted and listed. The output will be to the printer; the time/date heading and the end-of-list message will not be printed.

```
:LIST ACCOUNT ITEMS > '35000' NAME ADDRESS ID-SUPP <cr>
```

This sentence causes the values for NAME and ADDRESS (in items with item-ID's greater than '35000' to be listed (item-ID's will not be listed.

```
:LIST INVENTORY WITH QTY < "10" INV.DATE DATE.ORDERED TAPE <cr>
```

This sentence causes the tape file to be listed using the attribute definitions QTY, INV.DATE, and DATE ORDERED found in the Dictionary of the INVENTORY file. Only items with QTY less than "10" will be listed.

```
:LIST DICT ACCOUNT TAPE D/CODE A/AMC S/NAME V/TYP V/MIN LPTR <cr>
```

This sentence causes the tape file to be listed on the printer using the attribute definitions D/CODE, A/AMC/ S/NAME, V/TYP, and V/MIN found in the user's M/DICT (since a DICT level file was specified).

Figure A. Sample Usage of Modifiers and Throwaway Connectives

2.12 Generating Headings and Footings

LIST and SORT statements may optionally specify headings and footings. A heading is any title appearing at the top of the page. A footing is any title appearing at the bottom of the page.

HEADING

A user-generated heading can be specified in a LIST or SORT statement. The specified heading will be printed at the top of every page of output. The normal page number, time and date heading, and end-of-list message will not be printed when a user-generated heading is specified.

A HEADING specification may apper anywhere in the LIST or SORT statement. To specify a heading, the user enters the word HEADING followed by a string of characters enclosed in double quotes (" "). Special option characters may appear, enclosed in single quotes (′ ′). This gives the HEADING specification the following general form:

```
HEADING "{text} {′options′}..."
```

For example:

```
HEADING "INVENTORY LIST"
```

This prints the title "INVENTORY LIST" at the top of each page. Options are described in the topic HEADING AND FOOTING OPTIONS.

FOOTING

A user-generated footing can be specified in a LIST or SORT statement. The specified footing will be printed at the bottom of every page of output. FOOTING has the same general form as HEADING, i.e.:

```
FOOTING "{text} {′options′}..."
```

The FOOTING specification operates the same as described above for HEADING except that the normal page number time and date heading are not suppressed. The HDR-SUPP modifier may be used in sentences that specify FOOTINGS with dates or page numbers so that this information is not repeated at the top.

Figure B illustrates sample usage of HEADING and FOOTING specifications. Special option characters used in the examples are described in the topic HEADING AND FOOTING OPTIONS.

```
HEADING "{text} {'options'}..."

                  See HEADING AND FOOTING OPTIONS
                  for option characters

FOOTING "{text} {'options'}..."
```

Figure A. General Form of HEADING and FOOTING Specifications

```
:SORT ACCOUNT NAME HEADING "NAME LIST AT 'TL' PAGE NO. 'PL'" <cr>

NAME LIST AT 10:29:39  12 FEB 1979
PAGE NO. 1

ACCOUNT... NAME................

11000      M H KEENER
11015      L K HARMAN
11020      J T O'BRIEN
11025      P R BAGLEY
11030      F E CABRON
   .
   .
   .


:LIST INVENTORY FOOTING "'LL' INVENTORY REPORT FOR 'DL'PAGE 'P'" <c>_<cr>
:QTY DATE LOCATION HDR-SUPP <cr>

INVENTORY. QTY.. DATE........ LOCATION
                              ROW..BIN


1107         432 11/03/76       3    5
1011          17 11/05/76       2   11
1012           3 12/16/76      11    3
1003         115 01/09/77       9    6
   .         . .                .    .
   .         . .                .    .
   .         . .                .    .

INVENTORY REPORT FOR 12 FEB 1979
PAGE 1
```

Figure B. Sample Usage of HEADING and FOOTING Specifications

2.13 Heading and Footing Options

HEADING and FOOTING specifications allow special option characters to be replaced by the current time, date, page number, etc. Expanded print titles can be generated on some line printers.

Special option characters allow HEADING and FOOTING specifications to include date, time, page number, and file-name, and to perform formatting such as starting a new line. Options are specified by including any of the characters listed in Figure A, enclosed in single quotes.

Examples:

```
HEADING "STATUS REPORT 'L' PAGE:  'P'"

FOOTING "INVENTORY REPORT FOR 'D'           PAGE 'P'"
```

In the first example a heading has been specified which consists of the words "STATUS REPORT", followed by a carriage return and a line feed (L option), followed by the word "PAGE:," followed by the current page number (P option). In the second example a footing is specified that will consist of the words "INVENTORY REPORT FOR," followed by the system date (D option), followed by a number of spaces and the word PAGE, followed by the current page number (P option).

The special option characters to be enclosed in single quotes are listed in Figure A. To actually print a single quote mark within the text, a sequence of two single quotes ('') may be used.

Expanded Print

An expanded print capability is available on the Microdata Matrix Printer and some other matrix printers. Headings and footings may be printed in expanded print by preceding the text string with the ASCII "SO" character (<c>N, X'0E')

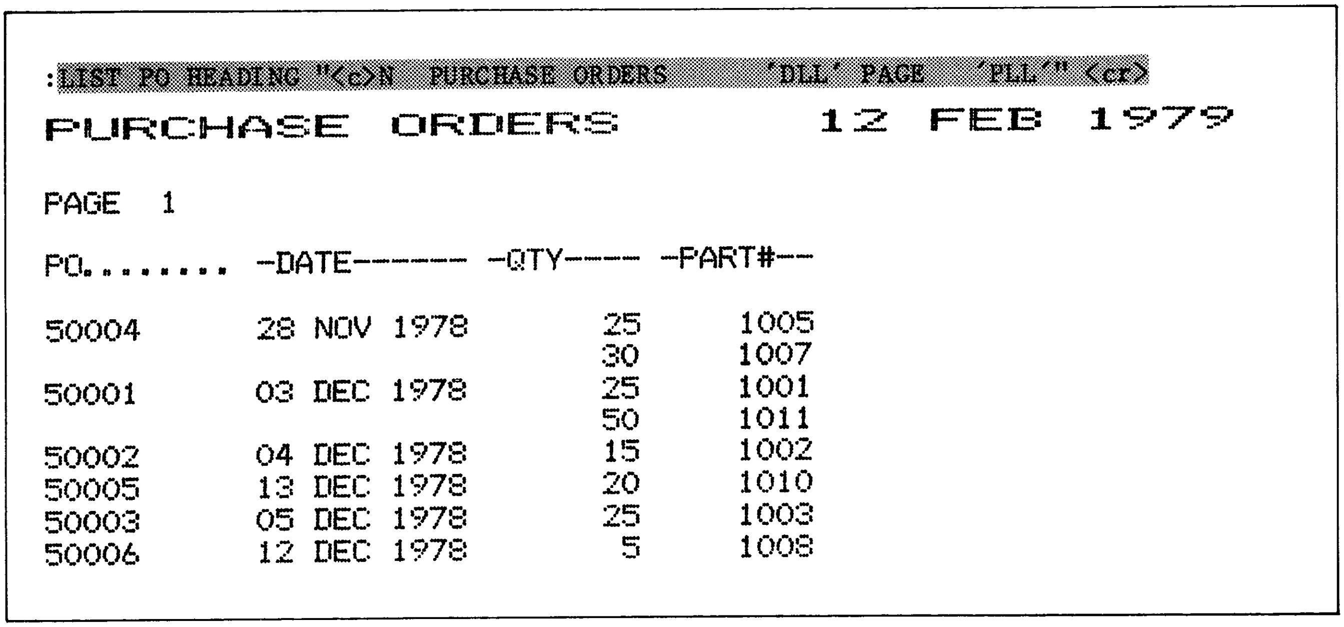
```
"<c>Ntext..."
```

Control N does not print on terminal.

The string that follows up to a carriage return, line feed will print in expanded type. Each character printed in expanded type requires two horizontal spaces. Figure B presents an example.

Character	Meaning
'B'	*BREAK. Inserts the value causing a control-break, if the 'B' option has been specified along with the control-break field (see the section OUTPUT OPTIONS FOR CONTROL-BREAKS). This option has no effect otherwise.*
'D'	*DATE. Inserts the current system data at this point in the heading.*
'F'	*FILE-NAME. Inserts the file-name.*
'L'	*LINE. Specifies start of a new line (carriage return and line feed insertion).*
'N'	*NOPAGE. Defeats automatic paging of output.*
'P'	*PAGE. Inserts the current page number.*
'PP'	*PAGE JUSTIFY. Inserts the current page number right justified in a field of four blanks.*
'T'	*TIME. Inserts the current system time and date.*
''	*TWO successive single quotes are used to print a single quote mark in heading text.*

Figure A. Special HEADING and FOOTING Option Characters



```
:LIST PO HEADING "<c>N  PURCHASE ORDERS       'DLL' PAGE   'PLL'" <cr>
PURCHASE ORDERS                12 FEB 1979

PAGE  1

PO........  -DATE------  -QTY----  -PART#--

50004       28 NOV 1978       25       1005
                              30       1007
50001       03 DEC 1978       25       1001
                              50       1011
50002       04 DEC 1978       15       1002
50005       13 DEC 1978       20       1010
50003       05 DEC 1978       25       1003
50006       12 DEC 1978        5       1008
```

Figure B. Sample Usage of Expanded Print Capability on Matrix Printer

2.14 Generating Totals and Grand Totals

LIST and SORT statements may optionally specify totals.

TOTAL

A LIST or SORT statement can be used to generate a total. A TOTAL specification has this general form:

```
TOTAL attribute-name
```

The TOTAL modifier causes a total to be computed for the attribute whose name immediately follows the word "TOTAL". For example:

```
LIST AFILE TOTAL A7
```

This sentence causes values for attribute A7 to be listed, followed by a total (sum) of these values. On the output, the total is identified by three asterisks (***) in the item-id column. This feature is illustrated in Figure C.

The subject of totaling appears elsewhere in this manual in conjunction with other ENGLISH capabilities. The TOTAL modifier is also used in conjunction with the BREAK-ON modifier to output subtotals, as described in the topic GENERATING SUBTOTALS USING CONTROL BREAKS. See the topic PROCESSING STAGES OF CORRELATIVES AND CONVERSIONS: TOTALS AND SUBTOTALS for information regarding totaling with function correlatives and function conversions.

GRAND-TOTAL

The GRAND-TOTAL modifier permits labeling the grand total field in place of the default '***' notation printed in the item-id field. The general form of the GRAND-TOTAL modifier is:

```
GRAND-TOTAL "text {'options'}..."
```

The options are the same as for control-breaks (see the topic OUTPUT OPTIONS FOR CONTROL-BREAKS). Note that the grand total text may overwrite the actual totals if the text is too long.

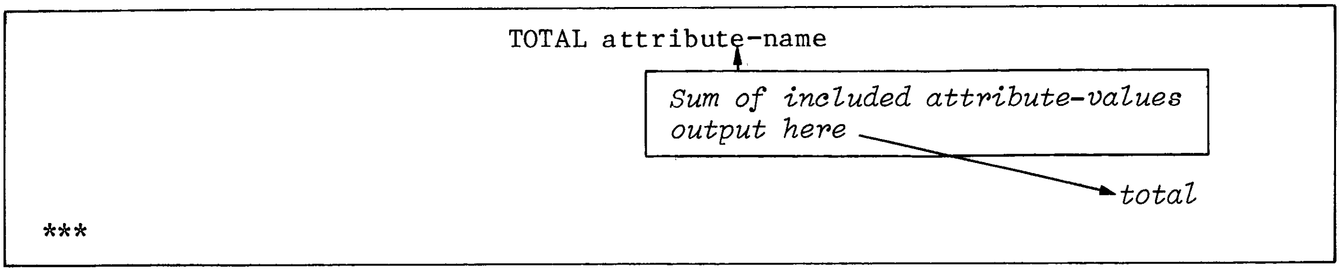


Figure A. General Form of TOTAL Specification and Output Line

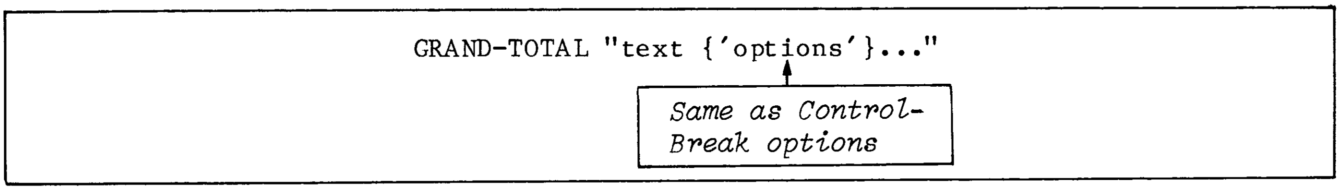
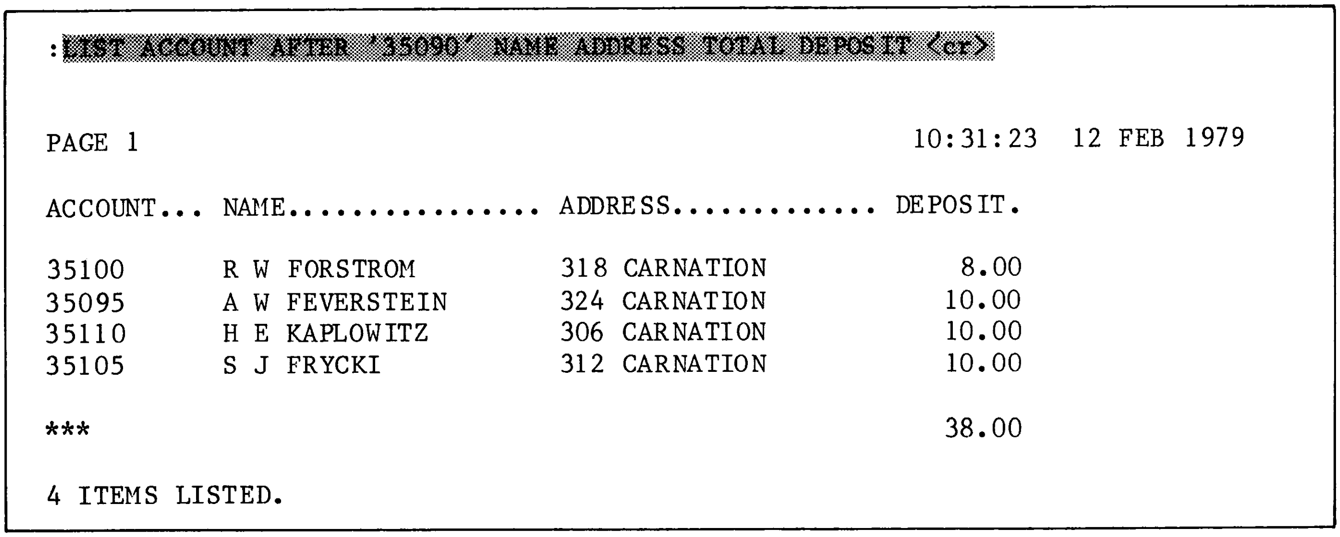

GRAND-TOTAL "text {'options'}..."
Same as Control-Break options

Figure B. General Form of GRAND-TOTAL Specification



```
:LIST ACCOUNT AFTER '35090' NAME ADDRESS TOTAL DEPOSIT <cr>


PAGE 1                                                10:31:23  12 FEB 1979

ACCOUNT... NAME................ ADDRESS............. DEPOSIT.

35100      R W FORSTROM         318 CARNATION            8.00
35095      A W FEVERSTEIN       324 CARNATION           10.00
35110      H E KAPLOWITZ        306 CARNATION           10.00
35105      S J FRYCKI           312 CARNATION           10.00

***                                                     38.00

4 ITEMS LISTED.
```

Figure C. Sample Usage of TOTAL Modifier

2.15 Breaking on Attribute Values

The BREAK-ON modifier may be used to segregate portions of a listing according to the value(s) of the BREAK-ON attribute-name(s).

The BREAK-ON modifier, in its simplest form, has this format:

```
BREAK-ON attribute-name
```

The "attribute name" indicates the attribute on which a break will occur. During the LIST or SORT operation, a control-break occurs whenever there is a change in the value of the specified attribute.

Up to 15 control-breaks are permitted in the sentence; the hierarchy of the breaks is implicitly specified by the sequence of BREAK-ONs in the input line, the first being the highest level.

A break occurs when there is a change in the value of the attribute associated with the BREAK-ON modifier. Value comparison is made on a left-to-right, character-by-character basis, with a maximum of the first 24 characters being used in the comparison. In generating the value for comparison, correlatives in the attribute definition are processed but conversions are not (see applicable subtopics in the chapter titled CORRELATIVES AND CONVERSIONS).

When a control-break occurs, three asterisks (***) are displayed in the BREAK-ON attribute column (i.e., the attribute whose value has changed, thus causing the break).

For multiple control-breaks, output proceeds from lowest level BREAK to highest level. Data associated wth the lowest level control-break is printed on the current page (even if the end of the page has been reached). If multiple control-breaks occur, normal pagination proceeds on the second and subsequent data lines.

The BREAK-ON modifier may be used in conjunction with the TOTAL modifier to generate subtotals (see next topic).

Figure B illustrates the use of the BREAK-ON modifier. Addidtional output formatting capabilities are described in the topic OUTPUT OPTIONS FOR CONTROL BREAKS.

```
BREAK-ON attribute-name
```

Figure A. Minimum BREAK-ON Form

```
:SORT ACCOUNT > '35000' BY STREET NAME BREAK-ON STREET CURR-BALNC <cr>


PAGE                                                 09:34:01  12 FEB 1979

ACCOUNT... NAME................ STREET......... CURR-BALANCE...

35090      D U WILDE            CARNATION       $        884.53
35095      A W FEVERSTEIN       CARNATION       $         19.25
35100      R W FORSTROM         CARNATION
35105      S J FRYCKI           CARNATION       $      5,569.53
35110      H E KAPLOWITZ        CARNATION       $     94,944.55

                                ***

35005      J S ROWE             COVE            $        464.72-
35010      S R KURTZ            COVE            $        467.33
35015      W F GRUNBAUM         COVE            $         88.47
35025      J D GUETZINGER       COVE            $          3.45

                                ***

35030      F M HUGO             DAHLIA          $        123.48
35035      M J LANZENDORPHER     DAHLIA          $        445.89
35040      C E ESCOBAR          DAHLIA          $     38,822.12-
35050      P J WATT             DAHLIA          $        337.18
35055      J W ROMEY            DAHLIA          $     33,478.95

                                ***

35060      J A SCHWARTA         DOCK            $     33,822.34
35065      L J RUFFINE          DOCK            $        558.43
35070      F R SANBORN          DOCK            $     22,144.67
35075      J L CUNNINGHAM       DOCK            $          7.70
35080      G A BUCKLES          DOCK            $    447,765.48
35085      J F SITAR            DOCK            $        200.00

                                ***
***

20 ITEMS LISTED.
```

Figure B. Sample Usage of BREAK-ON Modifier

2.16 Generating Subtotals Using Control-Breaks

The TOTAL modifier may be used with the BREAK-ON modifier for the purpose of generating subtotals in LIST and SORT statements when control-breaks occur.

The TOTAL modifier is used to generate and print subtotal values (in addition to a total) when it appears in the same sentence as BREAK-ON. The form is the same as for generating total, i.e.:

```
TOTAL attribute-name
```

Values for the specified attribute are accumulated and printed as subtotals whenever a control-break occurs. Multiple TOTAL modifiers may appear.

When a control-break occurs, a line of data is output, preceded and followed by blank lines. Three asterisks (***) are displayed in the BREAK-ON attribute column, and a subtotal is displayed in the appropriate column for each attribute specified in a TOTAL modifier. Subtotals are the values accumulated since the last control-break occurred.

At the end of the listing, a TOTAL line is generated for every BREAK specified, and a grand TOTAL line -- as if the TOTAL modifier were used alone -- is also printed. All end of listing sums are printed on the current page.

In computing the value for accumulation, correlatives are processed but conversion specifications are not (see the applicable subtopics in the chapter CORRELATIVES AND CONVERSIONS). Conversion is applied only when the value being accumulated is actually printed.

Figure B illustrates the use of BREAK-ON and TOTAL modifiers. Additional output formatting capabilities are described in the topic OUTPUT OPTIONS FOR CONTROL BREAKS.

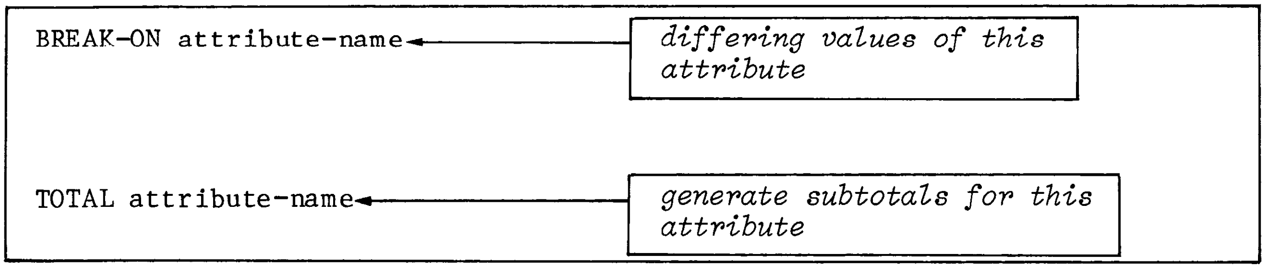
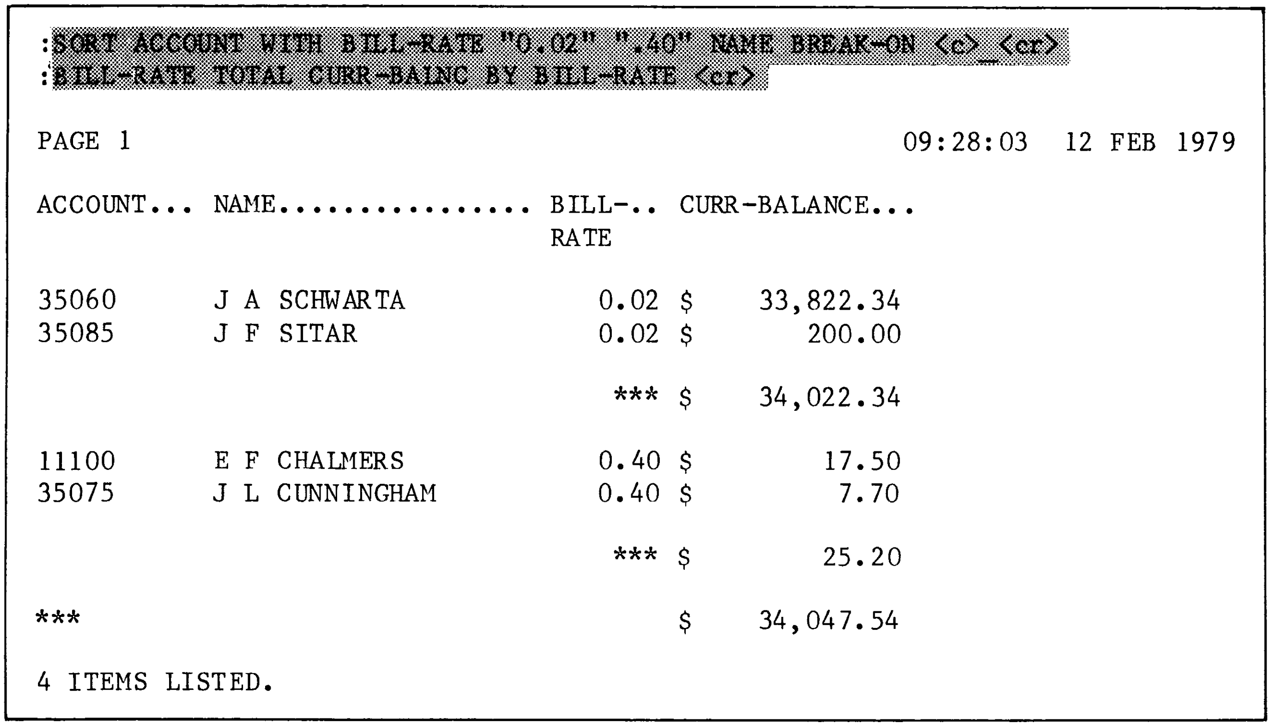

BREAK-ON attribute-name
differing values of this attribute
TOTAL attribute-name
generate subtotals for this attribute

Figure A. Minimum BREAK-ON Form and TOTAL Form



```
:SORT ACCOUNT WITH BILL-RATE "0.02" ".40" NAME BREAK-ON <c>_<cr>
:BILL-RATE TOTAL CURR-BALNC BY BILL-RATE <cr>

PAGE 1                                              09:28:03  12 FEB 1979

ACCOUNT... NAME................ BILL-.. CURR-BALANCE...
                                RATE

35060      J A SCHWARTA            0.02 $     33,822.34
35085      J F SITAR               0.02 $        200.00

                                    *** $     34,022.34

11100      E F CHALMERS            0.40 $         17.50
35075      J L CUNNINGHAM          0.40 $          7.70

                                    *** $         25.20

***                                     $     34,047.54

4 ITEMS LISTED.
```

Figure B. Sample Usage of Control-Breaks to Generate Subtotals

2.17 Output Options for Control Breaks

Labels and output control options may be specified for control-breaks.

A user-generated label can be specified to be printed in place of the default control-break label (***) by following the BREAK-ON attribute-name with the desired label, enclosed in double quotes. Within the label, output control options may be specified enclosed in single quotes. This gives the BREAK-ON specification the following general form:

```
BREAK-ON attribute-name {" {text} {'options'}..."}
```

The text, if specified, replaces the default "***" field in the column in which the control-break occurs. Options are used to modify some of the actions taken at control-break time; options are specified as one or more characters as follows:

BREAK-ON
Label
Option — Meaning

'B' BREAK. Specifies this control-break attribute-name as the one whose value is to be inserted in the ENGLISH page HEADING (or FOOTING) in place of the 'B' option in the HEADING (or FOOTING) specification (see the topic GENERATING HEADINGS AND FOOTINGS). Only the first 24 characters of the attribute are used. This may not be specified in both a HEADING and FOOTING. It may not be meaningful to specify this option in more than one BREAK-ON specification.

'D' DATA. Suppresses the break data line entirely if there was only one detail line since the last control-break occurred.

'L' LINE. Suppresses the blank line preceding the break data line. This option will override the 'U' option described below.

'P' PAGE. Causes a page eject after the data associated with this break has been output.

'R' ROLLOVER. One or more control-break lines occuring at the end of a page will be output on the same page. Without this option, page rollover occurs after printing the first control-break line at the end of a page.

'U' UNDERLINE. Causes underlining of all TOTAL fields.

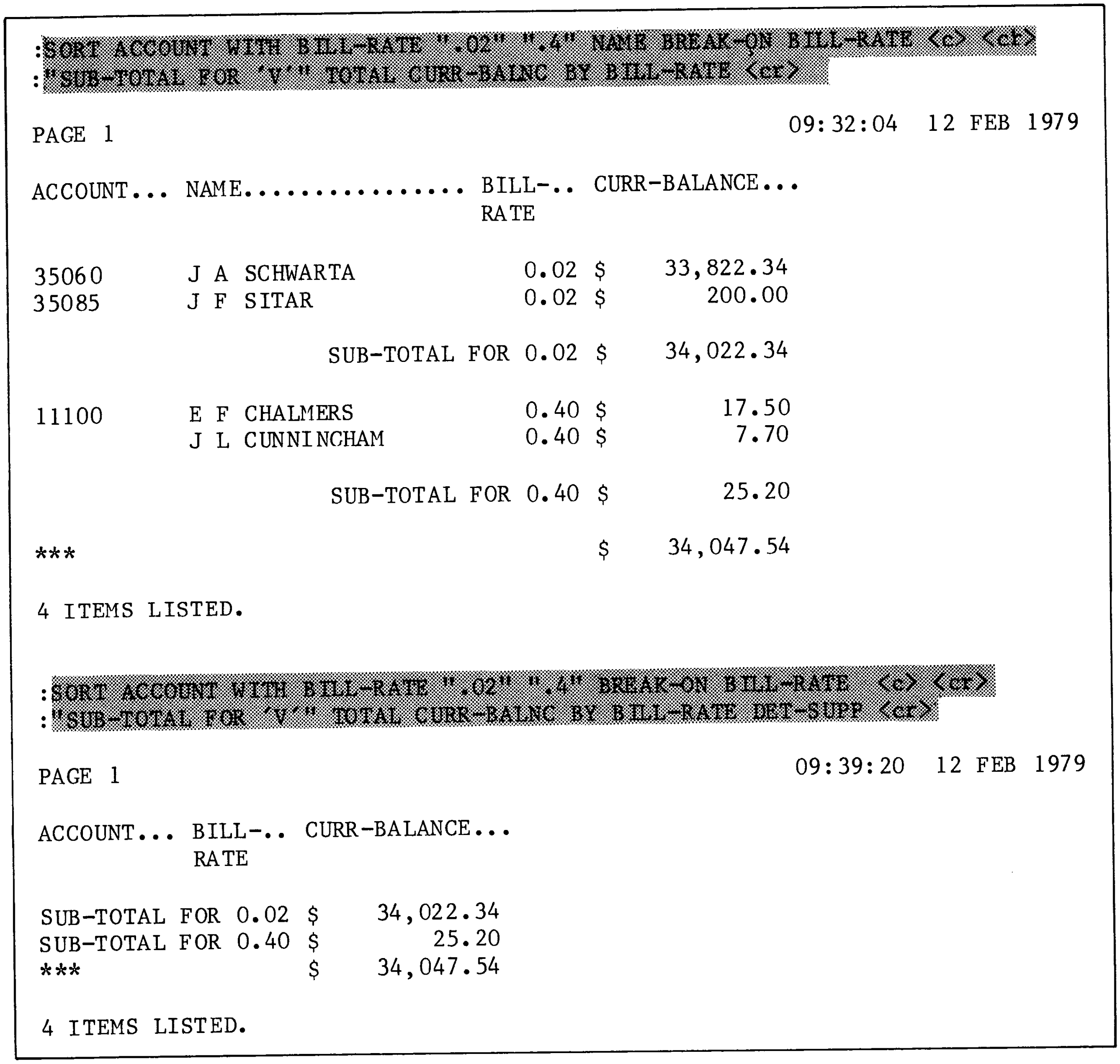
'V' VALUE. Causes the value of the control-break to be inserted at this point in the BREAK-ON label.

The first example in Figure B shows use of output options. If the modifier DET-SUPP is used in the sequence with TOTAL and/or BREAK-ON, then all detail will be suppressed and only the subtotal and total lines will be displayed. This is shown in the second ENGLISH sentence in Figure B. Suppression of the BREAK attribute data may be specified by using a V/MAX (Line 10) of zero for the attribute used with a BREAK-ON modifier (refer to the Reality Programmer's Reference Manual).

```
BREAK-ON attribute-name {" {text} {'options'}..."}

                                        See facing page for
                                        options characters
```

Figure A. General Form of BREAK-ON Specification



```
:SORT ACCOUNT WITH BILL-RATE ".02" ".4" NAME BREAK-ON BILL-RATE <c> <cr>
:"SUB-TOTAL FOR 'V'" TOTAL CURR-BALNC BY BILL-RATE <cr>

PAGE  1                                                09:32:04  12 FEB 1979

ACCOUNT... NAME............... BILL-.. CURR-BALANCE...
                               RATE

35060      J A SCHWARTA          0.02 $     33,822.34
35085      J F SITAR             0.02 $        200.00

                SUB-TOTAL FOR 0.02 $     34,022.34

11100      E F CHALMERS          0.40 $         17.50
           J L CUNNINCHAM        0.40 $          7.70

                SUB-TOTAL FOR 0.40 $         25.20

***                                 $     34,047.54

4 ITEMS LISTED.

:SORT ACCOUNT WITH BILL-RATE ".02" ".4" BREAK-ON BILL-RATE  <c> <cr>
:"SUB-TOTAL FOR 'V'" TOTAL CURR-BALNC BY BILL-RATE DET-SUPP <cr>

PAGE  1                                                09:39:20  12 FEB 1979

ACCOUNT... BILL-.. CURR-BALANCE...
           RATE

SUB-TOTAL FOR 0.02 $     34,022.34
SUB-TOTAL FOR 0.40 $         25.20
***                $     34,047.54

4 ITEMS LISTED.
```

Figure B. Sample Usage of Control-Break Options

2.18 Sublists: WITHIN Connective

The WITHIN connective can be used in an ENGLISH sentence to list a sublist of items belonging to an item.

ENGLISH has the capability to retrieve and list all items which are subitems of a specified item using the WITHIN connective. This capability is useful for bill-of-materials processing and for displaying tree-structured lists. The list may proceed up to 20 sublevels.

One attribute in each item of the file is selected to be a multivalued sublist. Each value is the item-id of a subitem which must also be in the file.

The DL/ID of the file must have V code on line 8. The form is:

```
V;;sub-list-AMC
```

(Note the double semicolon). "Sub-list-AMC" is the attribute number that contains the sublist.

The WITHIN connective functions only with LIST and COUNT verbs. The WITHIN connective must immediately precede the file-name. One, and only one, explicit item-id must be specified in the input line.

On a columnar listing, a field five characters in width, with a heading of "LEVEL" will be used to print the level number. On noncolumnar listings, the level number will precede each item.

Figure B describes the dictionary of the ASSEMBLIES file. Note the 'V' code on Line 8 of the DL/ID item. Figure C illustrates the use of the WITHIN connective.

```
WITHIN {DICT} file-name
```

Figure A. General Form of the WITHIN Connective

```
:LIST DICT ASSEMBLIES <cr>

PAGE 1                                           08:39:17  12 FEB 1979

ASSEMBLIES D/CODE A/AMC S/NAME......... V/CONV.... V/CORR......... V/TYP V/MAX

PART#       S          0 PART #                                     L        10
DESC        S          1 DESCRIPTION                                L        20
QOH         S          2 ON-HAND                                    R         4
VALUE       S          3 VALUE          MD2                         L         6
REC.DATE    S          4 REC.DATE       D                           L         9
SUB.ASSEM   S          5 SUB.ASSEM                                  L        10
LOCATION    S          6 LOCATION                                   L        11
DL/ID       D      30099 1                         V;;5             L        10

8 ITEMS LISTED.
```

Figure B. Dictionary of ASSEMBLIES File

```
:LIST WITHIN ASSEMBLIES 'A2000-1234' PART# DESC VALUE LOCATION <c> <cr>
:SUB.ASSEM QOH ID-SUPP <cr>

PAGE 1                                         08:41:06  12 FEB 1979

LEVEL PART#..... DESCRIPTION......... VALUE. LOCATION... SUB.ASSEM. ON-HAND

1     A2000-1234 SERVOS                 0.73 R-123-8888  A2001-7811      73
                                                         A2001-8900
                                                         A2001-9112
2     A2001-7811 D.C.MOTOR              0.55 R-17-1001   A2002-1000      55
                                                         A2002-1023
3     A2002-1000 D.C.MOTOR PLATFORM     0.73 R-123-8888                  73
3     A2002-1023 D.C.MOTOR POWER UNIT   0.73 R-123-1002                  73
2     A2001-8900 SERVO BOARD            0.12 L-44-1001                   12
2     A2001-9112 SERVO HOUSING          1.07 L-17-189    A2002-1032     107
                                                         A2002-1566
3     A2002-1032 HOUSING SEALS          1.02 L-09-1889                  102
3     A2002-1566 HOUSING PLATES         1.03 L-1-3309    A2004-1111     103
4     A2004-1111 HOUSING PACKAGE       12.00 R-12-1212                 1200

9 ITEMS LISTED.
```

Figure C. Sample Usage of WITHIN Connective

3.1 LIST Verb

LIST is an ENGLISH verb used to generate a formatted output of selected items and attributes from a specified file.

An ENGLISH sentence using the LIST verb is constructed as illustrated in Figure A. The selected items (and their associated selected attributes) will be listed at the terminal (or on the printer if the modifier LPTR or the 'P' option is used). The sequence of the output listing will be the order in which the item-ids have been enumerated in the ENGLISH sentence. If no item-ids have been specified in the ENGLISH sentence, then all item-ids are implied and LIST will present these items in the hash sequence in which they are stored in the file.

Generated output will be formatted into a columnar output (if possible) taking into account the maximum defined size of the specified attributes and their associated names, along with the width of the terminal page as defined by the TCL TERM verb (refer to the Reality Programmer's Reference Manual). If more attributes have been specified than will fit across the page, a noncolumnar output will be generated with the attribute names down the side and the associated attribute values to the right. For further details regarding the output format, refer to the topic FORMING OUTPUT-SPECIFICATIONS and the topic OMISSION OF THE OUTPUT- SPECIFICATION.

Consider the followng example:

```
    LIST ACCOUNT '35000' '35050' NAME ADDRESS
```

This sentence specifies that the attribute values for attributes NAME and ADDRESS in items 35000 and 35050 (in ACCOUNT file) are to be listed. In this case a columnar output will be produced.

Further examples of the LIST verb are shown in Figure B.

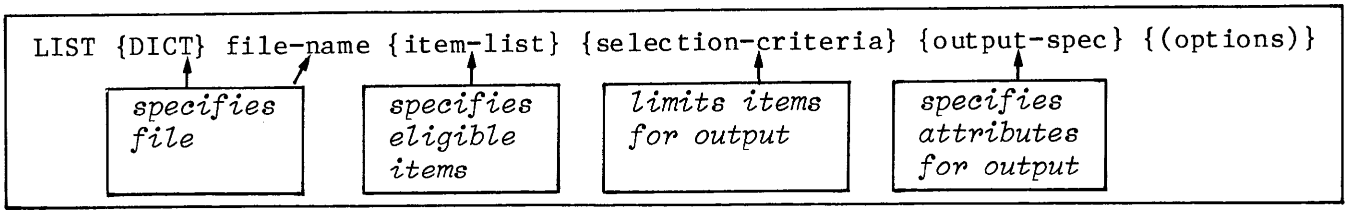

LIST {DICT} file-name {item-list} {selection-criteria} {output-spec} {(options)}
specifies file
specifies eligible items
limits items for output
specifies attributes for output

Figure A. General Form of ENGLISH Sentence Using LIST Verb

```
:LIST ACCOUNT WITH BILL-RATE "0.3" NAME ADDRESS BILL-RATE <cr>

PAGE 1                                               11:08:37  12 FEB 1979
ACCOUNT... NAME............... ADDRESS............. BILL-..
                                                    RATE


11115      D R MASTERS         100 AVOCADO             0.30
11085      A B SEGUR           101 BAY STREET          0.30
11040      E G MCCARTHY        113 BEGONIA             0.30
11050      J R MARSHECK        125 BEGONIA             0.30
11020      J T O'BRIEN         124 ANCHOR PL           0.30
11095      J B STEINER         124 AVOCADO             0.30
11110      D L WEISBROD        106 AVOCADO             0.30
11015      L K HARMAN          118 ANCHOR PL           0.30
11105      C C GREEN           112 AVOCADO             0.30
11090      J W JENKINS         130 AVOCADO             0.30
23030      L J DEVOS           201 CARNATION           0.30

11 ITEMS LISTED.

:LIST ACCOUNT > '23080' AND <= '23095' NAME ADDRESS <c> <cr>
:START-DATE CURR-BALNC DEPOSIT <cr>

PAGE 1                                               11:19:58  12 FEB 1979

ACCOUNT   :  23095
NAME    W E ZUMSTEIN
ADDRESS    224 BEGONIA
START-DATE    01 JAN 1968
DEPOSIT    11.00

ACCOUNT  :  23090
NAME    W J HIRSCHFIELD
ADDRESS    230 BEGONIA
START-DATE     01 JAN 1968
CURR-BALNC     $  20.45
DEPOSIT    10.00

3 ITEMS LISTED.
```

Figure B. Sample Usage of List Verb

3.2 SORT Verb

SORT is an ENGLISH verb used to generate a sorted and formatted output of selected items and attributes from a specified file.

An ENGLISH sentence using the SORT verb is constructed as illustrated in Figure A. The output produced by a SORT operation is identical to the output produced by a LIST operation (refer to topic LIST VERB), except that a SORT operation orders the output in a specified sorted order. If no BY-type modifier is used, then the SORT will sequence on the item-ids (in ascending order). If the BY modifier is used, then the SORT will sequence on the attribute whose name immediately follows BY (in ascending order). If the BY-DSND modifier is used, then the sort will sequence on the attribute whose name immediately follows BY-DSND (in descending order). The modifier BY-EXP specifies an ascending exploding sort on the attribute whose name immediately follows. BY-EXP-DSND specifies a descending exploding sort. A sort will be performed whenever a BY clause is specified, regardless of the verb used.

Multiple BY and BY-DSND sort keys may be intermixed at will, with the leftmost sort key being most significant. If a descendng sort is required on the item-id alone, then a BY-DSND modifier must be used followed by an attribute synonymous to the item-id (i.e., one whose A/AMC is zero).

The modifiers BY-EXP and BY-EXP-DSND specify exploding sorts on multivalued attributes. Individual multivalues are treated as independent single values. Each is associated with its item-id. The expanded list is then sorted. If necessary, items are printed more than once to maintain sequential order. For more details, see the topic EXPLODING SORT: MULTIVALUED ATTRIBUTES.

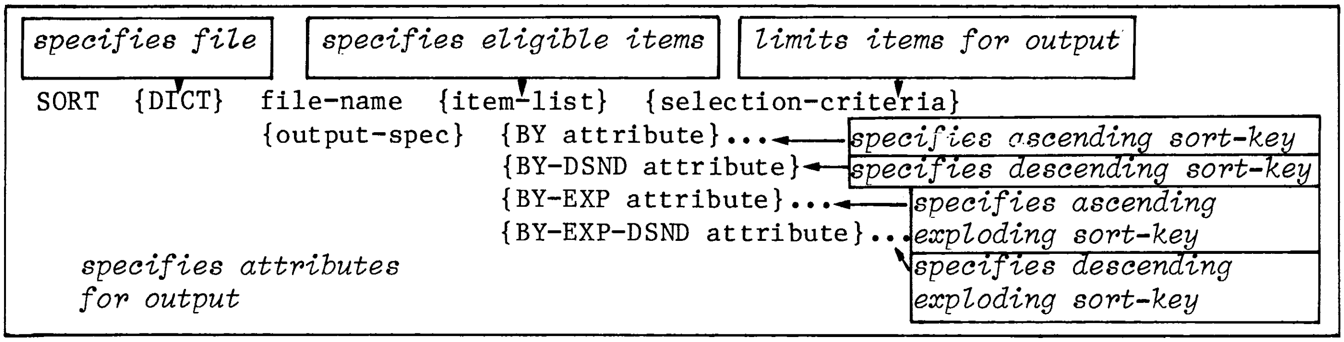
Sequencing via a SORT operation is accomplished by comparing the ASCII values of the characters from left to right. If attributes are right-justified, then the leftmost empty character postions are considered as blanks when compared. For further information regarding character comparison, refer to the topic USING RELATIONAL OPERATORS AND LOGICAL CONNECTIVES. Consider the following example:

```
    SORT INV BY QUAN BY PRICE
```

This sentence specifies an ascending sort of file INV, with primary sequencing performed on the attribute QUAN and secondary sequencing performed on attribute PRICE. Further examples are shown in Figure B.

The SORT verb may call on the correlative processor and the conversion processor. All correlative codes are processed in forming sort keys (refer to the appropriate topic in the chapter CORRELATIVES AND CONVERSIONS). Note that several codes (MD, MT, D) do not affect the results of sorting, and should be used as conversion codes only, in order to save processing time.

The default sort on the item-id may be left-justified string or right-justified numeric depending on Attribute 9 of the D-pointer (or DL/ID) defining the file.



```
SORT  {DICT}  file-name  {item-list}  {selection-criteria}
                {output-spec}  {BY attribute}...
                               {BY-DSND attribute}
                               {BY-EXP attribute}...
                               {BY-EXP-DSND attribute}...
```

specifies file
specifies eligible items
limits items for output
specifies ascending sort-key
specifies descending sort-key
specifies ascending exploding sort-key
specifies descending exploding sort-key
specifies attributes for output

Figure A. General Form of ENGLISH Sentence Using SORT Verb

```
:SORT ACCOUNT GE '23000' AND LE '23020' NAME START-DATE <cr>

PAGE 1                                             14:11:02  12 FEB 1979

ACCOUNT... NAME................ START-DATE..

23000      H T LEE              01 JAN 1968
23005      W B THOMPSON         29 DEC 1969
23010      W E MCCOY            01 JAN 1968
23015      R M COOPER           01 JAN 1968
23020      S L UNGERLEIDER      23 APR 1972

5 ITEMS LISTED.

:SORT ACCOUNT WITH CURR-BALNC > "95000" NAME CURR-BALNC <c> <cr>
:BY-DSND CURR-BALNC <cr>

PAGE                                               14:13:37  12 FEB 1979

ACCOUNT... NAME................ CURR-BALANCE...

11055      W H KOONS            $   958,343.75
35080      G A BUCKLES          $   447,765.48
11020      J T O'BRIEN          $   306,755.54
23040      P B SCIPMA           $   123,423.22
23045      P F KUGEL            $    99,422.34

5 ITEMS LISTED.

:LIST ACCOUNT AFTER '35070' NAME DEPOSIT BILL-RATE <c> <cr>
:BY DEPOSIT BY BILL-RATE <cr>

PAGE 1                                             14:15:47  12 FEB 1979

ACCOUNT... NAME................ DEPOSIT. BILL-..
                                         RATE

35090      D U WILDE                3.17
35100      R W FORSTROM             8.00   10.03
35080      G A BUCKLES             10.00    0.35
35095      A W FEVERSTEIN          10.00    0.35
35105      S J FRYCKI              10.00    0.35
35075      J L CUNNINGHAM          10.00    0.40
35110      H E KAPLOWITZ           10.00   10.03
35085      J F SITAR               12.00    0.02

8 ITEMS LISTED.
```

Figure B. Sample Usage of SORT Verb

3.3 Exploding Sort: Multivalued Attributes

The exploding sort allows a sort to be performed on a multivalued attribute. An exploding sort is specified by a BY-EXP or BY-EXP-DSND modifier.

The exploding sort will generate a separate sort key for each value for each item. The number of detail lines in a listing will thus be the total number of values in the specified attribute including all items.

When a listing is made using the exploding sort, a given item will appear on as many detail lines as it has values. Only one value of each multivalued attribute will be printed on that detail line. That will be the value which corresponds to the value in the sort key.

The modifiers BY-EXP and BY-EXP-DSND can be used to specify the attribute on which the exploding sort is applied. BY-EXP is for ascending sort, and BY-EXP-DSND is for descending sorts.

Figure A shows a patient file as it might be used in a clinic. The attributes 'DATE', 'TEST', and 'CHARGE' describe tests which were charged to a patient. These are multivalued because the patients returned for tests on more than one occasion. The exploding sort by charge in Figure B is a report showing how frequently patients have expensive (or inexpensive) tests. Figure C shows the dictionary elements used in the report.

The exploding sort capability can also be used with the SSELECT verb.

NOTE: DATA/BASIC has been expanded to use the exploding sort capability. The READNEXT statement will now allow the form 'READNEXT variable, variable ELSE statement(s)' in which value numbers are entered into the second variable, indicating the positional relationship of the multivalve within the attribute.

```
:LIST PATIENT NAME DATE TEST CHARGE SUPP <cr>


PATIENT... NAME........ DATE........ TEST CHARGE
                                     *    *

1003       GEORGE WONG  16 MAR 1976  BLD  30.00
                        23 JUN 1973  BLD  30.00
                        21 MAR 1976  EKG  70.00
                        06 OCT 1976  ECP  85.90
1022       ALICE DETT   21 APR 1974  EKG  70.00
                        28 MAR 1976  ECP  85.90
                        30 MAR 1976  BLD  18.00-
2011       TOM GEREAU   13 MAR 1976  BLD  30.00
                        17 APR 1976  BLD  30.00-
3003       RAY MANO     07 APR 1976  ECP  85.90
                        16 JUL 1976  BLD  30.30
                        25 AUG 1976  EKG  70.00
                        06 OCT 1976  EKG  70.00
4001       WALT WAXLER  29 DEC 1975  EKG  70.00
                        18 JAN 1976  EKG  70.00
                        25 JAN 1976  EKG  70.00
                        01 FEB 1976  EKG  70.00
```

Figure A. Listing of PATIENT File

```
:SORT PATIENT BY-EXP-DSND CHARGE NAME DATE TEST CHARGE SUPP <cr>
PATIENT... NAME........ DATE........ TEST CHARGE
                                     *    *

1003       GEORGE WONG  06 OCT 1976  ECP  85.90
1022       ALICE DETT   28 MAR 1976  ECP  85.90
3003       RAY MANO     07 APR 1976  ECP  85.90
1003       GEORGE WONG  21 MAR 1976  EKG  70.00
1022       ALICE DETT   21 APR 1974  EKG  70.00
3003       RAY MANO     25 AUG 1976  EKG  70.00
3003       RAY MANO     06 OCT 1976  EKG  70.00
4001       WALT WAXLER  29 DEC 1975  EKG  70.00
4001       WALT WAXLER  18 JAN 1976  EKG  70.00
4001       WALT WAXLER  25 JAN 1976  EKG  70.00
4001       WALT WAXLER  01 FEB 1976  EKG  70.00
3003       RAY MANO     16 JUL 1976  BLD  30.30
1003       GEORGE WONG  16 MAR 1976  BLD  30.00
1003       GEORGE WONG  23 JUN 1973  BLD  30.00
2011       TOM GEREAU   13 MAR 1976  BLD  30.00
1022       ALICE DETT   30 MAR 1976  BLD  18.00-
2001       TOM GEREAU   17 APR 1976  BLD  30.00-
```

Patient 1003 appears once for each multi-value of CHARGE

Figure B. Exploded Sort by Descending CHARGE

```
PATIENT... D/CODE A/AMC S/NAME......... V/CONV.... V/CORR......... V/TYP V/MAX

NAME       S          1                                            T        12
DATE       S          2                 D          D1;3;4          L        12
TEST       S          3                            D2;2            L         4
CHARGE     S          4                 MD2-       D2;2            R         6
```

Figure C. Dictionary of PATIENT File

3.4 LIST-LABEL and SORT-LABEL Verbs

LIST-LABEL and SORT-LABEL are ENGLISH verbs that may be used to generate data to print mailing labels or to produce other special purpose listings.

LIST-LABEL is equivalent to the LIST verb. SORT-LABEL is equivalent to the SORT verb, performing a sort on the data, as specified by any sort key specifications in the statement. The sequence of attributes specified in the statement will determine the sequence of data generated. All correlatives and conversion specifications will be operative (see the chapter CORRELATIVES AND CONVERSIONS).

The data generated will consist of the item-id (unless the ID-SUPP modifier is used), followed by the data corresponding to each attribute specification in the statement. Multivalues for an attribute will be treated as if they were separate value fields; thus, for most applications, multivalued attributes should not be specified.

The normal noncolumnar list heading (page number, time and date) will print on the top of each page unless suppressed by using the COL-HDR-SUPP modifier. If COL-HDR-SUPP is used, pagination and all top-of-forms will be suppressed, which essentialy produces a continuous format without page breaks.

Before starting the data output, one or more lines of parametric information will be requested. The first line is used to specify the format of the labels. Six numeric parameters are required, as follows:

```
?cols,rows,skip,indent,size,space{,C}
```

where: cols — is the number of items across the page (repeat count)
rows — is the number of print lines per label
skip — is the number of lines to skip between labels
indent — is the number of spaces to indent the data on the left
size — is the maximum width of each label value (truncation factor)
space — is the number of horizontal spaces between labels
C — is optional; if present, specifies that null or missing attribute data are to be ignored, thereby compressing the data structure. If not specified, null or missing values will be treated as all blanks.

The "rows" count must be a minimum number of one for each attribute specified, plus one for the item-id if ID-SUPP is not used.

Values must conform to the range:

```
cols * (size + space) + indent = current page width
```

If "indent" is non-zero, a set of "row header" data lines will be requested, these are printed to the left of each line, in the "indent" area. Null headers may be specified by entering null lines to the header data requests.

```
:LIST-LABEL ACCOUNTS-PAYABLE NAME ADDRESS CITY-ST-ZIP ID-SUPP (P) <cr>

?2,3,3,10,25,5 <cr>
?NAME <cr>
?ADDRESS <cr>
?CITY-ST <cr>


NAME          CALIFORNIA MAGNETICS          PRECISION METAL PRODUCTS
ADDRESS       2451 W. CHAPMAN AVE.          30288 JAMBOREE ROAD
CITY-ST       ANAHEIM, CA 92802             NEWPORT BEACH, CA 92660


NAME          PLEXI-PLASTICS                SURF RENTALS
ADDRESS       39200 YORBA LINDA BLVD.       3451 S. BEACH BLVD.
CITY-ST       PLACENTIA, CA 92670           HUNTINGTON BEACH, CA


NAME          ABC PRODUCTS                  QUALITY OFFICE SUPPLY
ADDRESS       2453 E. BAY STREET            3952 LA PALMA AVE.
CITY-ST       BREA, CA 92601                BUENA PARK, CA 90620


NAME          POWER ELECTRONICS             CONTACT SWITCHES
ADDRESS       7298 S. BROOKHURST STREET     39122 PACIFIC COAST HWY.
CITY-ST       GARDEN GROVE, CA 92644        CORONA DEL MAR, CA 92625


NAME          LIGHTNING ELECTRONICS         C & S PAPER SUPPLIES
ADDRESS       4390 ORANGETHORPE AVE.        3509 S. HARBOR BLVD.
CITY-ST       FULLERTON, CA 92631           COSTA MESA, CA 92627


NAME          PNP TRANSISTORS               ZIP DELIVERY SERVICE
ADDRESS       2432 E. 17TH STREET           2458 S. EUCLID STREET
CITY-ST       TUSTIN, CA 92680              LA HABRA, CA 90631
```

Figure A. Sample Usage of LIST-LABEL

3.5 COUNT Verb

COUNT is an ENGLISH verb which counts the number of items meeting the conditions specified by the combination of item-list and selection-criteria.

An ENGLISH sentence using the COUNT verb is illustrated in Figure A. The COUNT operation will count the number of items which meet conditions specified by the item-list and selection-criteria. The COUNT produces the following output:

```
xxx ITEMS COUNTED
```

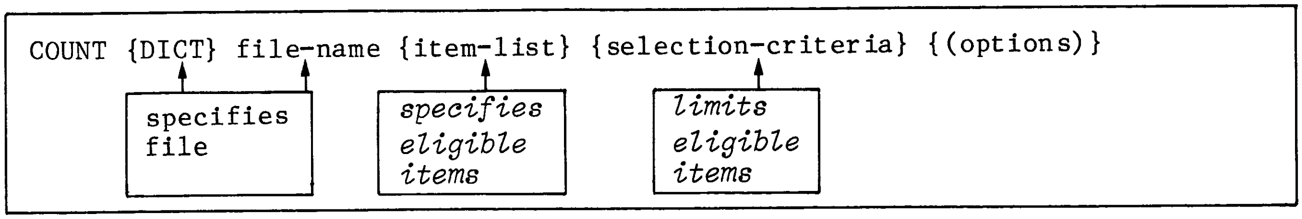
where "xxx" is the number of items counted. The maximum number of items which can be counted is 2,147,483,647.

Consider the following example:

```
COUNT AFILE > '533' WITH A3 = "ABC"
```

This sentence counts the items which have item-ids greater than 533 and which have values of ABC for Attribute A3.

Further examples of the COUNT operation are presented in Figure B.



```
COUNT {DICT} file-name {item-list} {selection-criteria} {(options)}
```

Figure A. General Form of ENGLISH Sentence Using COUNT Verb

```
:COUNT TEST <cr>

10 ITEMS COUNTED.


:COUNT DICT ACCOUNT WITH D/CODE "A" <cr>

55 ITEMS COUNTED.


:COUNT ACCOUNT WITH BILL-RATE "30" <cr>

11 ITEMS COUNTED.


:COUNT ACCOUNT GE '11115' WITH CURR-BALNC AND WITH BILL-RATE "30" <cr>

2 ITEMS COUNTED.


:COUNT ACCOUNT WITH NO SEWER-ASMT <cr>

57 ITEMS COUNTED.
```

Figure B. Sample Usage of COUNT Verb

3.6 SUM and STAT Verbs

SUM is an ENGLISH verb which generates a total sum for one specified attribute. STAT is an ENGLISH verb which generates a total sum, an average, and a count for one specified attribute.

ENGLISH sentences using the SUM and STAT verbs are illustrated in Figure A.

SUM

The SUM operation generates a total sum for the specified attribute. The output produced by a SUM operation has the following general form:

```
    TOTAL OF aaaa IS :  xxxx
```

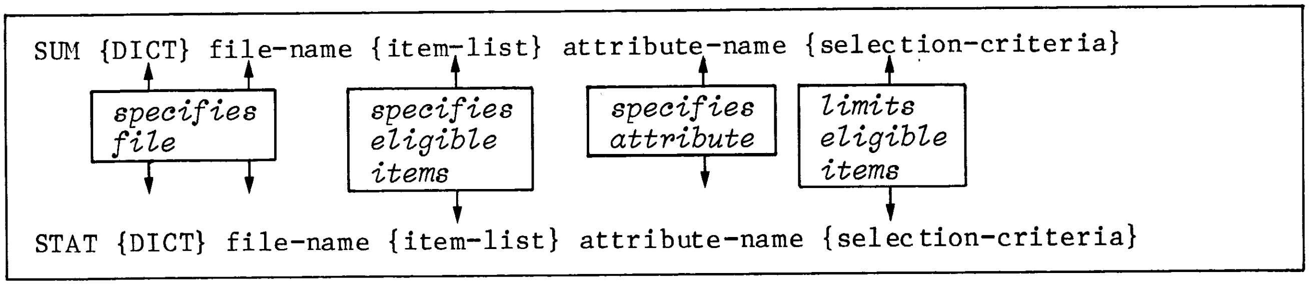
where "aaaa" is the attribute name and "xxxx" is the computed total. Figure B illustrates the use of this verb.

STAT

The STAT operation generates a total sum, an average, and a count for the specified attribute. The output produced by a STAT operation has the following general form:

```
    STATISTICS OF aaaa :
    TOTAL = xxxx AVERAGE = yyyy COUNT = zzzz
```

where "aaaa" is the attribute name, "xxxx" is the total sum, "yyyy" is the average, and "zzzz" is the count. Figure C illustrates the use of this verb.



```
SUM {DICT} file-name {item-list} attribute-name {selection-criteria}

STAT {DICT} file-name {item-list} attribute-name {selection-criteria}
```

Figure A. General Form of ENGLISH Sentence Using SUM or STAT Verbs

```
:SUM ACCOUNT CURR-BALNC <cr>
TOTAL OF CURR-BALNC IS  :  $2,405,129.91

:SUM ACCOUNT CURR-BALNC WITH CURR-BALNC > "100000" <cr>
TOTAL OF CURR-BALNC IS  :  $1,836,287.99

:SUM ACCOUNT > '35055' CURR-BALNC <cr>
TOTAL OF CURR-BALNC IS  :  $605,916.48

:SUM DICT ACCOUNT V/MAX <cr>
TOTAL OF V/MAX IS  :  2649

:SUM ACCOUNT DEPOSIT WITH CURR-BALNC < "50000" AND WITH NO SEWER-ASMT <cr>
TOTAL OF DEPOSIT IS  :  499.00
```

Figure B. Sample Usage of SUM Verb

```
:STAT ACCOUNT <cr>
STATISTICS OF ACCOUNT  :
TOTAL = 16199 AVERAGE = 241.77 COUNT = 67

:STAT ACCOUNT TRASH-CHGS <cr>
STATISTICS OF TRASH-CHGS  :
TOTAL = 990.94 AVERAGE = 13.4468 COUNT = 67

:STAT ACCOUNT CURR-BALNC WITH TRASH-CHGS GE "7.4255" <cr>
STATISTICS OF CURR-BALNC  :
TOTAL = $1,199,466.82 AVERAGE = $ 57,117.4676 COUNT = 21

:STAT ACCOUNT '11065' '23055' '35050' '35085' BILL-RATE <cr>
STATISTICS OF BILL-RATE  :
TOTAL = 20.43 AVERAGE = 5.1075 COUNT = 4

:STAT ACCOUNT DEPOSIT WITH NO CURR-BALNC <cr>
STATISTICS OF DEPOSIT  :
TOTAL = 39.00 AVERAGE = 9.7500 COUNT = 4
```

Figure C. Sample Usage of STAT Verb

3.7 SELECT and SSELECT Verbs

SELECT is an ENGLISH verb which provides the facility to select a set of items using the item-list and selection-criteria. The SSELECT verb combines the SORT capability with the SELECT capability.

SELECT and SSELECT provide a facility to select a set of items, using the full ENGLISH selection-criteria. These selected items (item-list) are available, one at a time, to other processors such as DATA/BASIC, TCL-II EDITOR, PROC, and the ENGLISH processors. In all cases, one can select from one file and use the item-ids to access another file.

SELECT is analogous to the LIST verb in that there is no sequencing of the items. SSELECT is analogous to the SORT verb, and a sort will be performed as specified by any sort key specifications in the statement. The output from either statement will be a message indicating the number of items selected, in the form:

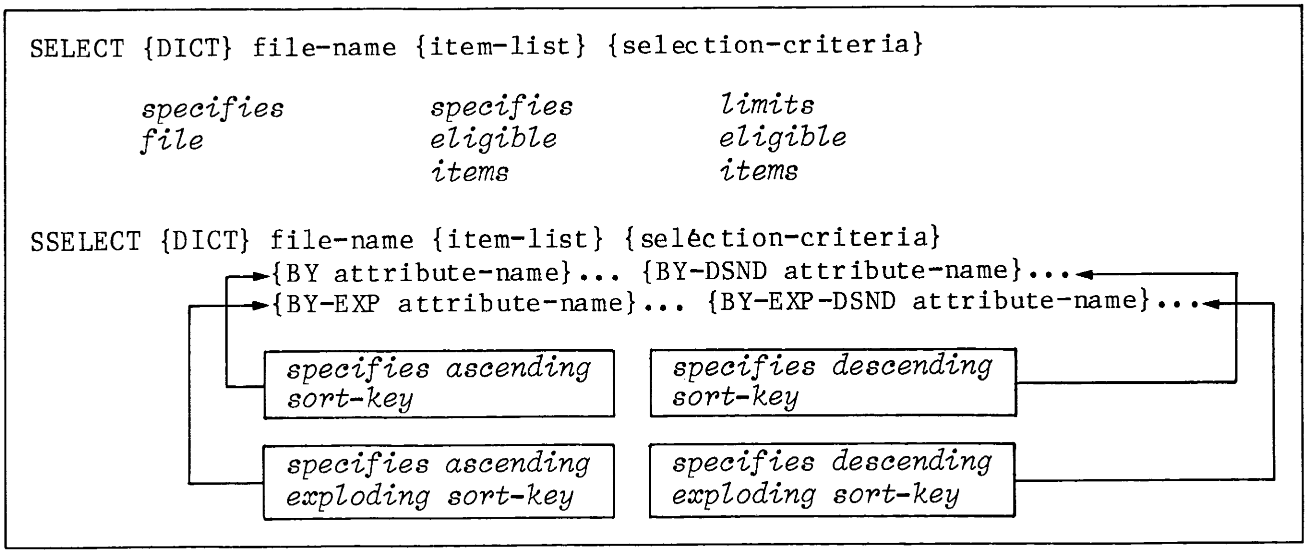
```
    xxx ITEMS SELECTED
    :
```

The selected items are now available to other processors, as follows:

BASIC program	Selected items are available to the DATA/BASIC program via the READNEXT statement (refer to the DATA/BASIC Programming Manual).
ENGLISH process	The statement is entered without an item-list (for instance, "LIST PARCEL-FILE NAME ADDRESS"); the selected item-list is used. The regular ENGLISH attribute selection criteria is applicable; however, selection on the item-ids is not. A number of verbs are provided to manipulate selected lists (refer to the following topic).
TCL-II process	The statement is entered without an item-list (for instance "COPY DICT PARCEL-FILE (P)"). (Refer to the Reality Programmer's Reference Manual).
PROC process	The SELECT (or SSELECT) may be processed within a PROC which can invoke another processor such as DATA/BASIC, the EDITOR or the COPY processor, for example. In the new PQN-type PROCs, the selected list of item-ids may be accessed with a variety of PROC commands for special processing. The possibilities are virtually unlimited here.

The statement that uses the selected item-list must immediately follow the SELECT or SSELECT statement; any other statement will result in the loss of the item-list. If the SELECT or SSELECT is generated by a PROC, the statement that uses the item list must be "stacked" by the PROC (using "STON") (refer to the PROC Programming Manual). Commands that can utilize a selected item-list have an asterisk next to the item-list parameter in the general form.

Note that some of the available disc space will be used to store the selected list of item-ids. This space will be made available once again after the list has been processed.



```
SELECT {DICT} file-name {item-list} {selection-criteria}

SSELECT {DICT} file-name {item-list} {selection-criteria}
    {BY attribute-name}... {BY-DSND attribute-name}...
    {BY-EXP attribute-name}... {BY-EXP-DSND attribute-name}...
```

specifies file
specifies eligible items
limits eligible items
specifies ascending sort-key
specifies descending sort-key
specifies ascending exploding sort-key
specifies descending exploding sort-key

Figure A. General Form of ENGLISH Sentence Using SELECT or SSELECT Verb

```
:SELECT ACCOUNT WITH SEWER-ASMT <cr>

11 ITEMS SELECTED
:EDIT ACCOUNT <cr>

:SELECT ACCOUNT > '11045' WITH CURR-BALNC LE "0" <cr>

3 ITEMS SELECTED.
:COPY ACCOUNT (P) <cr>
```

Figure B. Sample Usage of Select Verb

```
:SSELECT ACCOUNT WITH BILL-RATE "0.30" BY NAME <cr>

11 ITEMS SELECTED.
:RUN BP TEST <cr>

:SSELECT ACCOUNT > '11025' WITH DEPOSIT "10" BY-DSND ADDRESS <cr>

63 ITEMS SELECTED.
:ED ACCOUNT <cr>

:SSELECT ACCOUNT BY BILL-RATE BY-DSND DEPOSIT BY NAME <cr>

67 ITEMS SELECTED.
:COPY ACCOUNT (T,X) <cr>
```

Figure C. Sample Usage of SSELECT Verb

3.8 SAVE-LIST, GET-LIST, COPY-LIST, and DELETE-LIST Verbs

The verbs SAVE-LIST, GET-LIST, COPY-LIST, and DELETE-LIST are used to save, retrieve, copy, and delete selected item-id lists.

These verbs are useful if several passes are to be made on an item-list. These verbs bypass the time consuming retrieve and sort phase of the SSELECT verb. Pointers to the saved list are stored in the file called POINTER-FILE. These pointers are saved and restored by the save/restore processors.

The general form of the SAVE-LIST verb is:

```
SAVE-LIST {name}
```

This is entered immediately after a SELECT, SSELECT, or FORM-LIST statement (must be "stacked" if the sequence is generated by a PROC;see the PROC Programming Manual). The optional parameter "name" is any string of nonblank characters the user may specify to identify this selected list. An item with an item-id "account-name*L*name" will be created in the POINTER-FILE; any previously existing item-list with the same name is overlaid, and the disc space it represents will be returned to the system. "Account-name" is the name of the account the user is logged onto at the time of issuing this command.

The general form of the GET-LIST verb is:

```
GET-LIST {name {account-name}}
```

This statement retrieves a previously saved item-list, just as if the user entered a SELECT or SSELECT statement again. A message indicating the number of items in the item-list will be printed, and the item-ids are available one at a time to other processors. If the GET-LIST is generated by a PROC, the statement that uses the item-list must be "stacked" (refer to the PROC Programming Manual). The optional "account-name" allows one user to access an item-list generated and saved by another user.

The general form of the COPY-LIST verb is:

```
COPY-LIST {name {account-name}} {(options)}
```

This statement allows a prestored item-list to be copied to another name (and/or account-name) or to place the item-ids into a file item. Like the COPY processor (refer to the Reality Programmer's Reference Manual), when this command is entered the system will respond with:

```
TO:
```

at which point the user may specify a new name for the list and a different account-name, if desired. The form of this response should be {name {account-name}}. The item-list may be directed to a file item by responding ({DICT} file-name) {item-id} as for the COPY verb. The same options that apply to the COPY verb apply to the COPY-LIST verb.

The general form of the DELETE-LIST verb is:

```
DELETE-LIST {name {account-name}}
```

This statement deletes a previously saved item-list. The frames used for the storage of the item-list are returned to the system overflow space. Users with SYS2 privileges can optionally specify "account-name" to delete an item-list that was saved by another user.

```
    SAVE-LIST {name}

    GET-LIST {name {account-name}}

    COPY-LIST {name {account-name}} {(options)}

    DELETE-LIST {name {account-name}}
```

Figure A. General Form of SAVE-LIST, GET-LIST, COPY-LIST, and DELETE-LIST Verbs

```
:SSELECT ACCOUNT WITH BILL-RATE > ".35" BY NAME <cr>

24 ITEMS SELECTED.
:SAVE-LIST OVER.35 <cr>

[241] 'OVER.35' CATALOGED; 1 FRAMES USED.


:GET-LIST OVER.35 <cr>

24 ITEMS SELECTED.
:ED ACCOUNT <cr>
11000
TOP
.  .
   .
   .


:COPY-LIST OVER.35 (D) <cr>
TO:(CONTROL-FILE) EXCESS.RATES <cr>

1 ITEM COPIED.


:DELETE-LIST OBSOLETE.PARTS GARY <cr>
[242] 'OBSOLETE.PARTS' DELETED.
```

Figure B. Sample Usage of SAVE-LIST, GET-LIST, COPY-LIST, and DELETE-LIST Verbs

3.9 EDIT-LIST and FORM-LIST Verbs

The EDIT-LIST verb permits editing an item-list saved by the SAVE-LIST verb. The FORM-LIST verb allows users to retrieve item-lists stored as items in Reality user files rather than the system POINTER-FILE.

EDIT-LIST allows selected lists to be changed, merged, and deleted via the Reality EDITOR. FORM-LIST allows a properly constructed item in a user file to be the source of selected item-ids. Their use is restricted to cases in which the item-id lengths add up to less than the maximum item size (32,267 bytes).

EDIT-LIST

The verb EDIT-LIST can be used to create, modify, merge, and delete item-lists saved via a SAVE-LIST statement. Its general form is:

```
    EDIT-LIST {name {account-name}} {(options)}
```

where "name" is the name of the saved list, and the optional "account-name" specifies the account under which the list was saved.

The normal system EDITOR is used to edit item-lists. All commands are the same. The only difference is that these item-lists are stored in the system POINTER-FILE instead of individual user files. Within the EDITOR, each line will correspond to a separate item-id. For exploded sorts, each line will have a second value which is the value number within the item. The line number corresponds to the order in which the list will be processed.

Note that an item-list cannot be edited if its size exceeds that of the user′s work space. The same options available to the EDITOR apply here also.

Figure B shows an example of the use of the EDIT-LIST verb.

FORM-LIST

The verb FORM-LIST functions like the GET-LIST verb except that an item in a Reality file is the source of the item-list instead of a list saved by SAVE-LIST. These lists may be formed with the EDITOR, DATA/BASIC, PROC, or by the COPY-LIST verb. Its general form is:

```
    FORM-LIST file-name item-id {(n)}
```

Each line of the item will specify an item-id in the list. A second value may appear on each line to specify a value number if the list is to correspond to an exploded sort. The optional "n" specifies that the list of item-ids formed should start with attribute (line) "n" instead of the entire item.

An example of the use of this capability is shown in Figure C. A DATA/BASIC program (see the appendix) derives sublists from a master SSELECT saved list, and files the sublists as unique items in a file. The FORM-LIST verb is then used to retrieve any sublist.

```
                    EDIT-LIST list-name {account-name}


                    FORM-LIST file-name item-id
```

Figure A. General Form of EDIT-LIST and FORM-LIST Verbs

```
:SSELECT PATIENT BY-EXP DATE <cr>

17 ITEMS SELECTED.
:SAVE-LIST PP <cr>

[214] 'PP' CATALOGED; 1 FRAMES USED.

:EDIT-LIST PP <cr>
TOP
.L22 <cr>
001 1022]3
002 1003]1
003 1003]2
.
.
.
015 1003]4
016 1022]2
017 3003]1
EOF 17
.I <cr>
017+3003<c>]5 <cr>
017+ <cr>
.FI <cr>
[241] 'PP' CATALOGED; 1 FRAMES USED.

:GET-LIST PP <cr>

18 ITEMS SELECTED
:
```

Figure B. Sample Usage of EDIT-LIST Verb

```
:SSELECT PARTS <cr>

5400 ITEMS SELECTED.
:SAVE-LIST PARTS-BY-ID <cr>

[241] PARTS-BY-ID CATALOGED; 54 FRAMES USED.

:GET-LIST PARTS-BY-ID <cr>

5400 ITEMS SELECTED.
:RUN BP SUBLIST <cr>
13 SUBLISTS GENERATED

:FORM-LIST SLIST SUBLIST1 <cr>

1633 ITEMS SELECTED.
:LIST PARTS HEADING "THIS REPORT INCLUDES PARTS IN SUBLIST1" <cr>
```

Figure C. Sample Usage of FORM-LIST Verb

3.10 T-DUMP, ST-DUMP, I-DUMP, S-DUMP and T-LOAD Verbs

T-DUMP and I-DUMP are ENGLISH verbs which allow the user to selectively dump his dictionaries and data files to the magnetic tape or to the terminal respectively. The T-LOAD verb allows the user to load files from magnetic tape.

T-DUMP, ST-DUMP, I-DUMP and S-DUMP

An ENGLISH sentence using the T-DUMP or I-DUMP verb is illustrated in Figure A. The T-DUMP verb dumps the selected items (from the selected file) to the magnetic tape. The DICT modifier causes dictionary data to be dumped, in which case file definition items (D/CODE=D) will not be dumped. An EOF mark is written to the tape after the dump. For detailed information regarding magnetic tape operations, refer to the Reality Programmer's Reference Manual. The following option, enclosed in parentheses, may appear in the T-DUMP statement:

T-DUMP option	Meaning
I	Specifies that item-ids will be listed as they are dumped
T	Inhibits tape label (see Reality Programmer's Reference Manual)

The I-DUMP operation is identical to the T-DUMP operation, except that the dump is made to the terminal. No options are used with I-DUMP. ST-DUMP and S-DUMP correspond to T-DUMP and I-DUMP except that sorting is done.

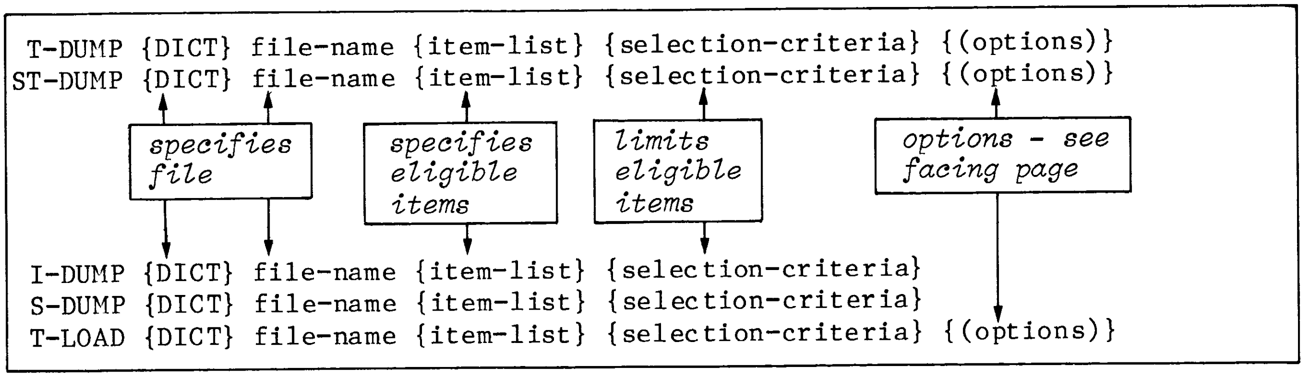
Figure B illustrates the use of the T-DUMP and I-DUMP verbs.

T-LOAD

An ENGLISH sentence usng the T-LOAD verb is illustrated in Figure A. The T-LOAD verb loads the specified file from magnetic tape. If selection-criteria are specified, attribute definitions will be retrieved from the dictionary of the file if loading a data file, or from the M/DICT if loading a dictionary (i.e., DICT was specified). The following options may appear in the T-LOAD statement:

T-LOAD option	Meaning
O	Overlay - The 'O' option will cause overlay of existing items with those from the tape if they have corresponding item-ids.
S	Suppress - Item-ids will be listed as they are loaded unless the 'S' option is used.

Multiple options are separated by commas. Figure C illustrates the use of the T-LOAD verb.



```
 T-DUMP {DICT} file-name {item-list} {selection-criteria} {(options)}
ST-DUMP {DICT} file-name {item-list} {selection-criteria} {(options)}
```

specifies file

specifies eligible items

limits eligible items

options - see facing page

```
 I-DUMP {DICT} file-name {item-list} {selection-criteria}
 S-DUMP {DICT} file-name {item-list} {selection-criteria}
 T-LOAD {DICT} file-name {item-list} {selection-criteria} {(options)}
```

Figure A. General Form of T-DUMP, ST-DUMP, I-DUMP, S-DUMP AND T-LOAD Verbs

```
:T-DUMP ACCOUNT > '23060' WITH CURR-BALNC <cr>

29 ITEMS DUMPED.
```

This sentence dumps to the magnetic tape all items in the ACCOUNT file which have items with item-ID's greater than '23060' as well as those with values for attribute CURR-BALNC.

```
:T-DUMP TEST-FILE <cr>

17 ITEMS  DUMPED.
```

This sentence dumps the entire TEST-FILE file to the magnetic tape.

```
:I-DUMP TEST '14' '15' '16' <cr>

14^THIS^IS^ITEM^14^111 222 333^AAA BBB CCC DDD^123456789^
15^THIS^IS^ITEM^15^ABCDEFGHIJK^.].].].].].^
16^THIS^IS^ITEM^16^1234 5678 9012 3456 7890^XXXXXX^
3 ITEMS DUMPED.
```

Figure B. Sample Usage of T-DUMP and I-DUMP Verbs

```
:T-LOAD INV WITH QTY < "10" <cr>
```

This sentence loads the INV file from the file positioned on the tape drive. The definition of QTY is found in the dictionary of the INV file.

```
:T-LOAD DICT ACCOUNT 'PAYMENTS' 'BILL-RATE' 'NAME' IF D/CODE = "A" (O,S) <cr>
```

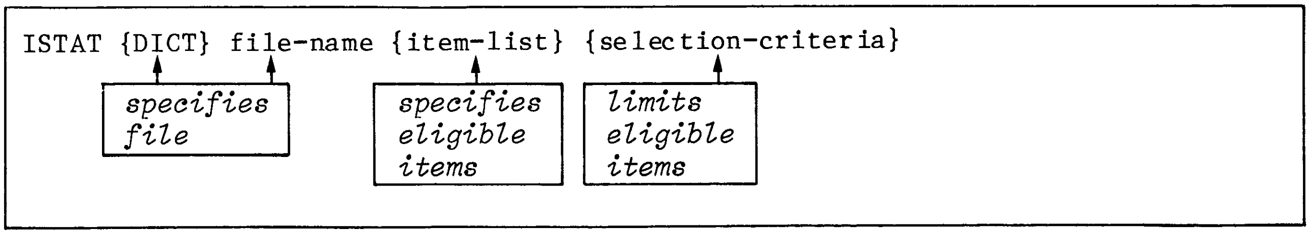
This sentence loads the dictionary section of the ACCOUNT file with items 'PAYMENTS' 'BILL-PAGE' 'NAME' if their D/CODE's are "A". The definition of D/Code is found in the M/DICT since DICT is specified.

Figure C. Sample Usage of T-LOAD Verb

3.11 ISTAT and HASH-TEST Verbs

The ISTAT verb provides a file hashing histogram showing item distribution with groups. HASH-TEST produces the same displays as ISTAT, but allows the user to select a different test modulo. HASH-TEST is useful in determining proper parameters for reallocating files.

An ENGLISH sentence usng the ISTAT verb is illustrated in Figure A. The ISTAT verb provides a file hashing histogram showing the distribution of items within groups in the file. For further information regarding file hashing, refer to the Reality Programmer's Reference Manual.



```
ISTAT {DICT} file-name {item-list} {selection-criteria}
```

Figure A. General Form of ENGLISH Sentence Using ISTAT Verb

```
:ISTAT ACCOUNT <cr>


FILE= ACCT MODULO= 3 SEPAR= 1                           11:48:05  12 FEB 1979
BYTES ITMS
05358 022 *>>>>>>>>>>>>>>>>>>>>>>
05373 022 *>>>>>>>>>>>>>>>>>>>>>>
05468 023 *>>>>>>>>>>>>>>>>>>>>>>>


ITEM COUNT=           67, BYTE COUNT=     16199, AVG. BYTES/ITEM=      241.7
AVG. ITEMS/GROUP=   22.3, STD. DEVIATION=    .5, AVG. BYTES/GROUP=    5399.7.


:ISTAT ACCOUNT > '23020' AND < '35090' <cr>


FILE= ACCT MODULO= 3 SEPAR= 1                           11:48:31  12 FEB 1979
BYTES ITMS
03089 012 *>>>>>>>>>>>>
02745 011 *>>>>>>>>>>>
02710 011 *>>>>>>>>>>>


ITEM COUNT=           34, BYTE COUNT=      8544, AVG. BYTES/ITEM=     251.2
AVG. ITEM/GROUP=    11.3, STD. DEVIATION=    .5, AVG. BYTES/GROUP=     2848.
```

Figure B. Sample Usage of ISTAT Verb

4 CORRELATIVES AND CONVERSIONS

4.1 Overview

Two types of special processing fields are available when using the ENGLISH processors. CORRELATIVE codes are used to define special processing interrelationships applied to attribute values as the values are retrieved from the file (prior to being sorted or used in a selection-criterion). CONVERSION codes are defined for attribute value just prior to output. The same conversions are also applied to values in the input line. Conversion codes are specified in Line 7 of attribute defining elements in the dictionary; correlative codes are specified in Line 8. This is exemplified in Figure C. In both cases, multiple codes may be specified, separated by a value mark (<c>], X'FD'); multiple codes are processed on a left-to-right basis.

In general, conversions are applied only prior to generating a value that is to be output. Values used for testing or other system purposes have only correlatives applied. This is defined in Figure A and may be illustrated:

```
stored    correlatives --->  intermediate  conversions --->  external
format                       format                          format
```

Processing codes are listed in Figure B. The same code may be specified either in Line 7 or Line 8 (i.e., as either a correlative or conversion) with the exception of D1 and D2 codes, which must be specified as correlatives. Since a correlative implies additional processing on sorts, selection, and in the determination of a control-break, processing codes for output should be specified as conversions wherever possible. Special processing can be effected by specifying correlative output processing. For example, a translate code is normally specified as a conversion, since it is used to convert internally stored vaues to an external format using a translation file. If an attribute with a T-conversion is used in a sort-key, or as a selection-criterion, the translation will not be applied. However, the T-code can be specified as a correlative to sort or select, using the translated value.

	Processing Stage	Correlatives Processed?	Conversions Processed?
1.	Output value, detail line of listing	yes	yes
2.	Output value, BREAK or TOTAL data line	no +	yes
3.	Value used for accumulation of a TOTAL	yes	no
4.	Value generated to check for a control-break or to test against print-limiters	yes	no
5.	Value generated for use in a sort-key	yes	no
6.	Value generated to test against for selection	yes	no
7.	Value specified by user in selection criteria	no	yes *

+ Break data line consists of totals, break field labels, and previously correlated break data values.

* In this case "input" conversion is done; in all other cases, "output" conversion is applied.

Figure A. Processing Code Effectivity

Name	Description
A	*ALGEBRAIC. Used to compute mathematical function.*
C	CONCATENATE. Used to concatenate values.
D	DATE. Used to convert dates.
D1	DEFINE PRIMARY. Used to define a primary associative attribute which is logically grouped with a set of secondary associative attributes (D2's). May be used as a correlative code only.
D2	*DEFINE SECONDARY. Used to define a secondary associative attribute. May be secondary associative attribute.*
F	*FUNCTION. Used to compute a mathematical function.on a defined set of attributes.*
G	*GROUP. Used to extract one or more contiguous segments from an attribute value.*
MD	*MASK DECIMAL. Used to convert and scale numbers.*
MP	*MASK PACKED. Used to convert packed decimal numbers.*
MT	*MASK TIME. Used to convert time.*
MX	*MASK HEXADECIMAL. Used to convert character strings to hexadecimal ASCII equivalents.*
T	*TEXT EXTRACTION. Used to extract a fixed number of characters from an attribute value.*
Tfile	*FILE TRANSLATION. Used to convert values by translating through a file.*
U	*USER-DEFINED. Used to evoke user-defined conversion.*
V	*SUBLIST CODE. Applies to DL/ID Line 8 only (see the topic SUBLIST: THE WITHIN CONNECTIVE).*

Figure B. Processing Code Summary

```
item 'INV.TIME' in DICT INV

001 A←————————————————Attribute Definition Item.
002 25←———————————————AMC (25th attribute).
003
004
005
006
007 MTHS←—————————————Output specification (MT Conversion).
008 G2*1←—————————————Internal specification (G correlative).
009 R←————————————————Right justified attribute.
010 10←———————————————Maximum Length.
```

Figure C. Sample Attribute Definition Item Containing Processing Codes

4.2 Defining Associative Attributes: D1 and D2

The D1 and D2 codes are used to identify primary and secondary associative attributes within the same item. D1 and D2 are specified as correlatives only.

The purpose of D1, D2 correlatives is to provide a facility whereby a set of attributes (the secondary D2s) can be logically grouped with a single master attribute (the primary D1). This type of relationship is useful in describing, for example, a list of purchase order numbers in a parts-file where the purchase order number is the D1 and the set of related attribute values (e.g., quantity-on-order, quantity-received, etc.) are D2s, with each D2 relating back to (and grouped with) the primary D1 value.

The general form of the D1 correlative is:

```
D1;amc{;amc}...
```

where:

D1 is the correlative code identifying a primary associative attribute.

amc is the numeric attribute mark count of each of the defined secondary associative attributes in the file; each amc specified in the D1 correlative must be numerically greater than the amc of the primary attribute itself.

; is a separator.

The general form of the D2 correlative is:

```
D2;amc
```

where:

D2 is the correlative code identifying a secondary associative attribute.

amc is a numeric attribute mark count of the defined primary associative attribute in the file.

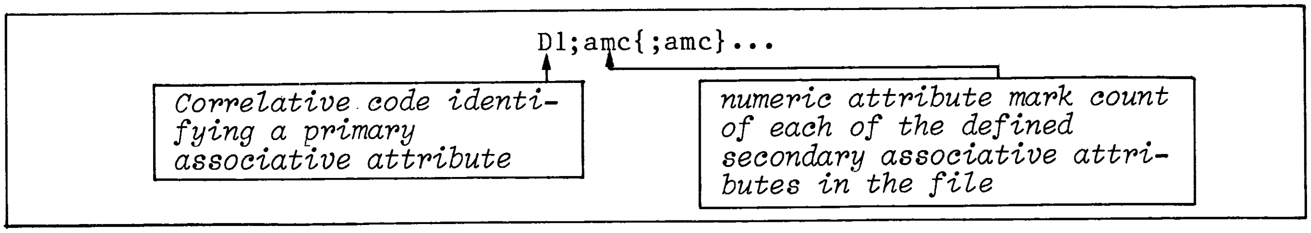
; is the separator.

Any D1 or D2 correlative must occur first in the correlative field.

The example in Figure C shows an ENGLISH output for a D1 attribute (DATE) and three associated D2 attributes (CODE, UNITS, and DOLLARS). The second ENGLISH output in this figure shows attribute definition items for these attributes. For further examples illustrating the use of D1 and D2 correlatives via ENGLISH, see the topic OUTPUT CRITERIA: MULTIVAUED ATTRIBUTE PRINT LIMITING.

The D1 attribute may have multivalues, each separated by a value mark (<c>], X'FD'). Each D2 attribute should have a corresponding number of multivalues; however, each of these multivalues may have multiple (secondary) value themselves. Each sub-multivalue (called a secondary value or subvalue) is separated by a secondary value mark (<c>\, X'FC').

D1 correlatives defined for attributes which also have F correlatives will be ignored. A print-limiter on the D1 attribute causes all corresponding D2 values to be suppressed.

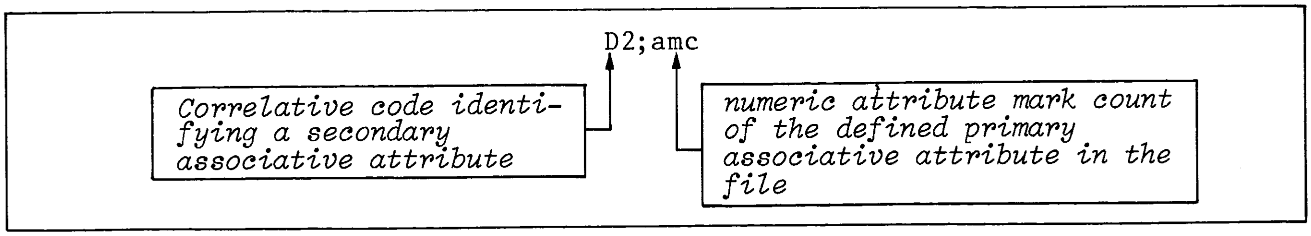


D1;amc{;amc}...

Correlative code identifying a primary associative attribute

numeric attribute mark count of each of the defined secondary associative attributes in the file

Figure A. General Form of D1 Correlative



D2;amc

Correlative code identifying a secondary associative attribute

numeric attribute mark count of the defined primary associative attribute in the file

Figure B. General Form of D2 Correlative

```
:LIST TEST-FILE '5330' DATE CODE UNITS DOLLARS <cr>

PAGE 1                                            18:15:24  12 FEB 1979

TEST-FILE. DATE....... CODE..... UNITS..... DOLLARS...
                       *         *          *

5330       07 APR 1978 P                         9.50
           18 MAR 1978 B                         9.50
           17 MAR 1978 T                         2.00
           13 MAR 1978 R               2721      7.50
           05 FEB 1978 P                         9.20
           15 JAN 1978 B                         9.20
           14 JAN 1978 T                         2.00
           10 JAN 1978 R               2696      7.20

END OF LIST

:LIST DICT TEST-FILE 'DATE' 'CODE' 'UNITS' 'DOLLARS' <cr>

PAGE 1                                            18:15:39  12 FEB 1979

TEST-FILE. D/CODE A/AMC S/NAME......... V/CONV.... V/CORR......... V/TYP V/MAX

DATE       S         20 DATE            D          D1;21;22;23     R        11
CODE       S         21 CODE                       D2;20           R         9
UNITS      S         22 UNITS                      D2;20           R        10
DOLLARS    S         23 DOLLARS         MD2        D2;20           R        10

END OF LIST
```

Figure C. Sample Use of D1, D2 Correlatives

4.3 Defining Group Extraction: G

The G code is used to select one or more contiguous segments of an attribute value for output.

One or more contiguous segments of an attribute value may be retrieved for output via use of the G code. The attribute value whose contiguous segment(s) is to be retrieved may consist of any number of segments, each separated by a nonnumeric character (except the minus sign or a system delimiter). This code functions somewhat like the FIELD function in DATA/BASIC.

The general form of the G code is:

```
G{m}*n
```

where:

G   is the code name

m   is the number of segments to skip; if omitted, zero is assumed and retrieval begins with the first segment.

*   is the nonnumeric character which is the segment separator (delimiter) in the attribute value (a system delimiter may not be used).

n   is the number of segments to be retrieved.

Figure A summarizes the general form of the G code. Figure B shows examples of the use of this code.

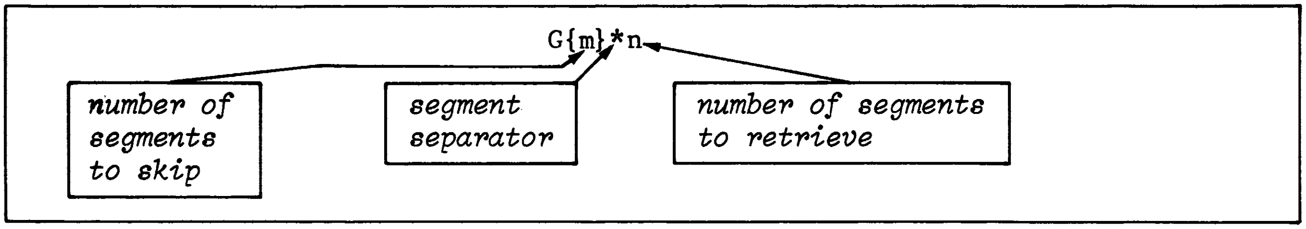


Figure A. General Form of G Code

Code	Attribute Value	Value Output
G$1	ABC$DEF$GHI$JKL	ABC
G1$2	ABC$DEF$GHI$JKL	DEF$GHI
G2$1	ABC$DEF$GHI$JKL	GHI
G1$1	ABC$DEF$GHI$JKL	DEF
G$2	ABC$DEF$GHI$JKL	ABC$DEF
G1A1	123A55555A22	55555
G2A1	123A55555A22	22

Figure B. Sample Usage of G Code

4.4 Defining Concatenation: C

The C code provides the facility to concatenate attributes and/or literal values prior to output.

The general form of the C code is:

```
    C{;}n{*n}...
```

where:

C   is the code name.

;   is optional and ignored.

*   is the character to be inserted between the concatenated attributes and/or literals. A semicolon (;) is a reserved character that means no separation character is to be used. Any nonnumeric (except a minus sign or system delimiter) is valid, including blank.

n   is any attribute mark count (AMC), or any literal enclosed in single quotes.

If the A/AMC (line two) of the attribute definition item containing the C code is non-zero, then a null value will be returned if that attribute contains a null (i.e., the concatenate code will be ignored). If the A/AMC is zero, then the concatenate will always be performed.

Figure A summarizes the general form of this code. Figure B gives an example (note the C conversion in item 'CAT' of file TEST).

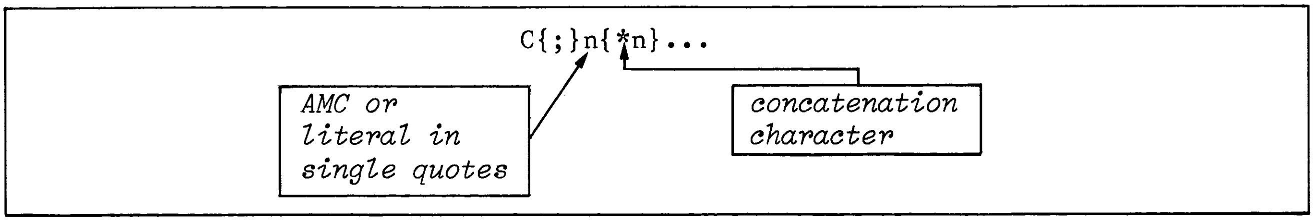


Figure A. General Form of C Code

```
Item 'CAT1'             Item 'CAT2'

001 A                   001 A
002 0                   002 99
003                     003
004                     004
005                     005
006                     006
007 C2;'55'=1/4         007 C2;'55'=1/4
008                     008
009 L                   009L
010 20                  010 20



Item '123'              Item '456'

001 ABC                 001 AAAA
002 DEF                 002
003                     003 BBBB
004 XZY                 004 CCCC


:LIST TEST '123' '456' CAT <cr>

PAGE 1                                     12:05:33  12 FEB 1979

TEST...... CAT1................ CAT2................

123        DEF55=ABC/XYZ        DEF55=ABC/XYZ
456        55=AAAA/CCCC

2 ITEMS LISTED.
```

Figure B. Sample Usage of C Code

4.5 Defining Text Extraction: T

The T code is used to extract a fixed number of characters from an attribute value.

A contiguous string of characters may be extracted from an attribute value using a T code. The general form of the T code is:

```
T{m,}n
```

where:

T is the code name.

m is the optional starting column number.

, is the separator (if omitted, the form Tn is assumed).

n is the number of characters to be retrieved.

If the form 'Tm,n' is used, then "n" characters starting from character "m" will be extracted. If the form 'Tn' is used, then "n" characters will be extracted beginning with the first character from left-to-right or right-to-left, depending upon whether type L or R (respectively) is specified in the dictionary attribute V/TYP (see the Reality Programmer's Reference Manual).

This code can be used to save space in file items by allowing an attribute (or item-id) to contain different fixed length values. For example, the two character state abbreviation and zip code can be concatenated together in one attribute in the form "ssnnnnn" rather than occupying two attributes. This example saves one byte per item and could result in significant space savings for large files (and decreased processing time due to smaller items).

Figure A summarizes the general form of the T code. Several examples are shown in Figure B (where V/TYP=L is assumed).

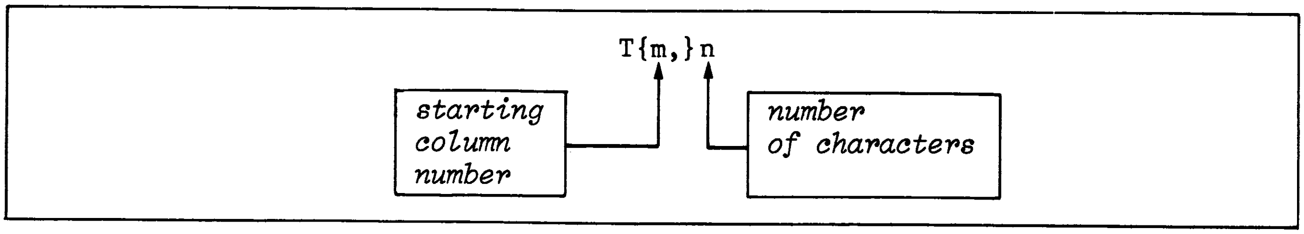


Figure A. General Form of T Code

Conversion	Attribute Value	Value Output
T3,2	ABCDEFG	CD
T3,5	ABCDEFG	CDEFG
T2	CA92631	CA
T3,5	CA92531	92631
T9	ABCDEFG	ABCDEFG
T8,1	65432XYZ	Z
T3,3	65432XYZ	432
T2,2	0123456789	12

Figure B. Sample Usage of T Code

4.6 Converting and Scaling Numbers: MD

The MD (mask decimal) code provides a facility for converting and scaling numbers to or from an internal format.

Numbers which contain decimal points, commas, and/or dollar signs may be converted and scaled to or from an internal signed integer format. Typically, numeric values are stored without decimal points, commas, or dollar signs to save space. The MD code allows values for the appropriate attribute to be entered in any form (with or without decimal point, commas, dollar signs, etc.) during input and it will convert them to the proper internal form. The MD code is almost always specified as a conversion (Line 7). The general form of the MD code is:

```
MDn{m}{Z}{,}{$}{i*}{c}
```

where:

MD is the code name.

n is a single numeric digit defining the number of digits to print following the decimal point. If n=0, the decimal point will not be output following the value.

m is an optional single numeric digit defining the "scaling factor" (as a power of 10), i.e., the number of implied decimal digits for the number on the file. If n<m, then the last digit will be rounded. If this parameter is omitted, m=n is assumed. However, if the "i*" option is used and "Z" or "," or "$" options are omitted, then "m" is required.

Z is an optional parameter specifying the suppression of leading zeros. A zero is always output preceeding the decimal point for values less than 1 and greater than -1.

, is an optional parameter for output which causes commas to be inserted between every thousandths position of the value.

$ is an optional parameter for output which causes a dollar sign to be appended preceding the converted output value.

i* is an optional parameter that causes the value to be overlaid on a field of "i" characters, "*" specifies the filler character and may be any nonnumeric (is typically an asterisk or a blank to cause dollar signs to align).

c is an optional parameter that is a credit indicator and may be one of the following:

- causes a minus sign to follow negative values; a blank to follow positive or zero values

C causes the letters 'CR' to follow negative values; two blanks to follow positive or zero values

< causes negative values to be enclosed with a "<...>" sequence; a blank follows positive or zero values

```
MDn{m}{Z}{,}{$}{i*}{c}
```

see text on facing page for meaning of parameters

Figure A. General Form of MD Code

MD Code	Stored Value	Converted Value
MD2	1234567	12,345.67
MD2	1234567	12,345.67
MD2,$	1234567	$12,345.67
MD2,$12*	1234567	$**12,345.67
MD2,$12*	0	$*******0.00
MD2,$12*	null	
MD23,	1234567	1,234.57
MD2,$12*	-1234567	$*-12,345.67
MD2,$12*-	-1234567	$*12,345.67-
MD2,$12*C	-1234567	$12,345.67CR
MD2Z$<	99999	$999.99b
MD2Z$<	-99999	$<999.99>
MD2Z,$12*C	1234567	$12,345.67bb
MD2Z,$12*C	0	
MD2Z,$12*C	null	
MD2,$12-	1234567	$12,345.67b
MD2,$12-	-1234	$12.34-
MD24,-	-1234567	$123.46-
MD2,$124#	1234567	$##12,345.67
MD0,	1234567	1,234,567

Figure B. Sample Usage of MD Code

4.7 Defining Date Format: D

The D code provides the facility for converting dates to or from a compact internal format suitable for arithmetic processing.

The general form of the D code is:

```
D{n}{*m}{s}
```

where:

D is the code name.

n is an optional single digit that specifies the number of digits to be printed in the year field on output conversion only ("n" must be 0, 1, 2, 3, or 4). If omitted, 4 is assumed, see the note below regarding dates in ENGLISH input sentences.

* Is an optional nonnumeric delimiter that specifies the delimiter of concatenated segments that will be skipped before the date portion of an attribute is retrieved. "*" Cannot be a system delimiter or ";".

m is a single digit that must accompany "*" (if "*" "*"s specified). Parameter 'm' is the number of concatenated segments to be skipped before the date portion of an attribute is retieved.

s is an optional nonnumeric character that is to be used as the separator between month, day, and year on output (the format being mm s dd s yyyy). A European date format will be printed as: dd s mm s yyyy, if the date format has been set to international style with the DATE-FORMAT verb (refer to the Reality Programmer's Reference Manual).

The internal date is defined as the number of days (plus or minus) from December 31, 1967. The following list illustrates the internal format:

Date	Internal Format
22 SEP 1967	-100
21 DEC 1967	-10
30 DEC 1967	-1
31 DEC 1967	0
01 JAN 1968	1
10 JAN 1968	10
09 APR 1968	100
26 SEP 1970	1000

The user should note that on input, if the year is not specified, then the current year as defined by the system will be used. If the year is input as two digits only (e.g., 29 or 73), then the twentieth century is used if the year is in the range 30 through 99 (inclusive), and the twenty-first century is used if the year is in the range 0 through 29 (inclusive). Also note that current date conversion routines handle dates between May 1, 1878 through September 30, 2057. Like the MD code, attributes defined by a D code may have the date specified in any format in the input sentence and it will be converted into internal form for comparison.

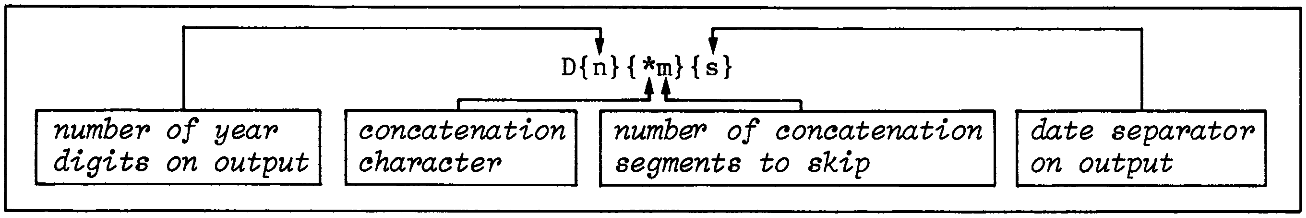

D{n}{*m}{s}
number of year digits on output
concatenation character
number of concatenation segments to skip
date separator on output

Figure A. General Form of D Code

D Code	Internal Value	Output Value
D	2704	27 MAY 1975
D/	2704	05/27/1975
D-	2707	05-27-1975
D0	2704	27 MAY
D0/	2704	05/27
D2*	2704	05*27*75
D	-13732	27 MAY 1930
D/	-13732	05/27/1930
D-	-13732	05-27-1930
D0/	19141	05/27
D2*	19141	05*27*30
D%1	ABC%2704	ABC%27 MAY 1975
D%1/	ABC%2704	ABC%05/27/1975
D%1-	ABC%2704	ABC%05-27-1975
D0%1	ABC%2704	ABC%27 MAY
D0	ABC%2704	ABC%2704

Figure B. Sample Usage of D Code

4.8 Defining Time Format: MT

The MT code provides a facility for converting an external time to or from an internal format suitable for arithmetic processing.

The internal time format is the number of seconds from midnight. The external time is 24-hour military format (e.g., 23:25:59) or 12-hour format (e.g., 11:25:59PM). The general form of the MT code is:

```
MT{H}{S}
```

where:

MT is the code name.

H is optional and specifies 12-hour external format. If omitted, 24-hour military format is assumed.

S is optional and specifies the appending of seconds. If omitted, seconds are not used.

When codes MTH or MTHS are used, 12-hour external format is specified. For input conversion, then, the time is entered with AM or PM immediately following the numeric time (AM is optional); on output, AM or PM is always printed immediately following the numeric time.

The user should note that 12:00AM is considered midnight, and 12:00PM is considered noon. AM and PM will be ignored on input if code MT or MTS is specified. Illegal values are converted to null on input. Negative values will be output as a null value, while other illegal values will convert to "00:00".

Figure B illustrates use of the MT conversion.

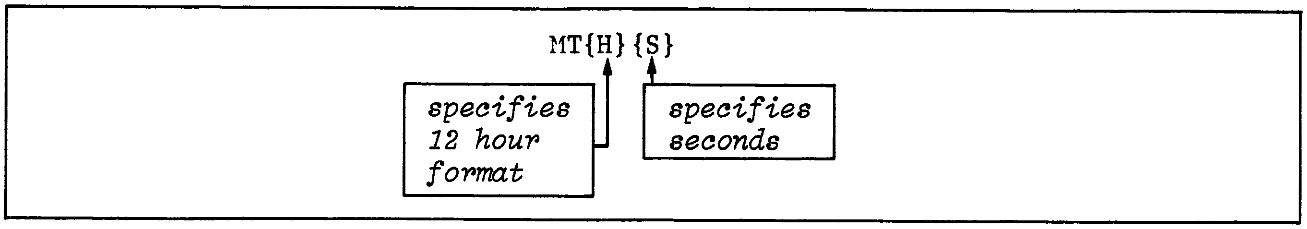


Figure A. General Form of MT Code

MT Code	Input Value		Stored Value	Output Value
MT	12		43200	12:00
MTH	12		0	12:00AM
MTS	12		43200	12:00:00
MTHS	12		0	12:00:00AM
MT	12:15AM	*	44100	12:15
MTH	12:15AM		900	12:15AM
MT	1		3600	01:00
MTH	1		3600	01:00AM
MT	6AM	*	21600	06:00
MTH	6AM		21600	06:00AM
MT	1PM	*	3600	01:00
MTH	1PM		46800	01:00PM
MT	13		46800	13:00
MTH	13		46800	01:00PM
MT	XYZ		null	blank
			ZYZ	00:00

* = AM or PM notation on input is ignored by the system

Figure B. Sample Usage of MT Code

4.9 Defining File Translation: Tfile

> The Tfile code provides a facility to convert a value by translating through a file. This facility allows ENGLISH to access more than one file at a time.

The value to be translated, specified by the A/AMC (Line 2), is used as an item-id for retrieving an item from the defined translation file. The input value is then converted by replacing it with a defined attribute-value from the translation item. The format for the Tfile code is:

```
T{*}file;c;input-amc;output-amc
```

where:

T is the code name.

file is the file-name through which the translation takes place. It may be preceded by a single asterisk character (*) to indicate a dictionary.

; is the separator.

c is the translate subcode, which must be one of the following:

V - Conversion item must exist on file, and specified attribute must have value for conversion.

C - If conversion item does not exist, or if specified attribute has no value, then use original value; otherwise perform conversion.

I - Input verify only; functions as a V for input and a C for output.

O - Output verify only; functions as a V for output and a C for input.

X - If conversion item does not exist, or if specified attribute has no value, then use the null value; otherwise perform conversion.

input-amc is the attribute mark count in the translation file for input translation. After locating the translation item usng the input value as the item-id, the attribute-value for this attribute, if any, will replace (convert) the original value. If this parameter is null, no input translation takes place.

output-amc is the attribute mark count in the translation file for output translation. Functions similarly to input-amc but is invoked for output translation. If this parameter is null, no output translation takes place.

Figure A summarizes the general form of the Tfile code. An example is shown in Figure B (note the Tfile conversion in item 'NAME' in the dictionary of the DETAIL file).

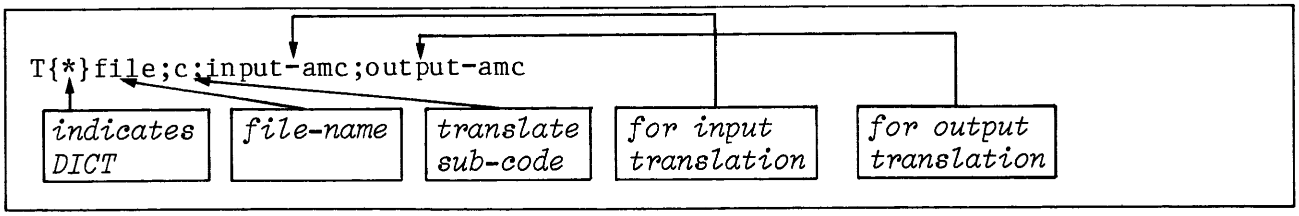


Figure A. General Form of Tfile Code

Dictionary Section of DETAIL file:

```
Item 'NAME'

001 A
002 3
004                 Specifies that the third attribute of each item
005                 will be used as the item-ID for translation
006
007 TMASTER;C;1;1   Specifies comparison against and retrieval
008                 from first attribute in translation file
009 1               'MASTER'.
010 10
```

Data Section of DETAIL file:

```
Item 'I1'          Item 'I2'          Item 'I3'

001 400            001 480            001 350
002 ABC            002 80             002 XYZ
003 1234           003 1235           003 1237
```

Data Section of MASTER FILE:

```
Item '1234'        Item '1235'        Item '1237'

001 SMITH          001 BROWN          001 JONES
002 JOHN           002 JOE            002 MARY
003 XYZ            003 ABC            003 1234
```

ENGLISH sentence:

```
:LIST DETAIL 'I1' 'I2' 'I3' NAME <cr>

PAGE 1                                      11:08:37  12 FEB 1979

DETAIL.... NAME......
I1         SMITH
I2         BROWN
I3         JONES

3 ITEMS LISTED.
```

Figure B. Sample Usage of Tfile Code

4.10 Defining ASCII, Packed Decimal, & User Conversion: MX, MP, & U

The MX code is used to convert any string of characters stored on file to or from its corresponding hexadecimal ASCII equivalent. The MP code is used to convert a value to or from its packed decimal representation. The U code permits a user defined special purpose subroutine to be invoked for special conversion.

MX Code

Using the MX code, any character string on file may be converted to or from its hexadecimal ASCII equivalent. One byte on the file will be converted to two hexadecimal digits. The general form of the MX code is:

MX

Figure C illustrates the use of the MX code.

MP Code

The MP code allows decimal numbers to be packed for storing. Packed decimal numbers occupy approximately half the disc storage space required by unpacked decimal numbers. The general form of the MP code is:

MP

On input, the MP conversion combines pairs of 8-bit ASCII digits into single packed 8-bit digits by stripping off the high-order four bits of each ASCII digit and storing the low-order four bits into successive halves of the stored bytes. Leading '+' signs are ignored. Leading '-' signs cause a 4-bit code, 'D' expressed in hexadecimal, to be stored as the upper half of the first internal digit. If there are an odd number of packed halves, a leading four bits of '0' are added. The range of data bytes in internal format (expressed in hexadecimal) is '00' through '99' and 'D0' through 'D9'. Only valid decimal digits (0-9) and sign (+,-) should be input; other characters cause no conversion to take place.

Packed decimal digits should always be unpacked for output, i.e., the MP code should be specified for both input and output of the data. Packed values that are output unconverted do not display on terminals in a recognizable format. Also, many of these characters are recognized by terminals as control characters.

Figure D presents examples of MP code conversion.

U Code

A user-defined special purpose subroutine (in assembly code) may be invoked for special conversion via the U code. The general form of the U code is:

Unxxx

where:

U is the code name.

n is the entry point.

xxx is the Mode-id (refer to the Reality Assembly Language Programming Manual).

At the point where conversion normally occurs for both input and output, the user-program is entered with the value to be converted in a work area. For the exact nature of the programming interface, consult the conversion subroutine in the Reality Assembly Language Programming Manual.

MX

MP

Unxxx

Mode-ID

Figure A. General Form of MX, MP, and U Codes

MX Code

Stored Value	Converted Value
ABC	414243
ABC#	41424323
T	54
%T	2554
XYZ	58595A
....	2E2E2E2E

Figure B. Sample Usage of MX Code

MP Code

Decimal Value	Byte Length	Packed Value (Hexadecimal)	Byte Length
99	(2)	99	(1)
-3	(2)	D3	(1)
98762	(5)	098762	(3)
+723	(4)	0723	(2)

Figure C. Sample Usage of MP Code

4.11 Defining Mathematical Functions: F

The F code is used to compute a value by performing indicated mathematic and logic operations on one or more operands. The operands may be constants, attribute values, or codes for certain system parameters such as date and time. Operand values are stored in a seven-entry pushdown stack designated STACK1 (top of stack), STACK2, ..., STACK 7.

The general form of the F code is:

```
F{n};element{;element}...
```

An "element" may be of any of the following:

1. A numeric AMC specifying an attribute value to be pushed onto the stack, optionally followed by an "R" (Repeat code), optionally followed by any conversion specification(s) enclosed in parentheses

2. A constant of the form Cn where "n" is a numeric or string constant to be pushed onto the stack

3. A D which specifies the current date is to be pushed onto the stack

4. A T which specifies the current time is to be pushed onto the stack

5. A special two-character operand designating a particular system counter

6. An operator which specifies an operation to be performed on the top two entries in the stack

The operands (items 1 through 5 above) always cause a single push onto the stack, with existing values (if any) moved down one position in the stack. The operands are listed in Figure A. Operand specification is further described in the topics F CODE STACK and F CODE SPECIAL OPERANDS. The operators are listed in Figure B. The relational operators compare STACK2 to STACK1; after the operation, STACK1 will contain either a 1 or 0, depending upon whether the result is true or false, respectively (e.g., if the F code were F;C3;C3;= then STACK1 would contain a 1).

The optional precision specification "n" is described in the topic F CODE SPECIAL OPERANDS.

Since the F code does not rely on the A/AMC (Line 2), Line 2 is usually specified as a dummy attribute (99, for example).

The F code has been largely replaced by the easier to use and more flexible algebraic function code 'A' described later in this chapter.

amc{R}{(conversion)}	*Numeric AMC, optional repeat code, optional conversion specification(s).*
Cn	*Numeric or string constant.*
D	*Code for system date.*
T	*Code for system time.*
Nx	*Two-letter codes for system counters.*

Figure A. F Code Operands

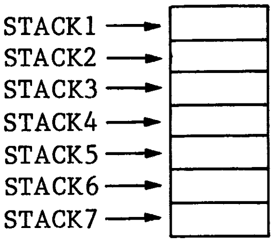
Operator	Operation
*	*Multiplication of the top two entries in the stack.*
/	*Division of STACK1 by STACK2.*
R	
+	*Addition of the top two entries in the stack*
-	*Subtraction of STACK2 from STACK1.*
S	*A total sum of all STACK1 multi-values is placed at the top of the stack.*
"	*Duplication of STACK1 pushed onto the stack.*
_	*Exchanges top two positions in stack.*
^	*Pop stack.*
:	*Concatenation of STACK1 with STACK2.*
[]	*Substring of STACK3. STACK2 specifies starting column, and STACK1 specifies number of characters.*
=	*"Equal" relational operator.*
<	*"Less than" relational operator.*
>	*"Greater than" relational operator.*
#	*"Not equal" relational operator.*
[	*"Equal to or less than" relational operator.*
]	*"Equal to or greater than" relational operator.*

Figure B. F Code Operators

4.12 F Code Stack

A pushdown stack is used to perform F code operations.

Arithmetic operations specified by an F code operate on the top two entries in a pushdown stack. This pushdown stack has a maximum capacity of seven entries, and may be visualized as follows:



STACK1 is the top position in the stack, STACK2 is the next position, etc. As a value is pushed onto the stack, it is pushed into position STACK1; the original value of STACK1 is pushed down to STACK2 and so on. As a value is fetched off the stack, it is popped from position STACK1; the original value of STACK2 moves up to STACK1; and so on. No more than seven consecutive pushes or pops can occur.

The F code comprises any number of operands or operators in reverse Polish format, separated by semicolons. When the function processor encounters an operand specification (e.g., a numeric attribute mark count or constant), it "pushes" the corresponding value onto the top of the stack (STACK1). When the function processor encounters an arithmetic operator, it performs the corresponding operation on the top two entries in the stack (STACK1 and STACK2). When the entire F code has been computed, the top entry in the stack (STACK1) will be the value retrieved.

As a notation sample, the operation "(1+2)*4=12" would be done with an F code thus:

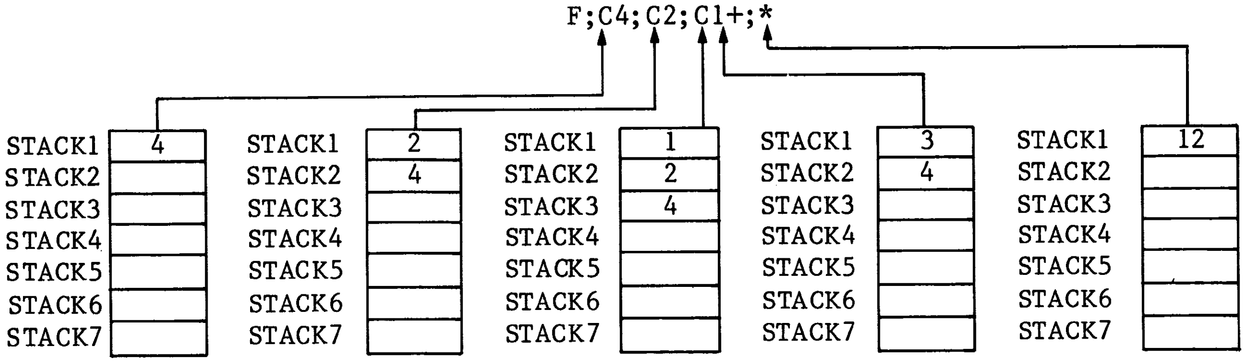
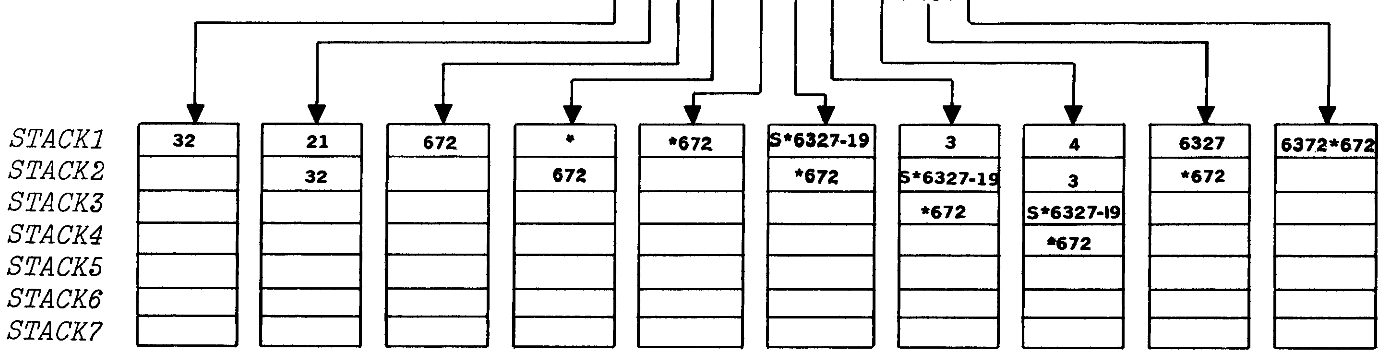


Figure A exemplifies an F code. This figure shows contents of the stack as each element is processed.

```
item '1137' in PARTS file
001 S*6327-19
002 32
003 21
```

```
item 'CODE' in DICT PARTS file
001 A
00222
:
.
008 F;2;3;*;C*;:;1;C3;C4;[];:
009 L
010 10
011 1
```

F code: F;2;3;*;C*;:;1:C3;C4;[];:



```
:LIST PARTS '1137' CODE

PAGE 1                                    08:31:41  03 MAR 1977
PARTS.....  CODE......
1137        6327*672

1 ITEM LISTED.
```

Figure A. Sample F Code and Associated Operations on Stack

4.13 F Code Special Operands

F code operands may be multivalued, may contain conversion specification(s), or may be a special two-character operand specifying one of several counters.

Attribute operands may be multivalued. When arithmetic operations are performed on two multivalued lists (vectors), the answer will also be multivalued and will have as many values as the longer of the two lists. Zeros will be substituted for the null values in the shorter list. For example, suppose the attribute with AMC=10 had a value of "5]10]15" and Attribute 15 had values "20]30]40]50". If the F correlative F;10;15;+ were processed, the result in STACK1 would be "25]40]55]50". If a single valued attribute is to be repetitively added (or subtracted, etc.) with a multivalued attribute, then the single letter R should immediately follow the AMC in the F code (e.g., F;10;25R;+).

Any conversion may be specified in the body of a function correlative. The conversion specification(s) must immediately follow the "operand" specification in the F correlative, and must be enclosed by parentheses. Multiple conversions may be specified by separating the individual conversion specifications by value marks (<c>], X'FD'). Examples are given in Figure B.

Special two-character operands may be used as F code elements for various system counters, as listed in Figure C. For example:

```
F;ND;3;/
```

On every detail line, this returns the value from the third attribute; on every break line (including the grand-total line), the average value of data in Attribute 3 is returned.

The optional parameter "n" following 'F' may be used to specify the number of fractional digits (0 to 4) to be retained during calculations when using a mixture of whole numbers and numbers with implied fractional digits when an MD conversion is specified in the body of a function. For example, suppose Attribute 1 contains 15934 (which is normally interpreted as 1.5934 using an MD4 code) and Attribute 2 contains 37. The function code F;1(MD4);2;+ yields 38 (fractional digits are dropped). This can be overcome by coding the function as F4;1(MD4);2;+ which returns 38.5934. Alternately, the function could be written as F;1;2;C10000;*;+]MD4 to scale the whole number 37 to 370000, adding 15934, then converting to 38.5934 using the MD4 code. The latter method was required before the precision specification was available.

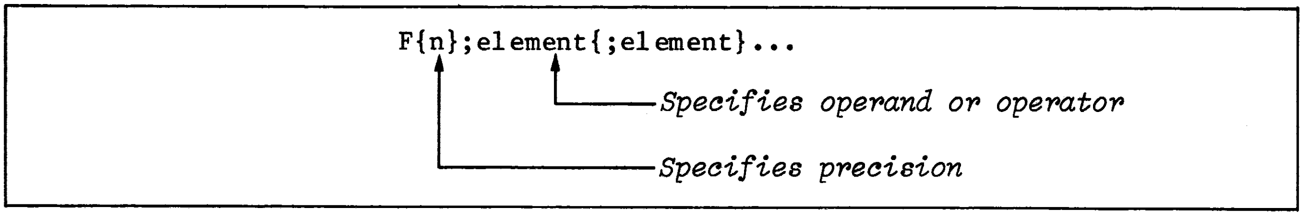


Figure A. General Form of F Code

F;10;11(T*SALES;X;3;3);*	*Places the data from attribute #10 into the stack; picks up ID from attribute #11, translates it from dictionary of file 'SALES' and places it into the stack; and multiplies the two values to give the result.*
F;1(U11FO);2(U21FO);+	*Calls user conversion routines to operate on data from attribute #1 & #2, then adds values.*
F;D(D2/]G2/1);3(D2]G2/]);-	*Computes the difference in the "year" fields of the system date and the date stored in attribute #3. The "D2/" converts the date from internal format to "MM/DD/YYYY"; the "G2/1" then isolates the "year" section of the date. The " " is actually a value-mark (shift-control-M).*

Figure A. Sample Usage of F Correlatives with Conversions

Operand	Description
NI	*Current item counter.*
ND	*Number of detail lines since last BREAK on a Break data line; has a value of 1 on any detail lines, and is equal to the item counter (i.e., total items) on a grand-total line. This operand is used to get averages, etc., within the control-break structure.*
NV	*Current multi-value counter (columnar listing only).*
NS	*Current sub-multi-value counter (columnar listing only).*

Figure B. F Code Counter Operands

4.14 Summary of F Code Stack Operations

This topic summarizes F code stack operations. The notation STACK1 -> STACK2 means that the contents of STACK1 (the top of the stack) is pushed down to position STACK2.

Element	Description	Action
amc {(conversion)}	attribute (with conversion)	Push corresponding attribute value, after optional conversion, onto pushdown stack (maximum seven levels): attribute value -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
Cn	constant	Push numeric or string constant "n" onto stack: "n" -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
D	date	Push numeric value representing current system date (internal form) onto stack: date -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 lost
+	add	STACK1 + STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
-	subtract	STACK1 - STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
*	multiply	STACK1 * STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
/	divide	STACK1 / STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
R	remainder	remainder(STACK1/STACK2) -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
S	sum	summation(STACK1) -> STACK1 Prior to this operation, STACK1 may be multivalued; this operator sums all those multivalues into a single value
"	duplicate	STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
_	exchange	STACK1 <—> STACK2
^	POP	STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2 -> STACK1 -> lost
:	concatenate	STACK1:STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
[]	extraction	STACK3[STACK2,STACK1] -> STACK1 (STACK2 = starting position, STACK1 = length), STACK4 -> STACK2, STACK5 -> STACK3, STACK6 -> STACK4, STACK7 -> STACK5

Element	Description	Action
T	time	Push numeric value representing current system time (internal format) onto stack: time -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
NI	item counter	Push numeric value representing current item counter onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
ND	detail line counter	Push numeric value representing number of detail lines since the last control-break onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
NV	multivalue counter	Push numeric value representing current multi-value counter onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
NS	sub-multivalue counter	Push numeric value representing current sub-multivalue counter onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
=	equal	1) If STACK1 = STACK2 then 1 -> STACK1 2) If STACK1 # STACK2 then 0 -> STACK1
#	not equal	1) If STACK1 # STACK2 then 1 -> STACK1 2) If STACK1 = STACK2 then 0 -> STACK1
<	less than	1) If STACK1 < STACK2 then 1 -> STACK1 2) If STACK1 not < STACK2 then 0 -> STACK1
>	greater than	1) If STACK1 > STACK2 then 1 -> STACK1 2) If STACK1 not > STACK2 then 0 -> STACK1
[	less than or equal to	1) If STACK1 <= STACK2 then 1 -> STACK1 2) If STACK1 not <= STACK2 then 0 -> STACK1
]	greater than or equal to	1) If STACK1 >= STACK2 then 1 -> STACK1 2) If STACK1 not >= STACK2 then 0 -> STACK1

3) In each case STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2

```
F{n};element{;element}...
```

Figure A. General Form of F Code

4.15 Defining Mathematical Functions: A Code Operands

The A code is designed to function similarly to the F code and replaces it with a simpler-to-write and easier-to-understand format.

The algebraic function code, unlike the F code, uses an algebraic notation rather than Reverse Polish Notation. It allows the use of dictionary names and is recursive in that it will apply functions stored in other attribute definitions. Parentheses may be used to indicate the order of operations.

The general form of the A code is:

```
    A{n};expression
```

where "expression" is made up of operators, operands, and special functions as described below and in the following topic. Expressions are formed exactly like DATA/BASIC expressions. The precision specification "n" functions identically to that in the F code and is described in the topic F CODE SPECIAL OPERANDS. Note: The A/AMC (Line 2) of attribute definitions containing A or F codes should be zero (not a dummy number) if they will be be referenced by name in other functions.

Operands

AMC Numbers

An attribute mark count (AMC) is specified by putting the number in the A code, just as in F codes. An AMC of zero indicates the item-id 9999 is the item size, and 9998 supplies a sequential item count. A conversion may be specified following the AMC enclosed in parentheses as in the F code.

Attribute Names

An attribute name may be used instead of an AMC. The dictionary name is used as an argument to the N function whose format is:

```
    N(attribute-name)
```

The operation of the N function is as follows. The name is looked up in the dictionary and an error message is printed if it is not found. The A/AMC (Attribute 2) is used as an AMC in the A code. Any correlatives existing in Attribute 8 of that dictionary item will be applied to the value obtained (including other A or F codes; A/AMC (Line 2) must be zero in such referenced attributes). This allows the A code to use functions defined in other dictionary items which may in turn be derived from other functions in other dictionary items, etc. This recursive capability is a major advancement over the now obsolete F code.

Literals

Any literal string or a numeric constant may be specified by enclosing the value in double quotes. The most common mistake in writing A codes is to omit the quotes around constants which would then imply an AMC as described above.

Special Operands

The A code also supports the special operands NI, NV, NS, ND, D, and T available for the F code as described in previous topics.

Any operand (AMC, N(attribute-name), or special operand) may be followed by "R" to specify that a single value should be repeated so that there will be the same number of values as a multivalued attribute used elsewhere in the calculation. They may also be preceded by a minus sign to change the sign of the value, if desired.

```
A{n};expression
```

Figure A. General Form of the A Code

A;N(COST)	*Retrieves the value defined in the attribute definition item COST in the dictionary of the file being used. If any correlatives are present on line 8 of COST, these will be applied before returning the value. If an A Code on line 8 of COST references other dictionary items, the recursive feature will be applied.*
A;5	*References attribute five.*
A;"HELLO"	*Forms the string "HELLO".*
A;"365"	*Specifies the numeric constant 365.*
A;NV	*Accesses the multivalue counter.*
A;T	*Accesses the current system time (internal form).*

Figure B. Sample Usage of A Code Operators

4.16 Defining Mathematical Functions: A Code Operators and Functions

The A code is designed to function similarly to the F code and replaces it with a simpler to write and easier to understand format.

Operators

Arithmetic Operators

The operators +, -, *, and / are available and take two operands (see previous topic) and return the sum, difference, product or quotient. It is important to note that division returns an integer result just as in F codes.

String Operators

The : operator specifies concatenation of the results of two expressions.

Relational Operators

The operators <, >, >=, <=, =, and # denote logical relational operations and take two expressions as operands and evaluates to 1 (true) or 0 (false) as in the F code.

Precedence

The precedence of operators is important to keep in mind when writing A codes. Infinite levels of parenthesis are allowed in A codes to specify the order of operations. In the absence of parenthesis, multiplication and division have greater precedence over addition and subtraction, which in turn have greater precedence than the relational operators. If two operators have the same precedence, they are applied from left to right. For example, 1*2+3<4 will evaluate as ((1*2)+3)<4 and 4/5*6 will evaluate as (4/5)*6.

Special Functions

Remainder Function: R

The remainder function takes two expressions as operands and returns the remainder of the first divided by the second. The format of the R function is:

```
R(expression,expression)
```

For example A;R(N(COST),"5") returns the remainder when COST is divided by 5.

Summation Function: S

The summation function takes one expression as an operand and works the same way as in the F code. The sum of all multivalues will be returned. The "S" function format is:

```
S(expression)
```

Substring Function: [n,n]

A substring may be specified by using square brackets just as in DATA/BASIC and may be specified after any expression or function. For example: N(NAME)["2","5"] will return a string of five characters starting at Position 2. The most common mistake here is the use of AMCs in the starting position and length specifications instead of the intended literals in double quotes. N(NAME)[2,5] would retrieve a substring from NAME where the starting position is found in Attribute 2 and a length found in Attribute 5.

```
A;expression
```

Figure A. General Form of A Code

A;N(PRICE)*N(QTY)	*Multiplies the value of the attribute defined by PRICE by the value QTY.*
A;N(COST)/("10"*N(DISCOUNT))	*Divides the value of COST by 10 times the value of DISCOUNT.*
A;N(AMTDUE)-S(N(PAYMENTS))	*Subtracts the sum of the multivalued payments from AMTDUE.*
A;5["1","2"]:" ":N(ZIP)	*Concatenates the first two characters of attribute five with a blank followed by the value from the attribute defined by ZIP.*
A;"5"+N(AMT)/10	*Divides AMT by attribute 10 (not the value "10") and then adds five.*
A;S(N(RATE)R*N(HOURS))	*Multiplies multivalued HOURS by single valued RATE (repeating), then sums the multivalued results.*

Figure B. Sample Usage of A Code

4.17 Processing Conversions/Correlatives: Detail Lines & Sorting

Attribute values may be modified through the use of correlatives and conversions applied at various stages of processing. The use of correlatives and conversions for detail line output and sorting is detailed in this topic.

Detail Lines

Each line of output that is not a total or subtotal line is called a detail line. Output on detail lines is printed after applying both the correlative and conversion fields. The arrangement below illustrates the application of correlatives and conversions to attribute values.

Internal format	Correlative processing (if present)	Intermediate format	Conversion processing (if present)	Detail line value

Figure A presents three attribute definition items which show how detail line output is derived. Attribute definition item 'INTERNAL' is used to list the value of Attribute 1 as it is sorted in the 'TEST' file. 'INTERMEDIATE' lists the attribute after the correlative field has been processed. 'CITY' continues the processng by applying a translate conversion to the intermediate result. The translate items for this conversion reside in the dictionary of the 'TEST' file. Figure B is the output listing of the 'TEST' file showing all three attribute definition items being applied to Attribute 1 of the file.

Sorting

Sorts are performed on the intermediate attribute values. Figure C shows a sort by the translated field 'CITY'. Note that the listing is not in order by the printed value of 'CITY' but rather is in order by the intermediate values listed under 'INTERMEDIATE'.

If a sort by city instead of by the intermediate code were necessary, then the translate code could be combined with the correlative "T3". The correlative field would then be "T3]T*TEST;V;2;1". The conversion field would be null. However, this organization should be avoided when possible because performing the translate at this point will slow many common file operations which do not use the conversion field and which thus would not otherwise require the translation to be performed. Codes such as D, MT, and MD should usually be specified as conversions (Line 7) since the outcome of a sort would be the same as specifing as a correlative.

```
TEST......... D/CODE.... A/AMC V/CONV.......... V/CORR... V/TYP V/MAX


INTERNAL      A          1                                L     10
INTERMEDIATE  A          1                      T3        L     10
CITY          A          1     T*TEST;V;2;1     T3        L     10

ABC           COVINA
POI           COVINA
POP           SOUTHBROCK
SDF           SOUTHBROCK
WER           SOUTHBROCK
ZZZ           COVINA
```

Figure A. Dictionary of TEST File

```
:LIST TEST INTERNAL INTERMEDIATE CITY <cr>

PAGE 1                                             11:57:46  12 FEB 1979

TEST...... INTERNAL.. INTERMEDIATE CITY......

JAN        ABC123     ABC          COVINA            After T3
FEB        ABC789     ABC          COVINA            correlative
MAR        WER77      WER          SOUTHBROCK
APR        SDF765     SDF          SOUTHBROCK
JUN        ZZZ765     ZZZ          COVINA            After T3
JUL        POP333     POP          SOUTHBROCK        correlative
AUG        WER772     WER          SOUTHBROCK        and T*TEST;
SEP        ABC133     ABC          COVINA            V;2;1 conversion
OCT        ABC122     ABC          COVINA
NOV        POI331     POI          COVINA
DEC        WER733     WER          SOUTHBROCK
MAY        WER89      WER          SOUTHBROCK
```

Figure B. Detail line Output with Correlatives and Conversions

```
:SORT TEST BY CITY INTERNAL INTERMEDIATE CITY <cr>

PAGE 1                                             12:03:34  12 FEB 1979

TEST...... INTERNAL.. INTERMEDIATE CITY......

FEB        ABC789     ABC          COVINA
JAN        ABC123     ABC          COVINA
OCT        ABC122     ABC          COVINA
SEP        ABC133     ABC          COVINA
NOV        POI331     POI          COVINA            Note that the sort
JUL        POP333     POP          SOUTHBROCK        is applied to the
APR        SDF765     SDF          SOUTHBROCK        intermediate values.
AUG        WER772     WER          SOUTHBROCK
DEC        WER733     WER          SOUTHBROCK
MAR        WER77      WER          SOUTHBROCK
MAY        WER89      WER          SOUTHBROCK
JUN        ZZZ765     ZZZ          COVINA
```

Figure C. A SORT Using Correlativ and Conversions

4.18 Processing Conversions/Correlatives: Selection & Control-Breaks

This topic details the use of correlatives and conversions for selection processing and for control-breaks.

Selection Processing

Selection processing compares the file values (with correlatives applied), to the values entered in the input line after input conversions have been processed.

When a selection value is specified in an input line, an input conversion (if applicable) is performed on that value before it is compared with the intermediate values extracted from the file. Note that this input conversion may be different from the output conversion (as in a translate code). Figure A shows additional translate items in the dictionary of the TEST file which allow the input conversion for the listing shown in Figure B. The input translation conversion is applied to the value "SOUTHBROCK" to yield selection value "POP". (Note that the input-AMC in the translate code is used). The selection value "POP" is compared, in turn, against each of the intermediate values (as listed under column INTERMEDIATE).

In Figure B, not all detail lines with value "SOUTHBROCK" are selected. This is because of the lack of symmetry between the input-AMC and the output-AMC in the translate code. On output, values "POP", "SDF", and "WER" convert into "SOUTHBROCK", but on input, "SOUTHBROCK" converts into "POP" only.

To select all items which print as "SOUTHBROCK", it will be necessary to perform the translate as a correlative instead of as a conversion. This will cost processing time, however, since the translation will be applied to all items in the file.

Control-Breaks

Control-breaks apply to the intermediate values generated by processing all correlative codes. Control-breaks are signalled by changes in these intermediate values. Figure A shows the dictionary used for the listing in Figure C. Note that in Figure C, the control-breaks apply to the intermediate value.

To break on the printed values of CITY, the translate code would have to be specified as a correlative instead of as a conversion. As with selection processing, this will cost processing time, since the translation will be applied to all items in the file.

```
TEST.......  D/CODE.... A/AMC V/CONV........... V/CORR.... V/TYP V/MAX

INTERNAL     A          1                                  L     10
INTERMEDIATE A          1                       T3         L     10
CITY         A          1     T*TEST;V;2;1      T3         L     10

ABC          COVINA
POI          COVINA
POP          SOUTHBROCK
SDF          SOUTHBROCK
WER          SOUTHBROCK
ZZZ          COVINA
SOUTHBROCK              POP
COVINA                  ABC
```

Figure A. Dictionary of TEST File (Extended)

```
:LIST TEST WITH CITY = "SOUTHBROCK" INTERNAL INTERMEDIATE CITY SUPP <cr>

TEST....... INTERNAL.. INTERMEDIATE CITY......

JUL         POP333     POP          SOUTHBROCK
```

Figure B. Selection Processing with Correlatives and Conversions

```
:SORT TEST BY CITY INTERNAL INTERMEDIATE BREAK-ON CITY SUPP <cr>

TEST....... INTERNAL.. INTERMEDIATE CITY......

FEB         ABC789     ABC          COVINA
JAN         ABC123     ABC          COVINA
OCT         ABC122     ABC          COVINA
SEP         ABC133     ABC          COVINA
                                    ***
NOV         POI331     POI          COVINA
                                    ***
JUL         POP333     POP          SOUTHBROCK
                                    ***
APR         SDF765     SDF          SOUTHBROCK
                                    ***
AUG         WER772     WER          SOUTHBROCK
DEC         WER733     WER          SOUTHBROCK
MAR         WER77      WER          SOUTHBROCK
MAY         WER89      WER          SOUTHBROCK
                                    ***
JUN         ZZZ765     ZZZ          COVINA
```

Figure C. Control-Break Processing with Correlatives and Conversions

4.19 Processing Conversions/Correlatives: Totals & Subtotals

This section details the use of correlatives and conversions for totals and subtotals.

TOTALS AND SUBTOTALS

Correlatives, if present, are applied to attribute values before they are used to accumulate totals and subtotals. Before the total or subtotal is printed, however, it is converted using the conversion field (such as with an MD conversion to position the decimal point). The method of processing totals for attributes with function (F or A) codes is detailed below.

The function code operates in two different fashions on an attribute with a TOTAL modifier, depending on whether it is specified as a correlative (line 8) or as a conversion (line 7).

Function Correlatives

As a correlative (the usual method), the function is applied to the detail lines printed (along with conversions) and these functioned values are accumulated for the subtotal and total. Therefore, a total of the functioned values displayed on the detail lines is computed. At total time, conversions are applied to this total of functioned values (such as an MD conversion) and the results are printed.

Function Conversions

As a conversion, the function is applied to the detail lines printed (along with conversions); however, these functioned values are not accumulated for the subtotal or total. Instead, at total time, the function will be applied to the totals that have been calculated for the attributes whose AMC's (or names in A codes) appear in the Function code. Therefore, the function of other totaled values is obtained. This is particularly useful in calculating averages (one of the values might be the special 'ND' operand).

The user must total (with the TOTAL modifier) those attributes whose AMC's appear in the Function code; otherwise an unpredictable total will result. Since the total displayed in this manner is not a total of the values listed on the detail lines above it, but is instead a function of other total values, the user may wish to suppress the detail listing for this attribute. This can be accomplished by assigning this attribute a dummy AMC (99, for example).

Figure A presents an example of the processing difference between function correlatives and function conversions.

```
    VALUE             :LIST INV QTY PRICE TOTAL VALUE <cr>
001 S
002 1                 INV....... QTY.. PRICE.. VALUE.....
003
004                   1001          12   $3.54     $42.48
005                   1007          34   $1.23     $41.82
006                   1004         113   $0.23     $25.99
007 MD2$              1002          15   $1.00     $15.00
008 F;1;2;*
009 R                 ***                         $125.29 -- Total of above
010 10                                                    "functioned" values

    VALUE             :LIST INV QTY PRICE TOTAL VALUE <cr>
001 S
002 1                 INV....... QTY.. PRICE.. VALUE.....
003
004                   1001          12   $3.54     $42.48
005                   1007          34   $1.23     $41.82
006                   1004         113   $0.23     $25.99
007 F;1;2;*]MD2$      1002          15   $1.00     $15.00
008
009 R                 ***                           $1.74 -- Unpredictable
010 10                                                    result since
                                                          QTY (AMC=1) and
                                                          PRICE (AMC=2) were
                                                          not totaled

    VALUE             :LIST INV TOTAL QTY TOTAL PRICE TOTAL VALUE <cr>
001 S
002 1                 INV....... QTY.. PRICE.. VALUE.....
003
004                   1001          12   $3.54     $42.48
005                   1007          34   $1.23     $41.82
006                   1004         113   $0.23     $25.99
007 F;1;2;*]MD2$      1002          15   $1.00     $15.00
008
009 R                 ***          174   $6.00   $1044.00 -- Function of
010 10                                                    totaled values
                                                          QTY and PRICE.
                                                          Not total of VALUE
                                                          figures above

    AVERAGE           :LIST INV QTY TOTAL PRICE TOTAL AVERAGE <cr>
001 S
002 99                INV....... QTY.. PRICE.. AVERAGE...
003 AVERAGE]PRICE                              PRICE
004                   1001          12   $3.54
005                   1007          34   $1.23    Detail suppressed with
006                   1004         113   $0.23    non-existant AMC=99
007 F;ND;2;/]MD2$     1002          15   $1.00
008
009 R                 ***                $6.00      $1.50 -- Function of
010 10                                                    totaled value
                                                          of PRICE
```

Figure A. Sample Uses of TOTAL with Function Correlatives and Function Conversions

APPENDIX A. ASCII CHARACTER SET

Decimal	Hex	EBCDIC Equivalent	ASCII Character	Prism Display	Prism Key	Special Use in Reality
0	00	00	NUL	None	<c>@	Delay char, sort key delimiter
1	01	01	SOH	None	<c>A	Prism home command
2	02	02	STX	None	<c>B	
3	03	03	ETX	None	<c>C	End of text
4	04	37	EOT	None	<c>D	
5	05	2D	ENQ	None	<c>E	
6	06	2E	ACK	None	<c>F	Cursor forward on Prism
7	07	2F	BEL	None	<c>G	Bell of Prism
8	08	16	BS	None	<c>H	Backspace on Prism
9	09	05	HT	None	<c>I	Tab
10	0A	25	LF	None	<c>J	Cursor down on Prism
11	0B	0B	VT	None	<c>K	Vertical address on Prism
12	0C	0C	FF	None	<c>L	Screen erase on Prism
13	0D	0D	CR	None	<c>M	Carriage return
14	0E	0E	SO	None	<c>N	
15	0F	0F	SI	None	<c>O	
16	10	10	DLE	None	<c>P	Horizontal address on Prism blank compression character
17	11	11	DC1	None	<c>Q	
18	12	12	DC2	None	<c>R	Retype entire line. Enable slave printer
19	13	3A	DC3	None	<c>S	Dump Prism screen to slave printer (option)
20	14	3C	DC4	None	<c>T	Disable slave printer
21	15	3D	NAK	None	<c>U	Cursor back on Prism
22	16	32	SYN	None	<c>V	
23	17	26	ETB	None	<c>W	
24	18	18	CAN	None	<c>X	Cancel line
25	19	19	EM	None	<c>Y	
26	1A	3F	SUB	None	<c>Z	Cursor up on Prism
27	1B	27	ESC	[	ESC, <c>[	EDITOR command delimiter. Invokes SCREEN PROCESSOR command-mode
28	1C	1C	FS	None		
29	1D	1D	GS	None		
30	1E	1E	RS	None		
31	1F	1F	US	None		
32	20	40	blank		space	
33	21	5A	!	!	<cs>A, !	
34	22	7F	"	"	<cs>B, "	String delimiter in ENGLISH and BASIC
35	23	7B	#	#	<cs>C, #	
36	24	5B	$	$	<cs>D, $	
37	25	6C	%	%	<cs>E, %	
38	26	50	&	&	<cs>F, &	
39	27	7D	’	’	<cs>G, ’	String delimiter in ENGLISH and BASIC
40	28	4D	(	(	<cs>H, (	

APPENDIX A. ASCII CHARACTER SET (Continued)

Decimal	Hex	EBCDIC Equivalent	ASCII Character	Prism Display	Prism Key	Special Use in Reality
41	29	5D	)	)	<cs>I,)	
42	2A	5C	*	*	<cs>J, *	
43	2B	4E	+	+	+	
44	2C	6B	,	,	,	
45	2D	60	-	-	-	
46	2E	4B	.	.	.	
47	2F	61	/	/	/	
48	30	F0	0	0	0	
49	31	F1	1	1	<cs>Q, 1	
50	32	F2	2	2	<cs>R, 2	
51	33	F3	3	3	<cs>S, 3	
52	34	F4	4	4	<cs>T, 4	
53	35	F5	5	5	<cs>U, 5	
54	36	F6	6	6	<cs>V, 6	
55	37	F7	7	7	<cs>W, 7	
56	38	F8	8	8	<cs>X, 8	
57	39	F9	9	9	<cs>Y, 9	
58	3A	7A	:	:	<cs>Z, :	
59	3B	5E	;	;	;	
60	3C	4C	<	<	<	
61	3D	7E	=	=	=	
62	3E	6E	>	>	>	
63	3F	6F	?	?	?	
64	40	7C	@	@	@	
65	41	C1	A	A	A	
66	42	C2	B	B	B	
67	43	C3	C	C	C	
68	44	C4	D	D	D	
69	45	C5	E	E	E	
70	46	C6	F	F	F	
71	47	C7	G	G	G	
72	48	C8	H	H	H	
73	49	C9	I	I	I	
74	4A	D1	J	J	J	
75	4B	D2	K	K	K	
76	4C	D3	L	L	L	
77	4D	D4	M	M	M	
78	4E	D5	N	N	N	
79	4F	D6	O	O	O	
80	50	D7	P	P	P	
81	51	D8	Q	Q	Q	
82	52	D9	R	R	R	
83	53	E2	S	S	S	
84	54	E3	T	T	T	
85	55	E4	U	U	U	
86	56	E5	V	V	V	
87	57	E6	W	W	W	
88	58	E7	X	X	X	
89	59	E8	Y	Y	Y	

Decimal	Hex	EBCDIC Equivalent	ASCII Character	Prism Display	Prism Key	Special Use in Reality
90	5A	E9	Z	Z	Z	
91	5B	80	[	[	[	String search delimiter
92	5C	E0	\	\	\	RUNOFF control character
93	5D	90	]	]	]	ENGLISH string search delimiter
94	5E	5F	^	^	^	ENGLISH string search delimiter
95	5F	6D	_	_	_	RUNOFF control character
96	60	79	`	@	<cs>0	
97	61	81	a	A	<c>!, <s>A	
98	62	82	b	B	<c>", <s>B	
99	63	83	c	C	<c>#, <s>C	
100	64	84	d	D	<c>$, <s>D	
101	65	85	e	E	<c>%, <s>E	
102	66	86	f	F	<c>&, <s>F	
103	67	87	g	G	<c>', <s>G	
104	68	88	h	H	<c>(, <s>H	
105	69	89	i	I	<c>), <s>I	
106	6A	91	j	J	<c>*, <s>J	
107	6B	92	k	K	<c>+	
108	6C	93	l	L	<c>,	
109	6D	94	m	M	<c>-	
110	6E	95	n	N	<c>.	
111	6F	96	o	O	<c>/	
112	70	97	p	P	<c>0	
113	71	98	q	Q	<c>1, <s>Q	
114	72	99	r	R	<c>2, <s>R	
115	73	A2	s	S	<c>3, <s>S	
116	74	A3	t	T	<c>4, <s>T	
117	75	A4	u	U	<c>5, <s>U	
118	76	A5	v	V	<c>6, <s>V	
119	77	A6	w	W	<c>7, <s>W	
120	78	A7	x	X	<c>8, <s>X	
121	79	A8	y	Y	<c>9, <s>Y	
122	7A	A9	z	Z	<c>:, <s>Z	
123	7B	C0	{	[	<c>;	
124	7C	6A	|	\	<c><	
125	7D	D0	}	]	<c>=	
126	7E	A1	~	^	<c>>	Sort key delimiter
127	7F	07	DEL	None		
128	80	04		None		
129	81	06		None		
130	82	08		None		
131	83	09		None		
132	84	0A		None		
133	85	13		None		

Decimal	Hex	EBCDIC Equivalent	ASCII Character	Prism Display	Prism Key	Special Use in Reality
134	86	14		None		
135	87	15		None		
136	88	17		None		
137	89	1A		None		
138	8A	1B		None		
139	8B	20		None		
140	8C	21		None		
141	8D	22		None		
142	8E	23		None		
143	8F	24		None		
144	90	28		None		
145	91	29		None		
146	92	2A		None		
147	93	2B		None		
148	94	2C		None		
149	95	30		None		
150	96	31		None		
151	97	33		None		
152	98	34		None		
153	99	35		None		
154	9A	36		None		
155	9B	38		None		
156	9C	39		None		
157	9D	3B		None		
158	9E	3E		None		
159	9F	41		None		
160	A0	42		None		
161	A1	43		None		
162	A2	44		None		
163	A3	45		None		
164	A4	46		None		
165	A5	47		None		
166	A6	48		None		
167	A7	49		None		
168	A8	4A		None		
169	A9	4F		None		
170	AA	51		None		
171	AB	52		None		
172	AC	53		None		
173	AD	54		None		
174	AE	55		None		
175	AF	56		None		

Decimal	Hex	EBCDIC Equivalent	ASCII Character	Prism Display	Prism Key	Special Use in Reality
176	B0	57		None		
177	B1	58		None		
178	B2	59		None		
179	B3	62		None		
180	B4	63		None		
181	B5	64		None		
182	B6	65		None		
183	B7	66		None		
184	B8	67		None		
185	B9	68		None		
186	BA	69		None		
187	BB	70		None		
188	BC	71		None		
189	BD	72		None		
190	BE	73		None		
191	BF	74	blank	None		
192	C0	75	@	@		
193	C1	76	A	A		
194	C2	77	B	B		
195	C3	78	C	C		
196	C4	8A	D	D		
197	C5	8B	E	E		
198	C6	8C	F	F		
199	C7	8D	G	G		
200	C8	8E	H	H		
201	C9	8F	I	I		
202	CA	9A	J	J		
203	CB	9B	K	K		
204	CC	9C	L	L		
205	CD	9D	M	M		
206	CE	9E	N	N		
207	CF	9F	O	O		
208	D0	A0	P	P		
209	D1	AA	Q	Q		
210	D2	AB	R	R		
211	D3	AC	S	S		
212	D4	AD	T	T		
213	D5	AE	U	U		
214	D6	AF	V	V		
215	D7	B0	W	W		
216	D8	B1	X	X		
217	D9	B2	Y	Y		
218	DA	B3	Z	Z		

Decimal	Hex	EBCDIC Equivalent	ASCII Character	Prism Display	Prism Key	Special Use in Reality
219	DB	B4	[	[		
220	DC	B5	\	\		
221	DD	B6	]	]		
222	DE	B7	^	^		
223	DF	B8				
224	E0	B9	$\overline{@}$	$\overline{@}$		
225	E1	BA	A	A		
226	E2	BB	B	B		
227	E3	BC	C	C		
228	E4	BD	D	D		
229	E5	BE	E	E		
230	E6	BF	F	F		
231	E7	CA	G	G		
232	E8	CB	H	H		
233	E9	CC	I	I		
234	EA	CD	J	J		
235	EB	CE	K	K		
236	EC	CF	L	L		
237	ED	DA	M	M		
238	EE	DB	N	N		
239	EF	DC	O	O		
240	F0	DD	P	P		
241	F1	DE	Q	Q		
242	F2	DF	R	R		
243	F3	E1	S	S		
244	F4	EA	T	T		
245	F5	EB	U	U		
246	F6	EC	V	V		
247	F7	ED	W	W		
248	F8	EE	X	X		
249	F9	EF	Y	Y		
250	FA	FA	Z	Z		System Delimiters:
251	FB	FB	[	[		Start Buffer (SB)
252	FC	FC	\	\	<c>\	Subvalue Mark (SVM)
253	FD	FD	]	]	<c>]	Value Mark (VM)
254	FE	FE	^	^	<c>^	Attribute Mark (AM)
255	FF	FF	_	_	<c>_	Segment Mark (SM)

NOTE: The <SHIFT> key may have to be depressed on some terminals if the desired character appears on the top half of a key top. For example, to generate an attribute mark (X'FE') on some terminals, it is necessary to depress and hold the <SHIFT> key to generate the "^" character, while depressing the <CTRL> key to cause the character to be a control character.

APPENDIX B. ACCOUNT FILE USED IN EXAMPLES

As an aid to understanding the numerous examples in this manual of the sample file named ACCOUNT, the following SORT output is provided. This output lists the values of key attributes for all the items in the ACCOUNT file.

```
:SORT ACCOUNT NAME ADDRESS BILL-RATE CURR-BALNC <cr>

PAGE 1                                                13:55:30  12 FEB 1979

ACCOUNT... NAME................ ADDRESS............ BILL-.. CURR-BALANCE...
                                                    RATE

11000      M H KEENER          100 ANCHOR PL          10.03 $      2,246.78
11015      L K HARMAN          118 ANCHOR PL           0.03 $          8.60
11020      J T O'BRIEN         124 ANCHOL PL           0.30 $    306,755.54
11025      P R BAGLEY          130 ANCHOL PL          10.03 $     82.045.33
11030      F E CABRON          101 BEGONIA            10.03 $         20.50
11035      R S MARCUS          107 BEGONIA            10.03 $     23,911.14
11040      E G MCCARTHY        113 BEGONIA             0.30 $        334.56
11045      F R DRESCH          119 BEGONIA            10.03 $      1,119.46
11050      J R MARSHECK        125 BEGONIA             0.30
11055      W H KOONS           131 BEGONIA            10.03 $    958,343.75
11060      F T NATORI          131 BAY STREET         33.33 $         34.86
11065      C V RANDALL         125 BAY STREET         10.03 $        552.13
11070      A A ALTHOFF         119 BAY STREET         10.03 $         22.60
11075      T F LINDSEY         113 BAY STREET         10.03 $         13.10
11080      E M AWAD            107 BAY STREET         10.03 $      2,937.35
11085      A B SEGUR           101 BAY STREET          0.30 $        224.55-
11090      J W JENKINS         130 AVOCADO             0.30 $      2,224.84
11095      J B STEINER         124 AVOCADO             0.30 $          3.83
11100      E F CHALMERS        118 AVOCADO             0.40 $         17.50
11105      C C GREEN           112 AVOCADO             0.30
11110      D L WEISBROD        106 AVOCADO             0.30 $        484.84
11115      D R MASTERS         100 AVOCADO             0.30 $          9.20
21780      E W AWAD            107 BAY STREET         10.03 $      9,932.22
23000      H T LEE             200 BAY STREET          0.35 $     12,332.11
23005      W B THOMPSON        206 BAY STREET         10.03 $     11,265.21
23010      W E MCCOY           212 BAY STREET          0.35 $     28,464.64
23015      R M COOPER          218 BAY STREET          0.35 $        385.56
23030      S L                 224 BAY STREET         10.03 $        983.34
           UNGERLEIDER
23025      D C BINGAMAN        230 BAY STREET          0.08 $         18.70
23030      L J DEVOS           201 CARNATION           0.30 $        484.94
23035      G A BORDEN          207 CARNATION           0.35 $      9,663.72
23040      P B SCIPMA          213 CARNATION           0.35 $    123,423.22
23045      P F KUGEL           219 CARNATION           0.35 $     99,422.34
23050      E G MCCARTHYJR      225 CARNATION           0.35 $         44.88
23055      S M NEWMAN          231 CARNATION           0.35 $      7,452.92
23060      S I SZABO           231 COVE STREET         0.35 $     54,668.13
23065      J A WOSK            225 COVE STREET         0.35 $      8,563.36
23070      L R MARCHANT        219 COVE STREET         0.01 $        345.12
23075      M J SADOSKI         213 COVE STREET         0.35 $      1,124.75
23080      J W YOUNG           207 COVE STREET        10.03 $         89.32
```

APPENDIX B. ACCOUNT FILE USED IN EXAMPLES (Continued)

```
PAGE 2                                           13:55:59  12 FEB 1979

ACCOUNT... NAME................ ADDRESS............ BILL-.. CURR-BALANCE...
                                                    RATE

23090      W J                  230 BEGONIA            0.35 $       20.45
           HIRSCHFIELD
23095      W E ZUMSTEIN         224 BEGONIA            0.01
23100      G J PACE             218 BEGONIA           10.03 $    9,562.24
23105      B C PAUL             212 BEGONIA            0.23 $    1,123.47
23110      J L VANGOTHEN        206 BEGONIA            0.35 $   47,452.93
23115      T F PIATKOSKI        200 BEGONIA            0.35 $   45,678.22
35000      J L DIESEM           300 COVE STREET        0.35 $      112.37
35005      J S ROWE             306 COVE STREET        0.35 $      464.72-
35010      S R KURTZ            312 COVE STREET       10.03 $      467.33
35015      W F GRUNBAUM         318 COVE STREET        0.35 $       88.47
35025      J D GOETZINGER       330 COVE STREET        0.35 $        3.45
35030      F M HUGO             301 DAHLIA             0.35 $      123.48
35035      M J                  307 DAHLIA             0.35 $      445.89
           LANZENDORPHER
35040      C E ESCOBAR          313 DAHLIA             0.35 $   88,822.12-
35050      P J WATT             325 DAHLIA            10.03 $      337.18
35055      J W ROMEY            331 DAHLIA             0.35 $   33,478.95
35060      J A SCHWARTA         331 DOCK WAY           0.02 $   33,822.34
35065      L J RUFFINE          325 DOCK WAY          10.03 $      558.43
35070      F R SANBORN          319 DOCK WAY           0.35 $   22,144.67
35075      J L CUNNINGHAM       313 DOCK WAY           0.40 $        7.70
35080      G A BUCKLES          307 DOCK WAY           0.35 $  447,765.48
35085      J F SITAR            301 DOCK WAY           0.02 $      200.00
35090      D U WILDE            330 CARNATION               $      884.54
35095      A W FEVERSTEIN       324 CARNATION          0.35 $       19.25
35100      R W FORSTROM         318 CARNATION         10.03
35105      S J FRYCKI           312 CARNATION          0.35 $    5,569.53
35110      H E KAPLOWITZ        306 CARNATION         10.03 $   94,944.55

67 ITEMS LISTED.
```

APPENDIX C. SUBLIST PROGRAM FOR EDIT-LIST AND FORM-LIST EXAMPLE

```
SUBLIST
*
*
*THIS PROGRAM TAKES A SELECT LIST ON A FILE WITH ID'S OF THE
*
*FORM SSSS*NNNNNNN...(WHERE SSSS IS A MAJOR DIVISION OF ID'S)
*
*AND GENERATES A SEPARATE SUBLIST FOR EACH VAUE OF SSSS.
*
*TO RUN THIS PROGRAM, ENTER :GET-LIST XXXX
*                           :RUN BASIC-PROGRAMS SUBLIST
*
*
*
OPEN '', 'SLIST' TO SLIST ELSE PRINT ' NO SLIST FILE';STOP
EQU AM TO CHAR(254)
LISTNO=0;*    THIS IS THE NUMBER OF THE CURRENT SUBLIST
FIRSTIME=1;*  IDENTIFY FIRST PASS THROUGH LOOP
CURRENTLIST=''
DONEFLAG=0;*  THIS FLAG INDICATES THE EXHAUSTION OF THE SELECTED STRING
*
*    LOOP THROUGH THE SELECTED LIST AND BUILD THE NEXT SUBLIST
*
*    LISTID IS THE PART OF THE ID WHICH IDENTIFIES THE SUBLIST
*           TO WHICH A VARIABLE BELONGS.
*
*    CURRENTLIST IS THE ID IDENTIFYING THE CURRENT SUBLIST.
*
2 READNEXT ID ELSE DONEFLAG=1;GOTO 5
LISTID=FIELD(ID,'*',1)
 IF FIRSTIME=1 OR CURRENTLIST # LISTID THEN
      CURRENTLIST=LISTID
*
*
*THE END OF 1 SUBLIST HAS BEEN FOUND.  WRITE THE ITEM OUT TO
*
*ITEM 'SUBLIST*N' IN FILE SLIST.  THIS ITEM WILL BE ACCESSABLE
*
*VIA THE VERB FORM-LIST.  THEN INITIALIZE FOR THE NEXT SUBLIST.
*
     5 IF FIRSTIME = 1 THEN FIRSTIME=0;GOTO 6
     WRITE SUBLIST ON SLIST,'SUBLIST':LISTNO
     IF DONEFLAG THEN STOP
     6 LISTNO=LISTNO+1
     SUBLIST=''
     END
SUBLIST=SUBLIST:ID:AM
GOTO 2
END
```

(THIS PAGE LEFT BLANK INTENTIONALLY)

INDEX

Fold Here and Staple

Do Not Tear—Fold Here

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1972 SANTA ANA, CA

POSTAGE WILL BE PAID BY ADDRESSEE:

MICRODATA CORPORATION
Attn: Marketing Services
P.O. Box 19501
Irvine, CA 92713

Microdata Corporation
17481 Red Hill Avenue, Irvine, California 92714
Post Office Box 19501, Irvine, California 92713
Telephone: 714/540-6730 · TWX: 910-595-1764