REALITY™ by Microdata

English

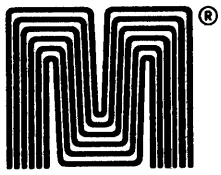
REFERENCE MANUAL

$6.00

REALITY™

ENGLISH REFERENCE MANUAL

May 1976



Printed in U.S.A.
98810 76 1008A

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

TABLE OF CONTENTS

```
:SORT THE ACCOUNT FILE WITH ITEMS BEFORE '23020' NAME ADDRESS (CR)

PAGE 1                                        14:43:38   20 AUG 1976

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

1 INTRODUCTION

1.1 THE ENGLISH LANGUAGE

ENGLISH is a user oriented data retrieval language used 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 relatively free-form sentence containing appropriate verbs, file names, data selection criteria, and control modifiers. Each user's vocabulary can be individually tailored to his particular 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, inter-file retrieval operations, display format, and more.

ENGLISH encompasses the following features:

- Limited freedom of word order and syntax for inquiries.
- 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 $2^{47}-1$).

It is assumed that the user has read the following manual prior to referencing this manual.

- Introduction To Reality

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 (non-shaded text).

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 (i.e., file-name, attribute-name, 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.*

Figure A. Conventions Used in General ENGLISH Formats

CONVENTION	MEANING
TEXT	*Shaded text represents the user's input.*
TEXT	*All other text represents output printed by ENGLISH.*
(CR)	*This symbol represents a carriage return entered by the user.*
o^CS	*This symbol represents a control-shift-O (i.e., a line continuation character) entered by the user. For further information regarding special control characters, refer to the appendix of this manual.*

Figure B. Conventions Used in ENGLISH Examples

1.2 HOW TO USE THIS MANUAL

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

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

The advantage of this format will become readily apparent to the reader as he 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 in knowing that, when he has completed reading a topic and goes to turn the page, he is done with one idea and ready to encounter a new one.

Subsequent sections of this manual describe various elements of the ENGLISH language. In presenting general formats and examples, certain conventions apply. Conventions used in general language formats are listed in Figure A, while conventions used in examples are listed in Figure B.

As an aid to understanding the numerous examples in this manual which deal with the sample file named ACCOUNT, the user may wish to periodically refer to the appendix of this manual, where an ENGLISH output is provided listing key attribute values for all the items in the ACCOUNT file.

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-ID's 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.

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 special word into an M/DICT item with the desired synonym name as the item-ID (refer to the Reality Programmer's Manual).

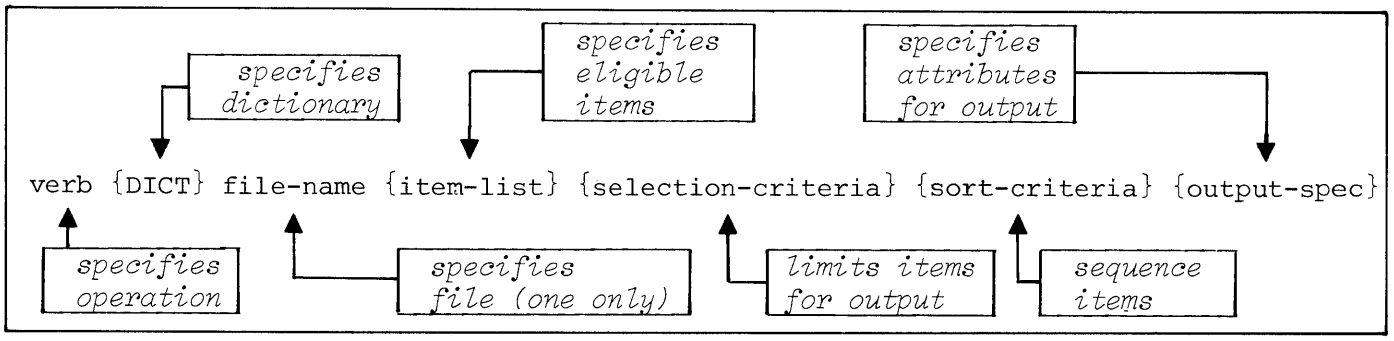


Figure A. General Form of ENGLISH Input Sentence

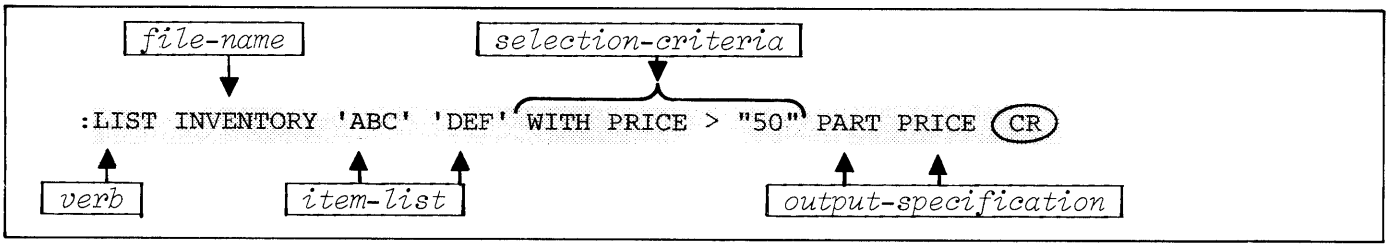


Figure B. Sample ENGLISH Input Sentenct

2 ELEMENTS OF THE ENGLISH LANGUAGE

2.1 FORMING ENGLISH INPUT SENTENCES

The user forms ENGLISH input sentences which specify the 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, together with the use of 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 input sentence then directs the appropriate ENGLISH processor to perform the specified data retrieval function. The general form of 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 those items eligible for consideration (the absence of an item-list implies all items). An item-list consists of specifically enumerated item-ID's, each enclosed within single quotes, additionally constrained by relational operators and logic connectives. Selection-criteria further limit the items for output to those meeting the specified conditions. Output-specifications enumerate those attributes 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 carriage return. A sentence may be continued to a second line by ending the first line with a shift-control-O followed by a carriage return.

4. Exactly one file-name must appear in each sentence. File-names may consist of any sequence of non-blank 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 file.

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

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

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

Figure A. Some 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)

:DELETE-LIST TEMP (CR)

:T-DUMP XYZ WITH VALUE1 (CR)

:I-DUMP XYZ < '505' (CR)

:ISTAT ACCOUNT (CR)
```

Figure B. Sample ENGLISH Input Sentences

2 ELEMENTS OF THE ENGLISH LANGUAGE

2.2 AN OVERVIEW OF THE ENGLISH VERBS

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

ENGLISH verbs are action oriented words which evoke specific ENGLISH processors. The common ENGLISH verbs are listed in Figure A and are briefly discussed below. (A separate section 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. Generated output is 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 verb TERM. If more attributes have been specified than will fit across the page, a non-columnar output is generated with the attribute names down the side and the associated attribute values to the right. LIST-LABEL and SORT-LABEL are analogous to LIST and SORT but allow special formatting for printing labels and other formatted output.

COUNT

The COUNT verb counts the number of items meeting the conditions as 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 STAT verb additionally provides a count and average for the specified attribute. The output generated by these verbs are the derived statistics.

SELECT and SSELECT

The SELECT verb provides a facility to select a set of items using the item-list and the selection-criteria. These selected items are available one at a time to TCL-II processors. The output from the SELECT verb is a message signaling the number of items extracted or selected. The user then responds by typing in a single TCL-II sentence. In the processing of the sentence, all items will be processed from the previously selected item-list. The SSELECT verb combines the SORT capability with the SELECT capability.

SAVE-LIST, GET-LIST, and DELETE-LIST

The SAVE-LIST, GET-LIST, and DELETE-LIST verbs are used to save, restore, and delete item-lists created by SELECT and SSELECT statements.

T-DUMP, I-DUMP, and ISTAT

The T-DUMP and I-DUMP verbs allow the user to selectively dump his dictionaries and data files to the magnetic tape or to the terminal, respectively. The ISTAT verb provides a file hashing histogram.

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* LE	*less than or equal to*
# *or* NE *or* NOT *or* NO	*not equal to or null attribute value*
	If a realtional 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-BALNC	*Selection-criteria which selects items having a null value for attribute CURR-BALNC.*

Figure C. Sample Usage of Relational Operators and Logical Connectives

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 the items eligible for processing (refer to the topic titled FORMING ITEM-LISTS), or may be used in selection-criteria to limit items to those whose attribute meets the specified conditions (refer to the topic titled 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-ID's or values are considered to be "equal". If an unequal pair of characters are found, the characters are ranked according to their numeric ASCII code equivalents (refer to the LIST OF ASCII CODES in the appendix to 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-ID's (values) are non-numeric, the item-ID (value) with more characters is considered "greater". If both item-ID's (values) are of equal length, character pair comparison, as for left-justified attributes, is used.

The logical connectives are listed in Figure B. Logical connectives bind together sets of item-ID's into item-lists, sets of values into value-lists, and sets of selection-criterion into selection-criteria. 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 (↑ or ^) may be used as an ignore character in any value or item-ID. All comparisons made against the value or item-ID then ignores the characters in the corresponding positions.

Figure C exemplifies the use of the relational operators and the 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 constructs 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 (V12.).

```
:SORT ONLY TEXT > '13' AND < '17' (CR)
PAGE 1                                          15:32:19  20 AUG 1976

TEST......

14
15
16

3 ITEMS LISTED.
:SORT ONLY TEST >= '13' AND <= '16' OR >= '18' AND < '19' (CR)
PAGE 1                                          15:33:01  20 AUG 1976

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  20 AUG 1976

TEST......

10
11
12
14
16
18

6 ITEMS LISTED.

:SORT ONLY TEST BEFORE '13' (CR)

PAGE 1                                          15:34:24  20 AUG 1976

TEST......

10
11
12

3 ITEMS LISTED.

:
```

Figure C. Sample Usage of Item-List

2.4 FORMING ITEM-LISTS

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

An item-list defines those 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-ID's surrounded by single quotes (e.g., 'XYZ'). These item-ID's 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-ID's 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 issued, only those items will be accessed, and processing will be faster.

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

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

This item-list selects all items with item-ID's less than 'A', or with item-ID's greater than 'B' but less than 'C', or with item-ID's 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 as follows:

```
< '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-ID's in file TEST. (TEST contains 10 items, with item-ID's '10' through '19'.) The word ONLY used in these examples specifies that only the item-ID's are to be listed.

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

Figure A. General Form of Simple Item-List

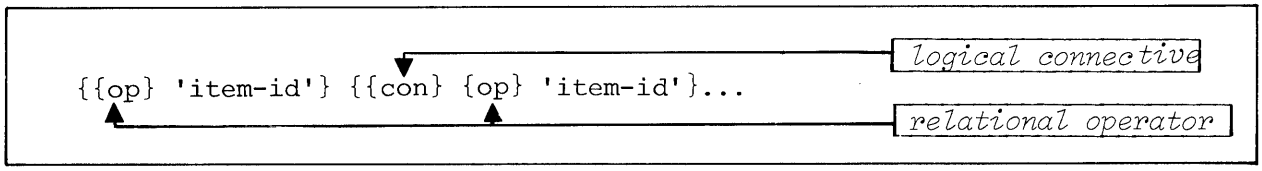


Figure B. General Form of Complex Item-List

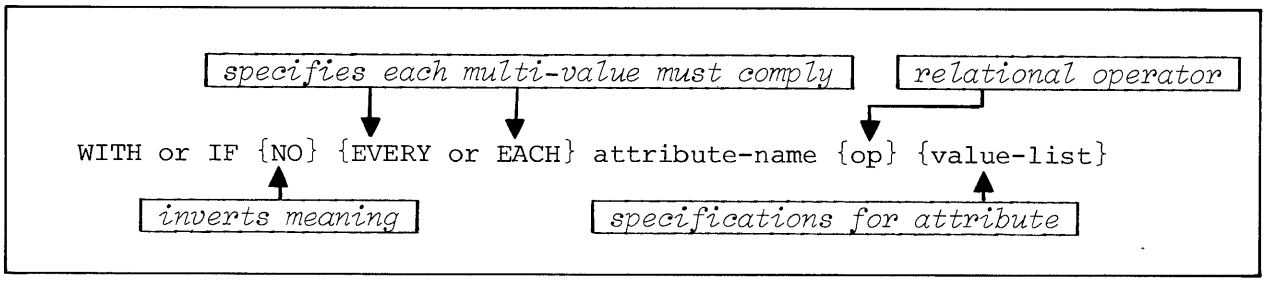


Figure A. General Form of Selection-Criterion

```
:LIST ACCOUNT NAME WITH AVG-USAGE "20" OR "25" AND O^CS (CR)
:WITH SEWER-ASMT "150" OR WITH AVG-USAGE "20" OR "30" O^CS (CR)
:AND WITH BILL-RATE > "30" (CR)

PAGE 1                                              17:36:04  20 AUG 1976

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

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

3 ITEMS LISTED.

:COUNT ACCOUNT WITH CURR-BALNC > "100" AND WITH SEWER-ASMT O^CS (CR)
:OR BILL-RATE = "30" (CR)

7 ITEMS COUNTED.

:LIST ACCOUNT TRNS-DATE WITH EVERY TRNS-DATE BEFORE "3/18/70" (CR)

PAGE 1                                              17:40:57  20 AUG 1976

ACCOUNT...  TRNS-DATE..

35090       31 DEC 1967
            31 DEC 1967
            31 DEC 1967
            31 DEC 1967
            31 DEC 1967
            31 DEC 1967

END OF LIST
```

Figure C. Sample Usage of Selection-Criteria

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 or more selection-criterion.

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. The rules for forming value-lists are identical to those for forming item-lists (refer to the topic titled FORMING ITEM-LISTS), except that double quotes must surround the actual values. For example, the following selection-criterion is met by those 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 those items which have at least one value for the specified attribute name. The selection-criterion may be further modified by using either or both of the modifiers EVERY or NO immediately following the WITH. The modifier EVERY requires that every value for the attribute meet the specified condition, i.e., if the attribute has multi-values, then each value must meet the condition. (The modifier EACH is a synonym for EVERY). The modifier NO reverses (inverts) the sense of the entire selection-criterion.

Several selection-criterion 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-criteria may consist of up to nine "AND clauses". An AND clause is made up of any number of selection-criterion bound by AND connectives. The AND clause is terminated when an OR connective is found in the left to right scan. (Note: the absence of an AND connective implies an OR connective.) For an item to pass the selection-criteria, the conditions specified 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)
```

Figure B illustrates further examples of selection-criteria.

```
:LIST ACCOUNT WITH NAME "[MAN" NAME (CR)
PAGE 1                                          18:13:27  20 AUG 1976

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  20 AUG 1976

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

11070       A A ALTHOFF

END OF LIST
:LIST ACCOUNT WITH NAME "[INE]" NAME (CR)
PAGE 1                                          18:16:56  20 AUG 1976

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

11095       J B STEINER
35065       L J RUFFINE

2 ITEMS LISTED.
```

Figure A. Sample Usage of String Searching Selection-Criterion

```
:SORT ACCOUNT SIZE WITH SIZE > "300" (CR)

PAGE 1                                          18:20:52  20 AUG 1976

ACCOUNT...  SIZE..

23060       596
23075       317
23080       318
35085       404

4 ITEMS LISTED.
```

Figure B. Sample Usage of SIZE Selection-Criterion

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 character ([) and the right-bracket character (]) may be used within the double-quotes in a selection-criterion. 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.

The user should note that this string searching capability may not be used in a simple item-list, but may be used with complex item-lists, as defined in Section 2.4.

Item Size

The size of items may be used as a selection-criterion. This is accomplished by referencing the special D/SIZE item in the user's Master Dictionary; this will cause the size of the item (as defined 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 item with the name SIZE (or any other unique name) in the dictionary of the file being used. For example, the user would create via the EDITOR an item with the name SIZE in the dictionary of the ACCOUNT file (e.g., EDIT DICT ACCOUNT SIZE) with the following attributes:

```
001 A
002 9999
003
004
005
006
007 MD0,
008
009 R
010 6
```

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

```
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  21 AUG 1976

ACCOUNT...  NAME.............  ADDRESS.............  CURR-BALNC..

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 1                                                09:11:53  21 AUG 1976

ACCOUNT   :  35060
NAME      J A SCHWARTA
ADDRESS      331 DOCK WAY
CURR-BALNC      $ 33,822.34
BILL-RATE      2
AVG-USAGE      31

END OF LIST
```

Figure C. Non-Columnar Output Format

2.7 FORMING OUTPUT-SPECIFICATION

Output-specifications enumerate those attributes to be listed.

All attribute names in an ENGLISH sentence which are not part of a selection-criterion (i.e., preceded by the modifier WITH) or are 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 out 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-ID's 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 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 or the attribute name, 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 are displayed in a single line header across the top of the page. The values for each of the items are then displayed in their respective columns. The attribute name header is 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 Figures B and C. Figure B shows a columnar output format, while Figure C shows a non-columnar output format.

---

*Refer to the topic titled USING MODIFIERS AND THROWAWAY CONNECTIVES

```
:LIST INV TRAN-DATE TRAN-TYPE TRAN-QTY (CR)

PAGE 1                                              18:18:36  20 AUG 1976

INV.......  TRAN-DATE  TRAN-TYPE  TRAN-QTY
                       *          *

1242-22     11 FEB     I          100
                       R           48
                       S           31
            12 FEB     I          144
                       R           43
                       S           66
1242-11     11 FEB     I           19
                       R          122
                       S           33
            12 FEB     I           97
                       R           39
                       S            7

2 ITEMS LISTED.
```

Figure A. Selection of All Values for Multi-Valued Attribute

```
:LIST INV TRAN-DATE "11 FEB 76" TRAN-TYPE O^CS (CR)
:TRAN-QTY(CR)

PAGE 1                                              18:20:04  20 AUG 1976

INV.......  TRAN-DATE  TRAN-TYPE  TRAN-QTY
                       *          *

1242-22     11 FEB     I          100
                       R           48
                       S           31
1242-11     11 FEB     I           19
                       R          122
                       S           33

2 ITEMS LISTED.
```

Figure B. Selection of Specific Value for Multi-Valued Attribute

2.8 OUTPUT CRITERIA: MULTI-VALUED ATTRIBUTE OUTPUT

Selection criteria may also be used to select specific values from multi-valued attributes.

Selection for output of specific values from multi-valued attributes can be accomplished by placing the desired value (or values) in double-quotes (" ") immediately following the attribute name. If the attribute is an associative attribute (D1) then the corresponding values from the D2 attributes will also be returned. (For further information regarding associative attributes, the user should refer to the section titled CORRELATIVES AND CONVERSIONS.)

The example in Figure A lists all the items in the INV file which contain any value for the attribute TRAN-DATE. In the example in Figure B, the TRAN-DATE "11 FEB 75" portion of the ENGLISH sentence indicates to the ENGLISH processor that only the dates equal to 11 Feb 76 are to be retrieved. Since TRAN-DATE is a "D1" attribute, only the values associated with the 11th of February for TRAN-TYPE and TRAN-QTY (which are "D2" attributes) are retrieved.

NAME	A/AMC	VALUE	MEANING
D/CODE	1	A	Attribute Definition Item.
		S	Attribute Synonym Definition Item.
		X	Special synonym to maintain order (but defined attribute is not output by ENGLISH).
A/AMC	2	attrib-num	For A-code attributes; defines attribute number.
S/NAME	3	text-name	For S-code attributes; defines attribute heading to be output by ENGLISH (note: these names may be padded with blanks to align non-columnar output).
S/AMC	4	attrib-num	For S-code attributes; defines attribute number.
V/TYPE	9	L	For columnar output only; specifies left justification. If value size is greater than column width, value is folded.
		R	For columnar output only; specifies right justification. If value size is greater than column width, value overlays previous 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 is larger than V/MAX.
		Ln	For non-columnar output; specifies left justification and reserves characters for each repetition of a multi-value. Multi-values will fold at end of line and repeat aligned with the start of the first value.
		Rn	For non-columnar output; as for L but right justification in the reserved value area.

NOTE: *For further details, see the Reality Programmer's Reference Manual*

Figure B. Attribute Definition Item Summary

2.9 OMISSION OF THE OUTPUT-SPECIFICATION

If no output-specifications appear, attributes defined by Attribute Synonym Definition Items are selected. This special feature is outlined below; however, for a complete description of Attribute Synonym Definition Items and their use, the user should refer to the Reality Programmer's Reference Manual.

If all output-specifications are omitted, then attributes which are defined in the dictionary via Attribute Synonym Definition Items (i.e., with D/CODE's of S or X) will be assumed as the output specification. The Attribute Synonym Definition Items used are those with item-ID's which are numeric and sequential (i.e., 1, 2, 3, 4 ...). Attributes with D/CODE's of S are listed; attributes with D/CODES's of X are not listed (i.e., they are only used to maintain the required sequential order). Attribute Synonym Definition Items have a special format (see Figure B) specifying the heading-name to be used for output.

Item-ID's are always included in the output format unless the modifier ID-SUPP is used (which suppresses the display of item-ID's). If an output is desired which is to only list the item-ID's, then the modifier ONLY must precede the file-name to inhibit the listing of the attributes defined by Attribute Synonym Definition Items.

Figure A shows a sample statement with the output-specification omitted. Figure B summarizes the various dictionary attributes as they apply to the formatting of output produced by an ENGLISH operation.

```
:LIST ACCOUNT '35095' (CR)

PAGE 1                                          18:24:04  20 AUG 1976

ACCOUNT  :  35095
NEXT-ACCNT     35100
CSTMR-NAME     A W FEVERSTEIN
SERVO-ADDR     324 CARNATION
MAIL-ADDR.     19401 DORAL
MAIL-CITY.     ANOTHER CITY
MAIL-STATE     CA
ZIP-CODE..     1925
ACCNT-STAT     A
DEPOSIT-=.     10.00
START-DATE     01 JAN 1968
BILL-RATE.     0.35
AVE-USAGE.     32
CURR-BALNC     19.25
60-DAYS..      9.80

END OF LIST
```

Figure A. Sample Omission of the Output-Specification

Modifier	Description
PAGE	When output is to the terminal, this modifier will halt output at the end of each page; output of the next page will resume when the user enters a carriage return.
TOTAL	Causes totals for the attributes which follow to be accumulated (see topic titled GENERATING TOTALS).
WITH or IF	Designates a selection-criteria (see topic titled FORMING SELECTION-CRITERIA).

A, AN, ARE, ANY, FILE, FOR, IN, ITEMS, OF, OR, and THE are throwaway connectives which do not affect the meaning of the ENGLISH sentence. They may be used anywhere in the sentence and are included to provide a degree of naturalness to the language.

Figure A illustrates the use of the modifiers and throwaway connectives.

```
:LIST THE ACCOUNT FILE PAGE DBL-SPC (CR)
```

This sentence causes the ACCOUNT file to be listed. Listing will be double-spaced, and output will halt at the end of each page.

```
: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 line 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.

Figure A. Sample Usage of Modifiers and Throwaway Connectives

2.10 USING MODIFIERS AND THROWAWAY CONNECTIVES

Modifiers may be used to further modify the meaning of ENGLISH sentences. Throwaway connectives do not affect the meaning of ENGLISH sentences; they may be used simply to add naturalness to the sentence.

The modifiers which may be used in an ENGLISH sentence are listed in the alphabetical order below.

Modifier	Description
BREAK-ON	Defines control-breaks (see topic titled 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 topic titled THE SORT VERB).
BY-DSND	Identical to BY, except that sequencing is done in a descending order (see topic titled THE SORT VERB).
COL-HDR-SUPP	Suppress the output of the time/date heading, the column headings, and the end-of-list message.
DBL-SPC	Causes output to be double-spaced.
DET-SUPP	Suppresses detail output when used with TOTAL or BREAK-ON modifiers (see topic titled GENERATING SUBTOTALS USING CONTROL-BREAKS).
DICT	Modifies the file-name so that the ENGLISH sentence references the file dictionary instead of the file (see topic titled FORMING ENGLISH INPUT SENTENCES).
EVERY or EACH	Modifies a selection-criterion so that every value for a multi-valued attribute must meet the specified condition for the criterion to be true. This modifier must immediately follow the modifier WITH (see topic titled FORMING SELECTION-CRITERIA).
HDR-SUPP or SUPP	Suppresses the output of the time/date heading and the end-of-list message.
ID-SUPP	Suppresses the display of item-ID's for LIST and SORT operations.
LPTR or (P)	Routes output to line printer.
ONLY	Inhibits the appending of the special synonym attributes when a null output-specification is encountered (see topic titled OMISSION OF THE OUTPUT-SPECIFICATION); when used, must precede the file-name.

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).*
'P'	*PAGE. Inserts the current page number.*
'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 B. Special HEADING Option Characters

```
:SORT ACCOUNT NAME HEADING " NAME LIST AT 'TL' PAGE NO. 'PL'" (CR)

NAME LIST AT 10:29:39  21 AUG 1976
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
11035       R S MARCUS
11040       E G MCCARTHY
11045       F R DRESCH
11050       J R MARSCHECK
11055       W H KOONS
11060       F T NATORI
11065       C V RANDALL
  .           .
  .           .
  .           .
```

Figure C. Sample Usage of Heading

2.11 GENERATING HEADINGS

LIST and SORT statements may optionally specify headings.

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 appear 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']text['options']..."
```

As examples:

```
HEADING "INVENTORY LIST"

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

The first example prints the title "INVENTORY LIST" at the top of each page. In the second example a heading has been specified which consists of the words "STATUS REPORT", followed by a carriage return and a line feed, followed by the word "PAGE:", followed by the current page number. The special option characters to be enclosed in single quotes are listed in Figure B. To actually print a single quote mark within the heading text, a sequence of two single quotes (' ') may be used.

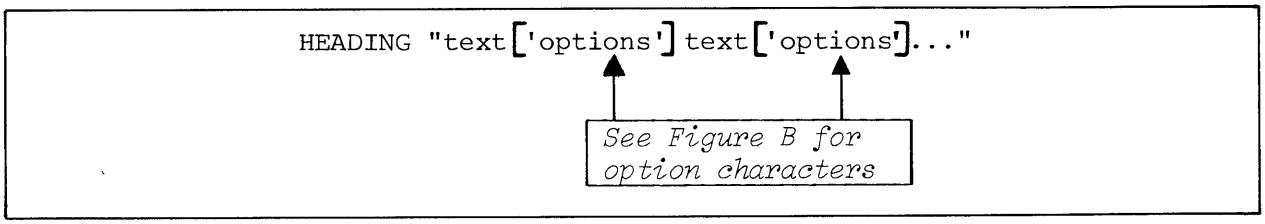


Figure A. General Form of Heading Specification

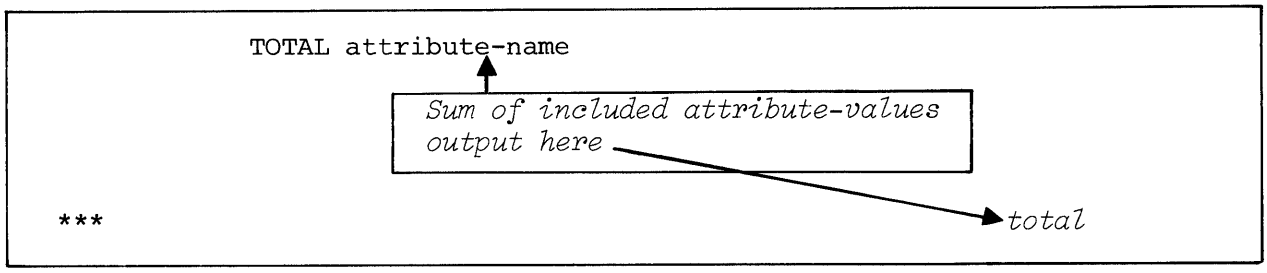


Figure A. General Form of TOTAL Specification and Output Line

```
:LIST ACCOUNT AFTER '35090' NAME ADDRESS TOTAL DEPOSIT (CR)

PAGE 1                                              10:31:23  21 AUG 1976

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 B. Sample Usage of TOTAL

2.12 GENERATING TOTALS

LIST and SORT statements may optionally specify totals.

A LIST or SORT statement can be used to generate a total. A TOTAL specification has the following 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 B.

The TOTAL modifier is also used in conjunction with the BREAK-ON modifier to output subtotals, as described in the section GENERATING SUBTOTALS USING CONTROL BREAKS.

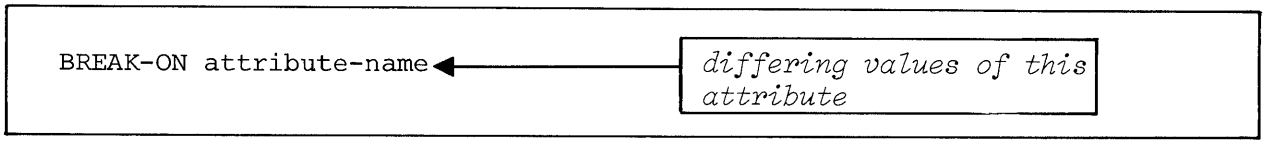


Figure A. Minimum BREAK-ON FORM

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

PAGE 1                                                09:34:01  27 APR 1976

ACCOUNT...  NAME.............  STREET.............  CURR-BALNC..

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 Control-Break

2.13 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 the following 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 the BREAK-ON's 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 comparision 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 the section 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), preceded and followed by blank lines.

For multiple control-breaks, output proceeds from lowest level BREAK to highest level. The data associated with the lowest level control-break is printed on the current page (even if the end of the page has been reached). If multiple 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 (see next topic).

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

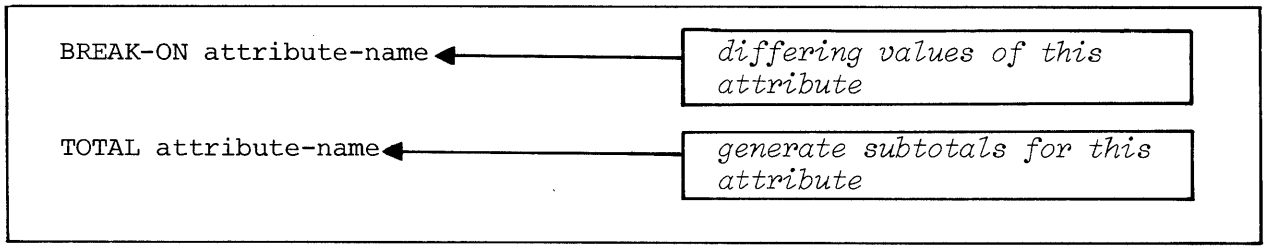


Figure A. Minimum BREAK-ON Form and TOTAL Form

```
:SORT ACCOUNT WITH BILL-RATE "2" "40" NAME BREAK-ON O^cs (CR)
:BILL-RATE TOTAL CURR-BALNC BY BILL-RATE (CR)

PAGE 1                                        09:28:03  22 AUG 1976

ACCOUNT...  NAME.............  BILL-RATE  CURR-BALNC..

35060       J A SCHWARTA              2  $ 33,822.34
35085       J F SITAR                 2  $    200.00

***                                      $ 34,022.34

11100       E F CHALMERS             40  $     17.50
35075       J L CUNNINGHAM           40  $      7.50

                                    ***  $     25.20

***                                      $ 34,047.54

4 ITEMS LISTED.
```

Figure B. Sample Usage of Control-Breaks

2.14 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 totals, 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 occured.

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 section 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 titled OUTPUT OPTIONS FOR CONTROL BREAKS.

```
BREAK-ON attribute-name  "text ['options'] text"
```

See Figure A for options characters

Figure B. General Form of BREAK-ON Specification

```
:SORT ACCOUNT WITH BILL-RATE "2" "40" NAME BREAK-ON BILL-RATE O^CS (CR)
:"SUB-TOTAL FOR 'V'" TOTAL CURR-BALNC BY BILL-RATE (CR)

PAGE 1                                                09:32:04  22 AUG 1976

ACCOUNT...  NAME.............  BILL-RATE  CURR-BALNC..

35060       J A SCHWARTA                2  $ 33,822.34
35085       J F SITAR                   2  $    200.00

                          SUB-TOTAL FOR  2  $ 34,022.34

11100       EF CHALMERS                40  $     17.50
35075       J L CUNNINGHAM             40  $      7.70

                          SUB-TOTAL FOR 40  $     25.20
***                                         $ 34,047.54

4 ITEMS LISTED.

:SORT ACCOUNT WITH BILL-RATE "2" "40" BREAK-ON BILL-RATE O^CS (CR)
:"SUB-TOTAL FOR 'V'" TOTAL CURR-BALNC BY BILL-RATE DET-SUPP (CR)

PAGE 1                                                09:39:20  22 AUG 1976

ACCOUNT...  BILL-RATE  CURR-BALNC..

    SUB-TOTAL FOR   2  $ 34,022.34
    SUB-TOTAL FOR  40  $     25.20
***                    $ 34,047.54

3 ITEMS LISTED.
```

Figure C. Sample Usage of Control-Break Options

2.15 OUTPUT OPTIONS FOR CONTROL BREAKS

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

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

BREAK-ON attribute-name ["text['options']text"]

The text, if specified, replaces the default "***" field in the attribute-name column when the control-break printout line occurs. Options are used to modify some of the actions taken at control-break time; options are specified as one or more characters as listed in Figure A. The first example in Figure C 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 of Figure C. Suppression of the BREAK attribute data may be specified by using a MAX-LENGTH of zero for the attribute used with a BREAK-ON modifier (refer to the Reality Programmer's Reference Manual).

CHARACTER	MEANING
'B'	*BREAK. Specifies this control-break attribute-name as the one whose value is to be inserted in the ENGLISH page heading in place of the 'B' option in the HEADING specification (see the section GENERATING HEADINGS). It may not be meaningful to specify this option within more than one BREAK-ON specification.*
'D'	*DATA. Suppresses the break data line entirely if there was only one detail line since the last time this control-break occurred.*
'L'	*LINE. Suppresses the blank line preceding the break data line. This option will override the 'U' option, below.*
'P'	*PAGE. Causes a page eject after the data associated with this break has been output.*
'R'	*ROLLOVER. Inhibits page rollover, thus forcing all the data associated with this break to be current on the same page.*
'U'	*UNDERLINE. Causes the underlining of all specified TOTAL fields.*
'V'	*VALUE. Causes the value of the control-break to be inserted at this point in the BREAK-ON heading.*

Figure A. Special BREAK-ON Heading Option Characters

```
:LIST ACCOUNT WITH BILL-RATE "30" NAME ADDRESS BILL-RATE (CR)

PAGE 1                                              11:08:37  21 AUG 1976

ACCOUNT...  NAME.............  ADDRESS.............  BILL-RATE

11115       D R MASTERS        100 AVOCADO                30
11085       A B SEGUR          101 BAY STREET             30
11040       E G MCCARTHY       113 BEGONIA                30
11050       J R MARSHECK       125 BEGONIA                30
11020       J T O'BRIEN        124 ANCHOR PL              30
11095       J B STEINER        124 AVOCADO                30
11110       D L WEISBROD       106 AVOCADO                30
11015       L K HARMAN         118 ANCHOR PL              30
11105       C C GREEN          112 AVOCADO                30
11090       J W JENKINS        130 AVOCADO                30
23030       L J DEVOS          201 CARNATION              30

11 ITEMS LISTED.

:LIST ACCOUNT >= '23080' AND <= '23095' NAME ADDRESS O^CS (CR)
:START-DATE CURR-BALNC DEPOSIT (CR)

PAGE 1                                              11:19:58  21 AUG 1976

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

ACCOUNT  L  23979
NAME    J W YOUNG
ADDRESS    207 COVE STREET
START-DATE    27 MAR 1970
CURR-BALNC    $     89.32
DEPOSIT    10.00

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 THE ENGLISH VERBS

3.1 THE LIST VERB

LIST is an ENGLISH verb which is 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 line printer if the modifier LPTR is used). The sequence of the output listing will be the order in which the item-ID's have been enumerated in the ENGLISH sentence. If no item-ID's have been specified in the ENGLISH sentence, then all item-ID's are implied and LIST will present these items in the hash sequence in which they are sorted on 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 platen as defined by the TCL verb TERM (refer to the Reality Programmer's Manual). If more attributes have been specified than will fit across the page, a non-columnar 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 titled FORMING OUTPUT-SPECIFICATIONS and the topic titled OMISSION OF THE OUTPUT-SPECIFICATION.

Consider the following 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 file ACCOUNT) are to be listed. In this case a columnar output will be produced.

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

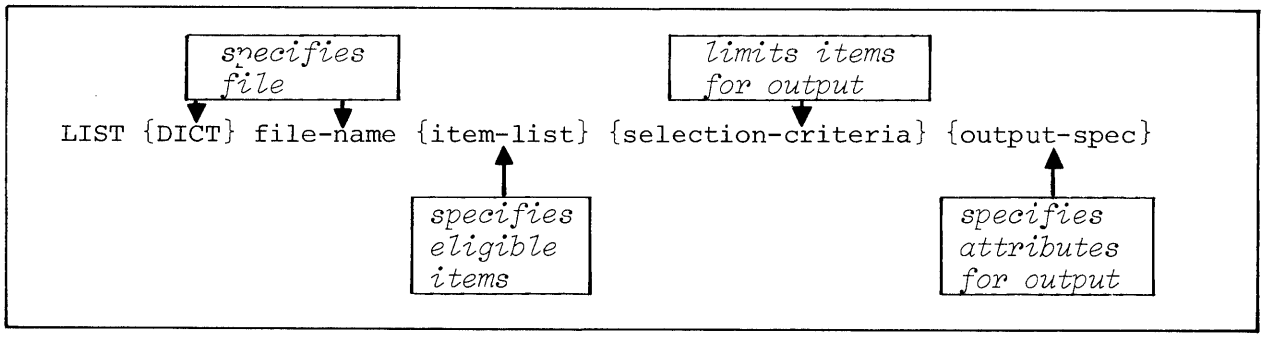


Figure A. General Form of ENGLISH Sentence Using LIST Verb

```
:SORT ACCOUNT GE '23000' AND LE '23020' NAME START-DATE (CR)

PAGE 1                                                  14:11:02  21 AUG 1976

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 O(CS) (CR)
:BY-DSND CURR-BALNC  CR

PAGE 1                                                  14:13:37  21 AUG 1976

ACCOUNT..   NAME.............  CURR-BALNC..

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.

:SORT ACCOUNT > '35070' NAME DEPOSIT BILL-RATE O(CS) (CR)
:BY DEPOSIT BY BILL-RATE  CR

PAGE 1                                                  14:15:47  21 AUG 1976

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

35090       D U WILDE           3.17
35100       R W FORSTROM        8.00       10.03
35110       H E KAPLOWITZ      10.00       10.03
35080       G A BUCKLES        10.00          35
35095       A W FEVERSTEIN     10.00          35
35105       S J FRYCKI         10.00          35
35075       J L CUNNINGHAM     10.00          40
35085       J F SITAR          12.00           2

8 ITEMS LISTED.
```

Figure B. Sample Usage of SORT Verb

3.2 THE SORT VERB

SORT is an ENGLISH verb which is 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 titled THE LIST VERB), except that a SORT operation orders the output in a specified sorted order. If the modifiers BY or BY-DSND are not used, then the SORT will sequence on the item-ID's (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).

Multiple BY and BY-DSND sort-keys may be intermixed at will, with the leftmost sort-key being most significant. If a descending 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.

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 positions are considered as blanks when compared. (For further information regarding character comparisons, refer to the topic titled 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 Function Correlative Processor (refer to the section titled CORRELATIVES) and the Conversion Processor (refer to the section titled CONVERSION). All correlative codes are processed in forming sort keys. (Refer to the overview of CORRELATIVES AND CONVERSIONS.) Note that several codes (MD, MT, Ml,D) do not affect the results of sorting, and should only be used as conversion codes in order to save processing time.

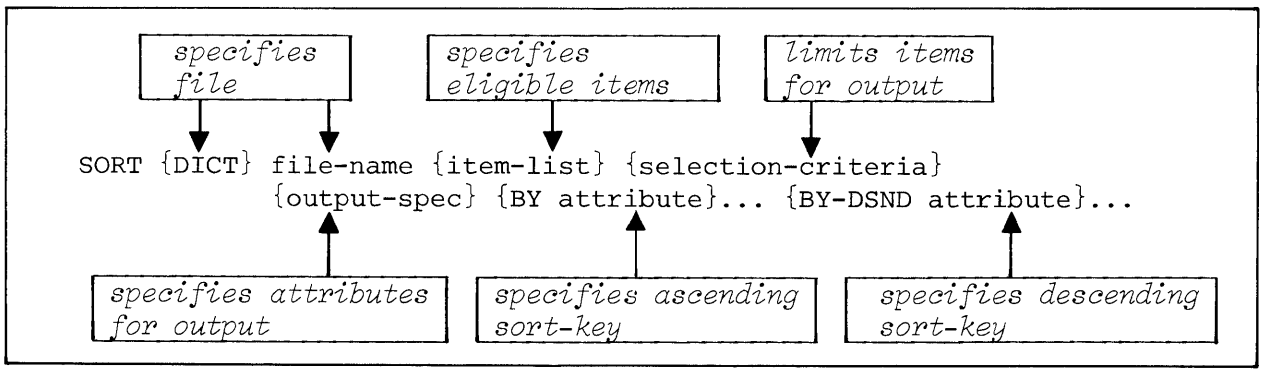


Figure A. General Form of ENGLISH Sentence Using SORT Verb

Input request lines:

```
:SORT-LABEL DICT NEWAC WITH D/CODE "A""S" LPTR BY D/CODE BY A/AMC

?6,5,1,10,11,1
?NEWAC....
?D/CODE...
?A/AMC....
?S/NAME...
?S/AMC....
```

Resulting printout:

```
PAGE 1                                                  16:02:39  25 JAN 1976

NEWAC...  *A0          *A1          D/CODE       *A2          A/AMC        F/BASE
D/CODE... A            A            A            A            A            A
A/AMC...  000          01           01           02           02           02
S/NAME...
S/AMC....

NEWAC.... *A3          F/MOD        S/NAME       *A4          F/SEP        S/AMC
D/CODE... A            A            A            A            A            A
A/AMC.... 03           03           03           04           04           04
S/NAME...
S/AMC....

NEWAC.... *A5          L/RET        *A6          L/UPD        *A7          V/CONV
D/CODE... A            A            A            A            A            A
A/AMC.... 05           05           06           06           07           07
S/NAME...
S/AMC....

NEWAC.... *A8          V/CORR       *A9          V/TYP        *A10         V/MAX
D/CODE... A            A            A            A            A            A
A/AMC.... 08           08           09           09           10           10
S/NAME...
S/AMC....

NEWAC.... *A11         V/MIN        *A12         V/EDIT       #A13         F/REALLOC
D/CODE... A            A            A            A            A            A
A/AMC.... 11           11           12           12           13           13
S/NAME...
S/AMC....

NEWAC.... *A9999       D/SIZE       1            2            3            4
D/CODE... A            A            S            S            S            S
A/AMC.... 9999         9999         01           02           03           04
S/NAME...                           D/CODE       A/AMC        S/NAME       S/AMC
S/MAC....                           01           02           03           04

NEWAC.... 7            8            9            10           11
D/CODE... S            S            S            S            S
A/AMC.... 07           08           09           10           11
S/NAME... V/CONV       V/CORR       V/TYP        V/MAX        V/MIN
S/AMC.... 07           08           09           10           11
```

Figure A. Sample Usage of SORT-LABEL

3.3 THE 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 section 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. Multi-values for an attribute will be treated as if they were separate value fields, and thus for most applications, multi-valued attributes should not be specified.

The normal non-columnar 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 essentially 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; is 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.

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.

An example of SORT-LABEL request entry and printout is shown in Figure A.

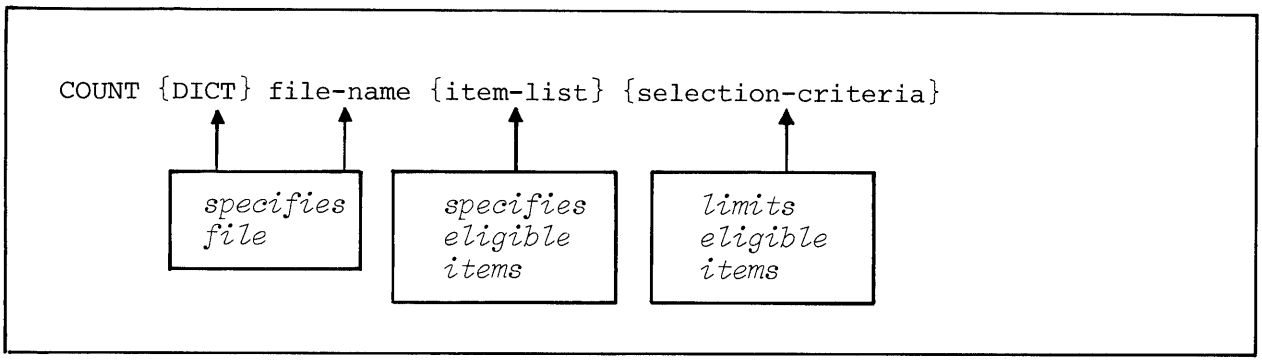


Figure A. General Form of ENGLISH Sentence Using COUNT Verb

```
: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

3.4 THE COUNT VERB

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

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

```
xxx ITEMS COUNTED
```

where xxx is the number of items which were counted. The maximum number of items which can be counted is 2,147,483,647 (2^{32}-1).

Consider the following example:

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

This sentence counts the items which have item-ID's greater than 533 and which have values of ABC for attribute A3.

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

```
:SUM ACCOUNT CURR-BALNC (CR)
TOTAL OF CURR-BALNC IS  :  $2,405,118.10
: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  :  2887
:SUM ACCOUNT DEPOSIT WITH CURR-BALNC LT "50000" (CR)
TOTAL OF DEPOSIT IS  : 542.17
:SUM ACCOUNT DEPOSIT WITH CURR-BALNC < "50000" AND WITH NO SEWER-ASMT (CR)
TOTAL OF DEPOSIT IS  :  460.00
```

Figure B. Sample Usage of SUM Verb

```
:STAT ACCOUNT TRASH-CHGS (CR)
STATISTICS OF TRASH-CHGS  :
TOTAL = 504.94 AVERAGE = 7.4255 COUNT = 68
: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 = 57 AVERAGE = 14.25 COUNT = 4
:STAT ACCOUNT DEPOSIT WITH NO CURR-BALNC (CR)
STATISTICS OF DEPOSIT  :
TOTAL = 39.00 AVERAGE = 7.8000 COUNT = 5
```

Figure C. Sample Usage of STAT Verb

3.5 THE 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 or STAT verbs are constructed as 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 compared 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.

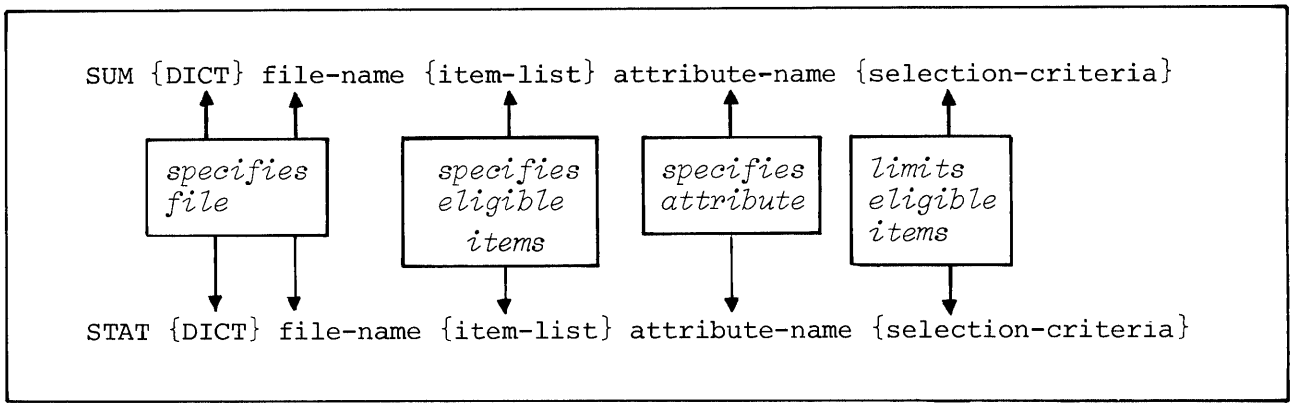


Figure A. General Form of ENGLISH Sentence Using SUM or STAT Verbs

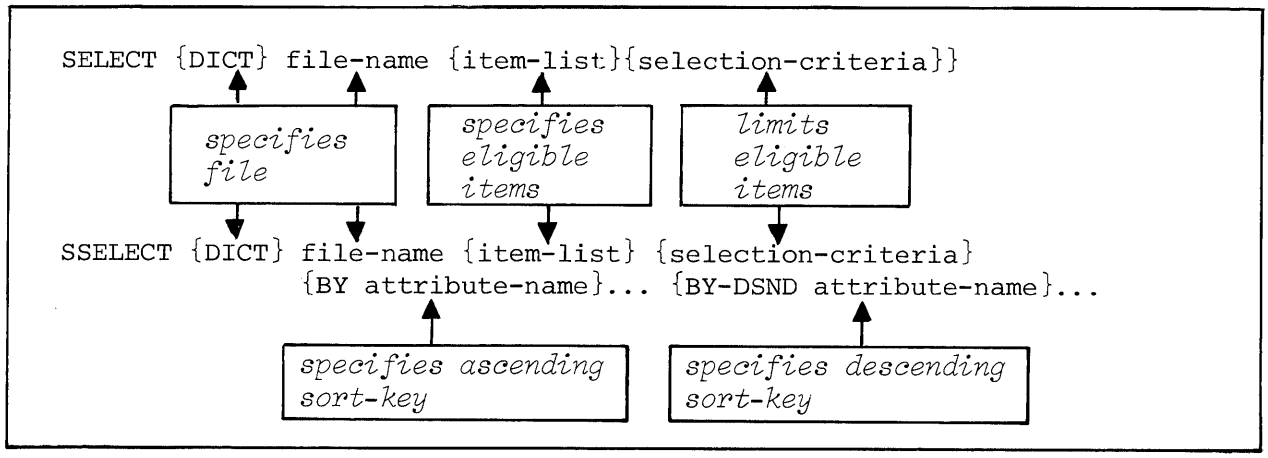


Figure A. General Form of ENGLISH Sentence Using SELECT or SSELECT Verb

```
: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)
```

Figure B. Sample Usage of Select Verb

```
:SSELECT ACCOUNT WITH BILL-RATE "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

3.6 THE SELECT AND SSELECT VERBS

SELECT is an ENGLISH verb which provides the facility to select a set of items using the item-list and the 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 are available one at a time, to other processors such as BASIC, BATCH, and the ENGLISH processors as well. In all the cases, one can do the selection from one file and use the item-ID's 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 processes, as follows:

BASIC program	The selected items are available to the BASIC program via the READNEXT statement (refer to the DATA BASIC Manual).
BATCH process	The item-ID from the selected item-list is available by specifying an asterisk (*) in the (only) input line to the BATCH string after the selection. At least one asterisk must be specified; the current item-ID will be substituted for every asterisk before the BATCH-string is processed (refer to the PROC AND BATCH Reference 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-ID's is not.
TCL-II process	The statement is entered without an item-list (for instance, "COPY DICT PARCEL-FILE (P,L)"). (Refer to the Reality Programmer's Reference Manual.)

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 "ST ON") (refer to the PROC AND BATCH Reference Manual).

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

```
SAVE-LIST x

GET-LIST x  account-name

DELETE-LIST x  account-name
```

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

```
:SSELECT ACCOUNT WITH BILL-RATE > ".35" BY NAME (CR)

24 ITEMS SELECTED.
:SAVE-LIST OVER.35 (CR)
 241  SYSPROG *L*OVER.35 CATALOGED.
1 FRAMES USED

:GET-LIST OVER.35 (CR)

24 ITEMS SELECTED.
:COPY ACCOUNT  R
  TO: (TAPE)

:DELETE-LIST OVER.35 (CR)
 242  SYSPROG *L*OVER.35 DELETED.
```

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

3.7 THE SAVE-LIST, GET-LIST, AND DELETE-LIST VERBS

The verbs SAVE-LIST, GET-LIST and DELETE-LIST are used to save, restore, and delete selected item-lists.

Use of SAVE-LIST, GET-LIST and DELETE-LIST is useful if several passes are to be made on the item-list. These verbs bypass the time consuming retrieve and sort phase of the SSELECT verb. These verbs use the system-level file called 'POINTER-FILE' to store a pointer to the saved list. These pointers are saved and restored by the SAVE/RESTORE processors.

The general form of the SAVE-LIST verb is as follows:

```
SAVE-LIST x
```

This is entered immediately after a SELECT or SSELECT statement (must be "stacked" if generated by a PROC--see the PROC AND BATCH Reference Manual). The parameter "x" is any string of non-blank characters that the user may specify to identify this selected item list; it may also be null. An item whose item-ID is "account-name*C*x" is created in the Pointer-File; any previously existing item list with the same identification is overlaid and the disc space it represents will be returned to the system. "Account-name" is the name that the user is logged on with.

The general form of the GET-LIST verb is as follows:

```
GET-LIST x  account-name
```

This statement retrieves a previously saved item-list, just as if the user entered a SELECT or SSELECT type of statement again. A message indicating the number of items in the item-list will be printed, and the item-list is available 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 AND BATCH Reference Manual). The optional "account-name" allows one user to access an item-list generated and saved by another user.

The general form of the DELETE-LIST verb is as follows:

```
DELETE-LIST x  account-name
```

This statement deletes a previously saved item-list. On a delete list, the frames used in the storage of the list are released to the system overflow space pool. Users with system privilege level two can optionally specify "account-name" to delete an item-list that was generated by another user.

```
:T-DUMP ACCOUNT > '23060' WITH CURR-BALNC (CR)
31 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 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 C. Sample Usage of T-DUMP and I-DUMP Verbs

```
:ISTAT ACCOUNT (CR)

FILE = ACCOUNT  MODULO=  3  SEPAR=  1          16:49:56  22 AUG 1976

005644 023 *>>>>>>>>>>>>>>>>>>>>>>>
005645 022 *>>>>>>>>>>>>>>>>>>>>>>
005646 023 *>>>>>>>>>>>>>>>>>>>>>>>

ITEM COUNT=68, BYTE COUNT = 16583, AVG. BYTES/ITEM =243
     AVG. ITEMS/GROUP = 22, STND DEV = .5

:ISTAT ACCOUNT > '23020 AND < '35090' (CR)
:FILE= ACCOUNT  MODULO=3 SEPAR= 1              16:51:07  22 AUG 1976

005644 012 *>>>>>>>>>>>>
005645 011 *>>>>>>>>>>>
005646 012 *>>>>>>>>>>>>

:ITEM COUNT= 34, BYTE COUNT = 8845, AVG. BYTES/ITEM = 260
      AVG. ITEMS/GROUP = 11, STND DEV = .5
```

Figure D. Sample Usage of ISTAT Verb

3.8 THE T-DUMP, I-DUMP, AND ISTAT VERBS

T-DUMP and I-DUMP are ENGLISH verbs which allow the user to selectively dump his dictionaries and data files to the magentic tape or to the terminal, respectively. The ISTAT verb provides a file hashing histogram.

T-DUMP and I-DUMP

An ENGLISH sentence using the T-DUMP or I-DUMP verb is constructed as 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 further information regarding magnetic tape operations, refer to the 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.

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

ISTAT

An ENGLISH sentence using the ISTAT verb is constructed as illustrated in Figure B. The ISTAT verb provides a file hashing histogram for the selected items in the selected file, as illustrated by the examples in Figure D. For further information regarding file hashing, refer to the Reality Programmer's Reference Manual.

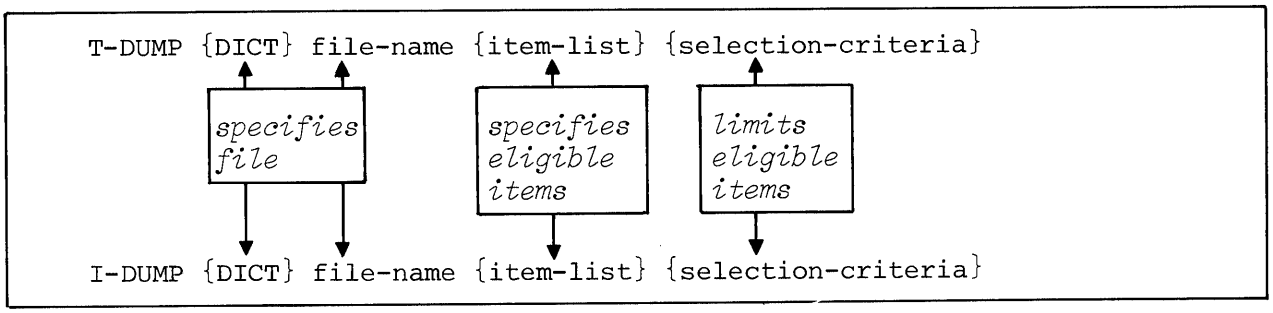


Figure A. General Form of ENGLISH Sentence Using T-DUMP or I-DUMP Verbs

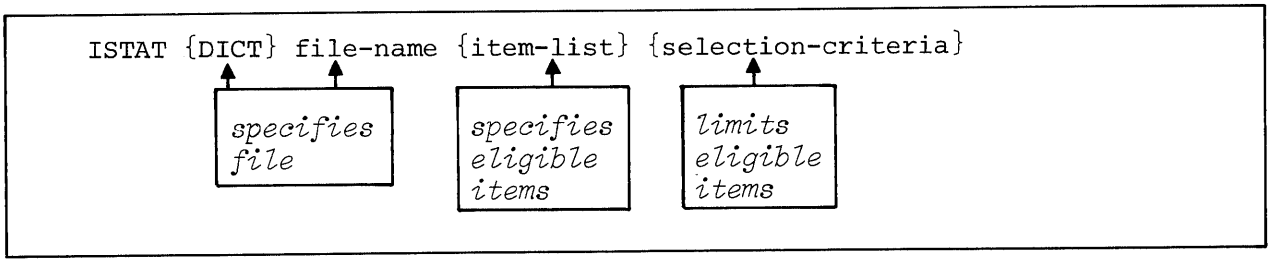


Figure B. General Form of ENGLISH Sentence Using ISTAT Verb

NAME	GENERAL FORM	DESCRIPTION
C	Cn*n*n*...	*CONCATENATE. Used to concatenate values.*
D	D{n}{*m}{s}	*DATE. Used to convert dates.*
D1	D1;amc;amc;amc...	*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	D2;amc	*DEFINE SECONDARY. Used to define a secondary associative attribute. May be used as a correlative code only.*
F	F;element;element...	*FUNCTION. Used to compute a mathematical function on a defined set of attributes.*
G	G{m}*n	*GROUP. Used to extract one or more contiguous segments from an attribute value.*
MD	MDn{m}{Z}{$}{i*}{c}	*MASK DECIMAL. Used to convert and scale numbers.*
MT	MT{H}{S}	*MASK TIME. Used to covert time.*
MX	MX	*MASK HEXADECIMAL. Used to convert character strings to Hexadecimal ASCII equivalents.*
T	T{m,}n	*TEST EXTRACTION. Used to extract a fixed number of characters from an attribute value.*
Tfile	Tfile;c;in-amc;out-amc	*FILE TRANSLATION. Used to convert values by translating through a file.*
U	Unxxx	*USER-DEFINED. Used to evoke user-defined conversion.*

Figure B. Processing Code Summary

item 'NAME'

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.*
011 1 ← *Minimum length.*

Figure C. Sample Attribute Definition Item Containing Processing Codes

4 CORRELATIVES AND CONVERSIONS

4.1 AN OVERVIEW

Two types of special processing fields are available when using the ENGLISH processors. CORRELATIVE codes are used to define special processing inter-relationships which are 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 attributes which will apply special conversion processing on the associated 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 (shift-control-m on most terminals); 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 may be illustrated as follows and is further defined in Figure A.

stored format — correlatives ---> — intermmediate format — conversions ---> — external 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 values 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 in order 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 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 contains only totals, Break field headings, 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

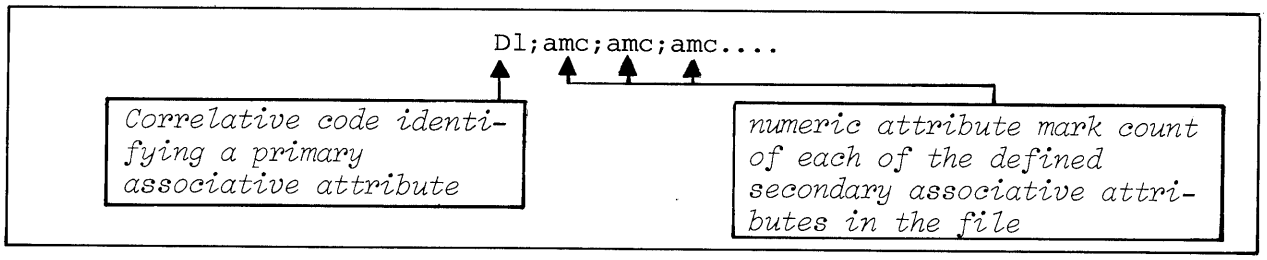


Figure A. General Form of D1 Correlative

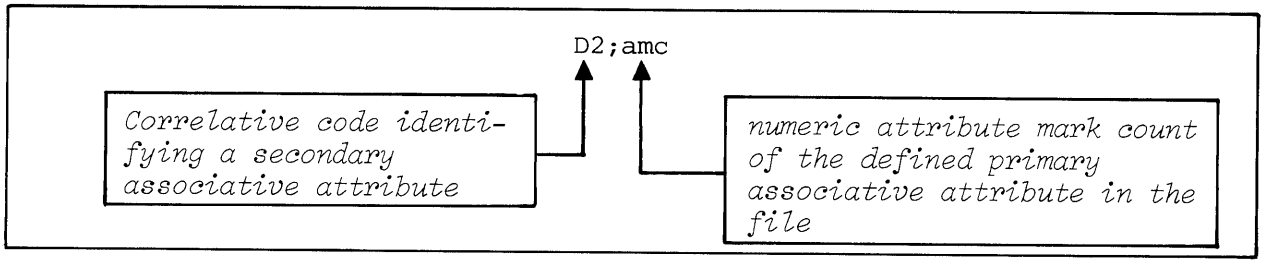


Figure B. General Form of D2 Correlative

```
:LIST TEST-FILE '5330' DATA CODE UNITS DOLLARS (CR)
PAGE 1                                          18:15:24  23 AUG 1976
TEST-FILE  DATE          CODE        UNITS        DOLLARS
                         *           *            *
5330       07 APR 1972               P                    9.50
           18 MAR 1972               B                    9.50
           17 MAR 1972               T                    2.00
           13 MAR 1972               R      2721          7.50
           05 FEB 1972               P                    9.20
           15 JAN 1972               B                    9.20
           14 JAN 1972               T                    2.00
           10 JAN 1972               R      2696          7.20
END OF LIST

:LIST DICT TEST-FILE 'DATE' 'CODE' 'UNITS' 'DOLLARS' (CR)
PAGE 1                                          18:15:27  23 AUG 1976
TEST-FILE  D/CODE.  A/AMC  S/NAME...  S/AMC  V/CONV...  V/CORR..... V/TYP V/MAX
DATE       S        20     DATE       20     D          D1;21;22;23 R     R11
CODE       S        21     CODE       21                D2;20       R     R11
UNITS      S        22     UNITS      22                D2;20       R     R11
DOLLARS    S        23     DOLLARS    23     MD2        D2;20       R     R11

END OF LIST
```

Figure C. Sample Use of D1, D2 Correlatives

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 may be specified as correlatives only.

The purpose of the D1, D2 correlatives is to provide a facility whereby a set of attributes (the secondary D2's 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 D2's, with each D2 relating back to (and grouped with) the primary D1 value.

The general form of the D1 correlative is as follows:

D1;amc;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 the separator.

The general form of the D2 correlative is as follows:

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 a 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 the attribute definition items for these attributes. For further examples illustrating the use of D correlatives via ENGLISH, see the topic titled SELECTION CRITERIA: MULTI-VALUED ATTRIBUTE SELECTION.

The D1 attribute may have multi-values, each separated by a value mark (ASCII 253). Each D2 attribute should have a corresponding number of multi-values; however, each of these multi-values may be multi-valued themselves. Each sub multi-value (called a secondary value) is separated by a secondary value mark (ASCII 252). For further information, see the Reality Programmer's Reference Manual.

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.

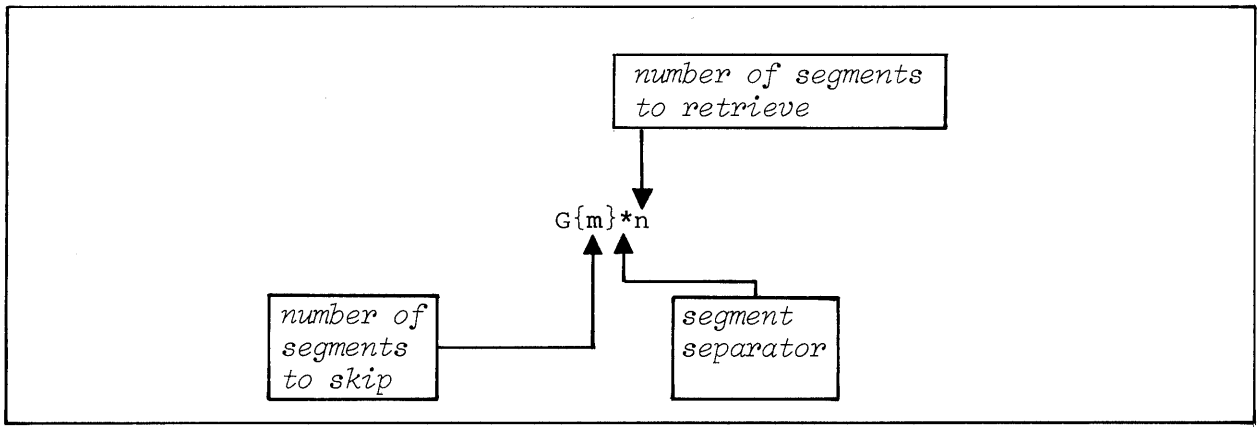


Figure A. General Form of G Code

CODE	ATTRIBUTE VALUE	VALUE OUTPUT
G$1	ABCDEFGHI$JKL	ABC
G1$2	ABCDEFGHI$JKL	DEF$GHI
G2$1	ABCDEFGHI$JKL	GHI
G1$1	ABCDEFGHI$JKL	DEF
G$2	ABCDEFGHI$JKL	ABC$DEF
G1A1	123A55555A22	55555
G2A1	123A55555A22	22

Figure B. Sample Usage of G Code

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 non-numeric character (except the minus sign or a system delimiter).

The general form of the G code is as follows:

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 non-numeric character which is the segment separator 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 illustrates several examples of the use of this code.

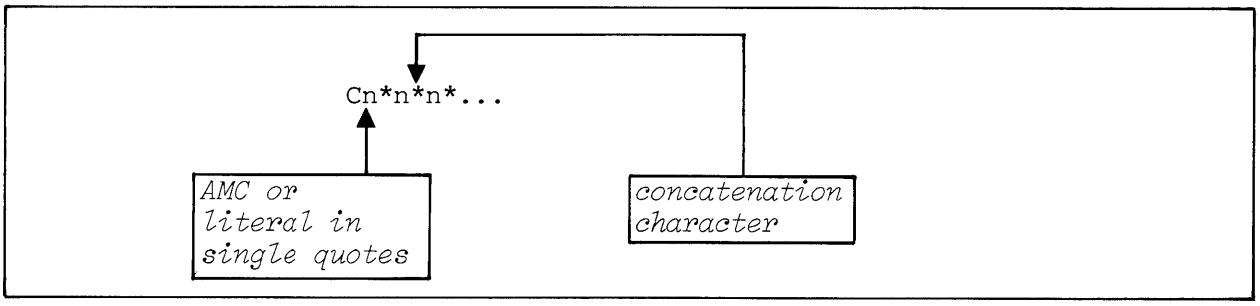


Figure A. General Form of C Code

Dictionary section of TEST file:

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

Data section of TEST file:

```
item '123'        item '456'

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

ENGLISH sentence:

```
:LIST TEST '123' '456' CAT (CR)

PAGE 1                                        12:05:33  24 AUG 1976

TEST.  CAT.................

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

2 ITEMS LISTED.
```

Figure B. Sample Usage of C Code

4.4 DEFINING CONCATENATIONS: 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 as follows:

```
Cn*n*n*...
```

where:

C is the code name.

* 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 non-numeric (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.

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

The user should note that the C code will not function with the BATCH processor.

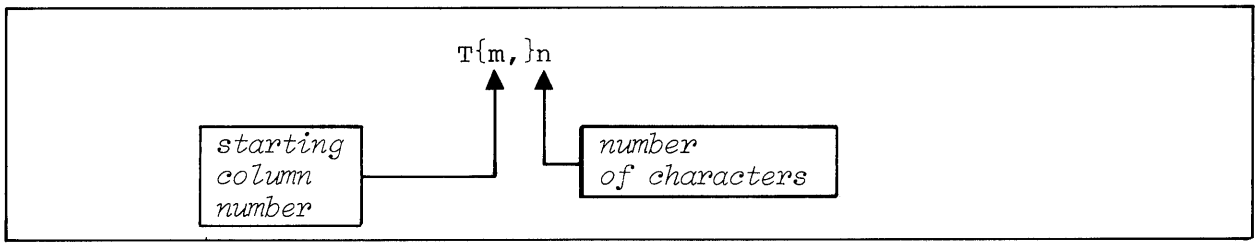


Figure A. General Form of T Code

CORRELATIVE	ATTRIBUTE VALUE	VALUE OUTPUT
T3,2	ABCDEFG	CD
T3,5	ABCDEFG	CDEFG
T2	ABCDEFG	AB
T7	ABCDEFG	ABCDEFG
T9	ABCDEFG	ABCDEFG
T8,1	65432XYZ	Z
T3	65432XYZ	654
T2,2	0123456789	12

Figure B. Sample Usage of T 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 via the use of a T code. The general form of the T code is as follows:

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.

If the form "Tm,n" is used, then n characters starting from character position 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 for dictionary attribute V/TYPE (see the Reality Programmer's Reference Manual).

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

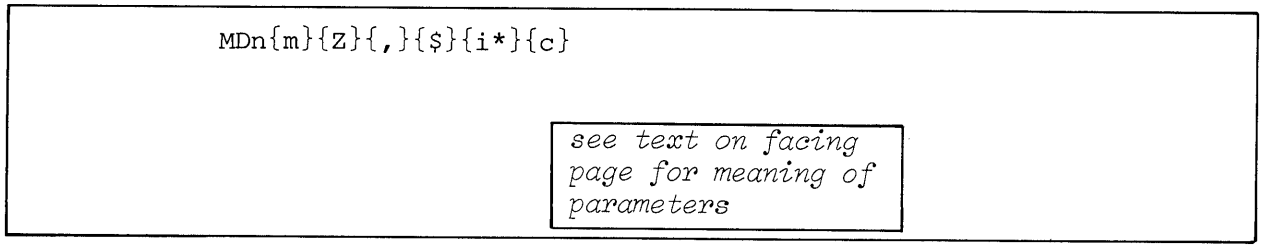


Figure A. General Form of MD Code

MD CODE	STORED VALUE	CONVERTED VALUE
MD2	1234567	12345.67
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	$999.99
MD2Z$<	-99999	$<999.99>
MD2Z,$12*C	1234567	$**12,345.67
MD2Z,$12*C	0	
MD2Z,$12*C	null	
MD2,$12 -	1234567	$ 12,345.67
MD2,$12 -	-1234	$ 12.34-
MD24,-	-1234567	123.46-
MD2,$12#	1234567	$##12,345.67
MD0,	1234567	1,234,567

Figure B. Sample Usage of MD Code

4.6 CONVERTING AND SCALING NUMBERS: MD

The MD code provides a facility for converting and scaling numers 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. The general form of the MD code is as follows:

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 this parameter is omitted, m=n is assumed.

Z is an optional parameter specifying the suppression of leading zeros.

, is an optional parameter for output which causes commas to be inserted between every thousands 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 non-numeric (is typically an asterisk).

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.

Figure B illustrates a number of MD conversion examples.

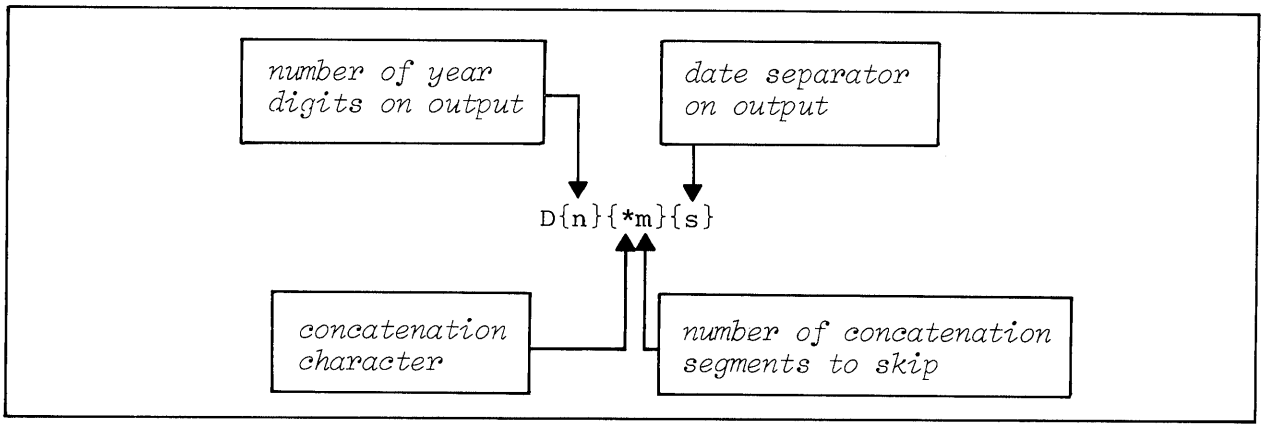


Figure A. General Form of D Code

D CODE	INTERNAL VALUE	OUTPUT VALUE
D	2704	27 MAY 1976
D/	2704	05/27/1976
D-	2707	05-27-1976
D0	2704	27 MAY
D0/	2704	05/27
D2*	2704	05*27*76
D	-13732	27 MAY 1930
D/	-13732	05/27/1930
D-	-13732	05-27-1930
D0/	19141	05/27
D2*	19141	05*27*20
D%1	ABC%2704	ABC%27 MAY 1976
D%1/	ABC%2704	ABC%05/27/1976
D%1-	ABC%2704	ABC%05-27-1976
D0%1	ABC%2704	ABC%27 MAY
D0	ABC%2704	ABC%2704

Figure B. Sample Usage of D Code

4.7 DEFINING DATA 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 as follows:

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.

* is an optional non-numeric delimiter that specifies the delimiter of the concatenated segments to be skipped before the date is found; * cannot be a system delimiter.

m is a single digit that must accompany * (if * is specified). Parameter m is the number of concatenated segments to be skipped before the date is found.

s is an optional non-numeric character that is to be used as the separator between the month, day, and year on output (the format being: mm s dd s yyyy). If s is omitted, the format will be: dd mmm yyyy.

The internat 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
18 MAY 1995	10000

Figure B illustrates a number of D conversion examples.

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).

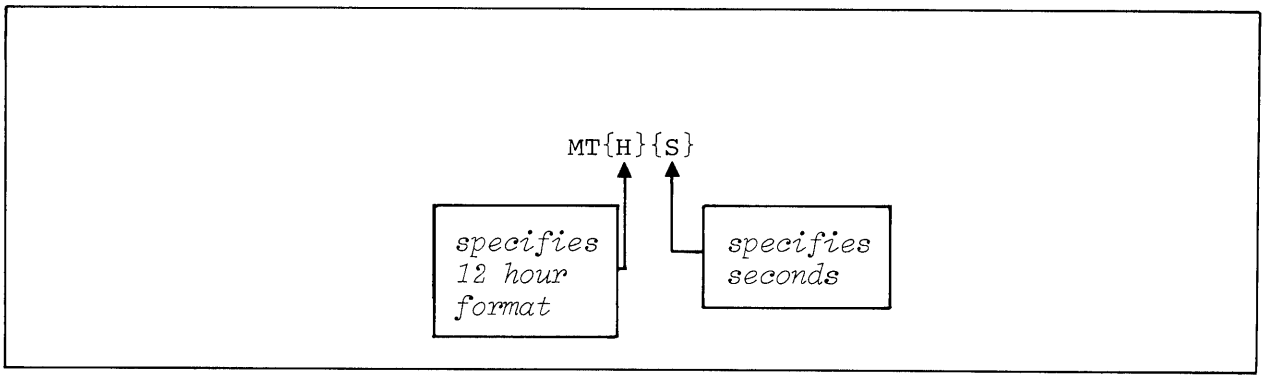


Figure A. General Form of MT Code

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	9	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

Figure B. Sample Usage of MT Code

4.8 DEFINING TIME FORMAT: MT

The MT code provides a facility for converting an external time 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 as follows:

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 12:00 AM is considered midnight, and 12:00 PM is considered noon. AM and PM will be ignored on input if code MT is specified. Illegal values are converted to null on input.

Figure B illustrates a number of examples using the MT conversion.

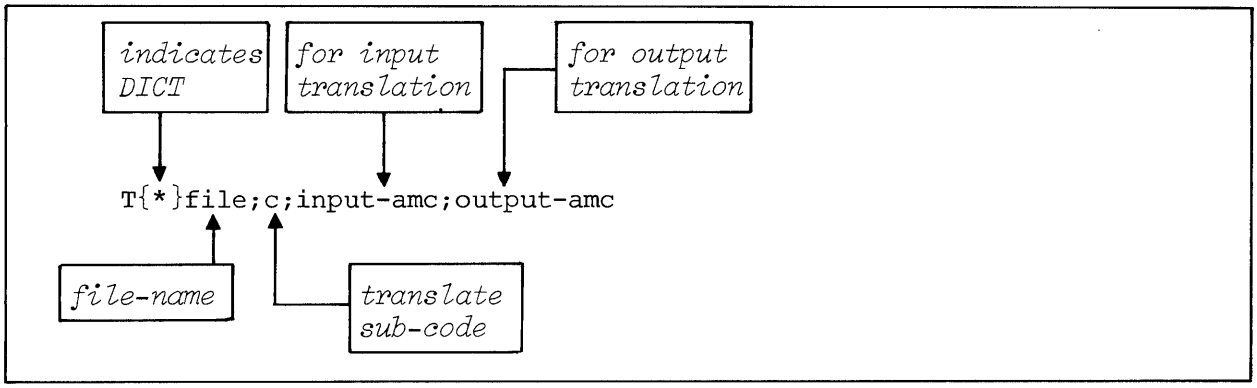


Figure A. General Form of Tfile Code

Dictionary Section of DETAIL file:

```
item 'NAME'
001 A
002 3
004
005
006
007 TMASTER;C;V;1;1
```

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  24 AUG 1976

DETAIL     NAME...

I1         SMITH
I2         BROWN
I3         JONES

3 ITEMS LISTED
```

Figure B. Sample Usage of Tfile Code

4.9 DEFINING FILE TRANSLATION: Tfile

The Tfile code provides a facility for converting a value by translating through a file.

The value to be translated 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 as follows:

```
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 sub-code, 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 like a V for input and a C for output.
- O - Output verify only; functions like 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 for input translation. After locating the translation item using the input value as the item-ID the attribute-value for the defined amc, 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 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).

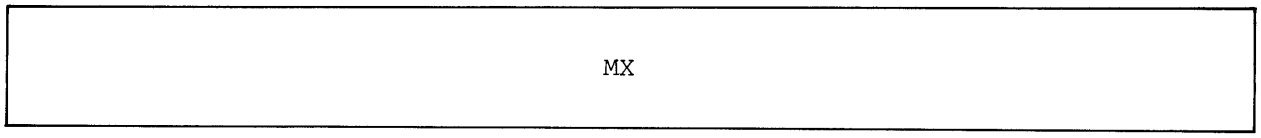


Figure A. General Form of MX Code

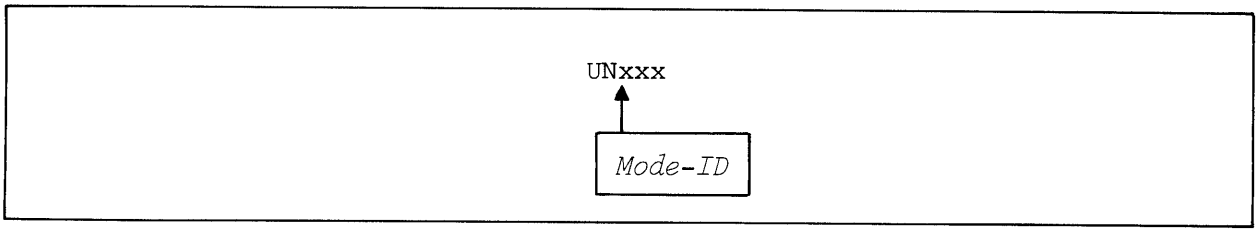


Figure B. General Form of U Code

MX CODE	STORED VALUE	CONVERTED VALUE
MX	ABC	414243
MX	ABC#	41424323
MX	T	54
MX	%T	2554
MX	XYZ	58595A
MX	....	2E2E2E2E

Figure C. Sample Usage of MX Code

4.10 DEFINING ASCII AND USER CONVERSIONS: MX AND U

The MX code is used to convert any string of characters stored on file to or from its corresponding hexadecimal ASCII equivalent. The U code permits a user-defined special purpose subroutine to be evoked 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 as follows:

```
MX
```

Figure C illustrates the use of the MX code.

U Code

A user-defined special purpose subroutine may be evoked for special conversion via the U code. The general form of the U code is as follows:

```
Unxxx
```

where:

U is the code name.

nxxx is the Mode-ID (refer to the Reality Assembly Language 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 Reality Assembly Language Manual.

OPERATOR	OPERATION
*	*Multiplication of the top two entries in the stack.*
/	*Division of STACK1 by STACK2.*
R	*Same as "/" but remainder is returned to top of stack (instead of quotient).*
+	*Addition of the top two entries in the stack.*
-	*Subtraction of STACK2 from STACK1.*
S	*A total sum of all previous computations is placed at the top of the stack. The S operator can only occur as the last entry in the F code.*
←	*Exchanges top two positions in stack.*
=	*"equal" relational operator.*
<	*"less than" relational operator.*
>	*"greater than" relational operator.*
#	*"not equal" relational operator.*
[	*" equal to or greater than" relational operator.*
]	*"equal to or less than" relational operator.*

Figure A. F Code Operators

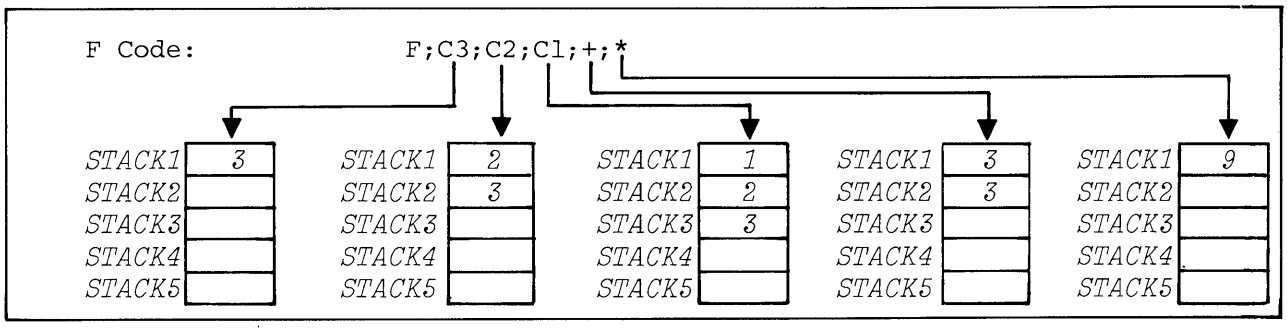
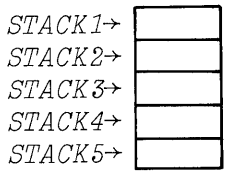


Figure B. Sample F Code and Associated Operations on Stack

4.11 DEFINING MATHEMATICAL FUNCTIONS: F

The F code is used to compute a value as a mathematical function on a defined set of attributes within one item.

All arithmetic operations specified by an F code operate on the last two entries in a push-down stack. This push-down stack has a maximum capacity of five (5) 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.

An F code is comprised of any number of operands or operators in reverse Polish format separated by semicolons. When the function processor encounters an operand specification (i.e., 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.

The general form of the F code is as follows:

```
F;element;element;element...
```

An "element" may be any of the following: a numeric AMC specifying an attribute value to be pushed onto the stack, optionally followed by any conversion specification(s), enclosed in parentheses; a constant of the form Cn where n is a numeric constant to be pushed onto the stack; a D which specifies that the current date is to be pushed onto the stack; a T which specifies that the current time is to be pushed onto the stack; a special 2-character operand; or an operator which specifies an operation to be performed on the top two entries in the stack. The operators are listed in Figure A. 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 STACK 1 would contain a 1).

Figure B illustrates an example of an F code. This figure shows the contents of the stack as each element is processed.

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(U11F0);2(U21F0);+	*Calls user conversion routines to operate on data from attribute #1 and #2, then adds the values.*
F;D(D2/]G2/1);3(D2/]G2/1);-	*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

4.12 F CODE SPECIAL OPERANDS

F code operands may be multi-valued, may contain conversion specification(s), or may be a special 2-character operand specifying one of several counters. Different interpretations are given to a F correlative vs. F conversion for an attribute with a TOTAL modifier.

Attribute operands may be multi-valued. When arithmetic operations are performed on two multi-valued lists (vectors), the answer will also be multi-valued 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 the attribute with AMC-15 had a value of "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 multi-valued 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 subvalue marks (shift-contol-1's). Some examples are given in Figure A.

Special 2-character operands may be used as F code elements, as listed in Figure B. For example.

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

On every detail line, this returns the value from AMC3; on every Break line (including the grand-total line), the average value of the data in attribute 3 is returned.

The Function code operates in two different fashions on an attribute with a TOTAL modifier, depending on whether it is specified as a correlative or as a conversion. As a correlative, the function is applied before the accumulation of the total, and is ignored on the Break data line; therefore, the total of the functioned value is computed. As a conversion, the function is ignored on detail lines, and is applied only on the Break data line and other subtotal fields in the output. Therefore, the function of other totalled values is obtained. If the function is specified as a conversion, the numeric operand (AMC) in the Function code must correspond to an attribute that is being totalled *within the statement*. If such an attribute does not exist, a value of zero is returned. Note that the numeric operators may be dummy AMC's in that they may reference other attributes within the statement that have function correlatives.

ELEMENT	DESCRIPTION	ACTION
ND	detail line counter	Push numeric value representing number of detail lines since last control-break: counter → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
NV	multi-value	Push numeric value representing current multi-value counter: counter → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
NS	sub-multi-value counter	Push numeric value representing current sub-multi-value counter: counter → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
=	equal	1) If STACK1 = STACK2 then 1 → STACK1 2) If STACK1 ≠ STACK2 then 0 → STACK1 3) In either case STACK5 → STACK4 → STACK3 → STACK2
≠	not equal	1) If STACK1 ≠ STACK2 then 1 → STACK1 2) If STACK1 = STACK2 then 0 → STACK1 3) In either case STACK5 → STACK4 → STACK3 → STACK2
<	less than	1) If STACK1 < STACK2 then 1 → STACK1 2) If STACK1 not < STACK2 then 0 → STACK1 3) In either case STACK5 → STACK4 → STACK3 → STACK2
>	greater than	1) If STACK1 > STACK2 then 1 → STACK1 2) If STACK not > STACK2 then 0 → STACK1 3) In either case STACK5 → STACK4 → STACK3 → STACK2
[	greater than or equal to	1) IF STACK1 [STACK2 then 1 → STACK1 2) If STACK1 not STACK2 then 0 → STACK1 3) In either case STACK5 → STACK4 → STACK3→ STACK2
]	less than or equal to	1) If STACK1] STACK2 then 1 → STACK1 2) If STACK1 not STACK2 then 0 → STACK1 3) In either case STACK5 → STACK4 → STACK3 → STACK2

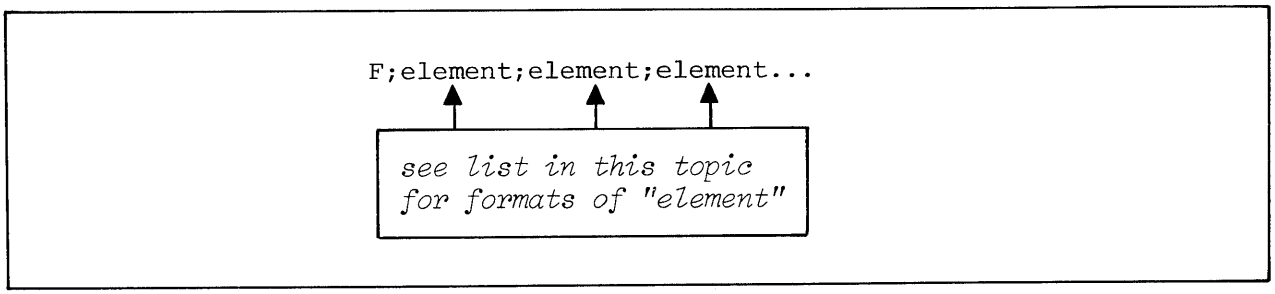


Figure A. General Form of F Code

4.13 SUMMARY OF F CODE STACK OPERATIONS

This topic presents a summary of the F code stack operations. The user should note that the symbol STACK1 → STACK2 means that the content of STACK1 (the top of the stack) is pushed down to position STACK2.

ELEMENT	DESCRIPTION	ACTION
amc	attribute	Push corresponding attribute value onto push-down stack (maximum five levels): attribute value → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
amc (conversion)	attribute with conversion	Push corresponding attribute value, after conversion, onto push-down stack: attribute value → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
Cn	constant	Push numeric constant "n" onto stack: n → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
+	add	STACK2 + STACK1 → STACK1, STACK5 → STACK4 → STACK3 → STACK2
-	subtract	STACK1 - STACK2 → STACK1, STACK5 → STACK4 → STACK3 → STACK2
*	multiply	STACK1 * STACK2 → STACK1, STACK5 → STACK4 → STACK3 → STACK2
/	divide	STACK1 / STACK2 → STACK1, STACK5 → STACK4 → STACK3 → STACK2
R	remainder	REMAINDER(STACK1/STACK2) → STACK1, STACK5 → STACK4 → STACK3 → STACK2
S	sum	Σ(STACK1) → STACK1 Prior to this operation, STACK1 may be multi-valued; this operator sums all those multi-values into a single value.
←	exchange	STACK2 ←→ STACK1
D	date	Push numeric value representing current system date onto stack: date → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
T	time	Push numeric value representing current system time onto stack: time → STACK1 → STACK2 → STACK3 → STACK4 → STACK5
NI	item counter	Push numeric value representing current item counter onto stack: counter → STACK1 → STACK2 → STACK3 → STACK4 → STACK5

ACCOUNT...	NAME.....	ADDRESS.....	CURR-BALNC..	BILL-RATE
23095	W E ZUMSTEIN	224 BEGONIA		1
23100	G J PACE	218 BEGONIA	$ 9,562.24	10.03
23105	B G PAUL	212 BEGONIA	$ 1,123.47	23
23110	J L VANGOTHEN	206 BEGONIA	$ 47,452.93	35
23115	T F PIATKOSKI	200 BEGONIA	$ 45,678.22	35
35000	J L DIESEM	300 COVE STREET	$ 112.37	35
35005	J S ROWE	306 COVE STREET	$ 464.72-	35
35010	S R KURTZ	312 COVE STREET	$ 467.33	10.03
35015	W F GRUNBAUM	318 COVE STREET	$ 88.47	35
35025	J D GOETZINGER	330 DOVE STREET	$ 3.45	35
35030	F M HUGO	301 DAHLIA	$ 123.48	35
35035	M J LANZENDORPHER	307 DAHLIA	$ 445.89	35
35040	C E ESCOBAR	313 DAHLIA	$ 88,822.12-	35
35050	P J WATT	325 DAHLIA	$ 337.18	10.03
35055	J W ROMEY	331 DAHLIA	$ 33,476.25	35
35060	J A SCHWARTA	331 DOCK WAY	$ 33,822.34	2
35065	L J RUFFINE	325 DOCK WAY	$ 558.43	10.03
35070	F R SANBORN	319 DOCK WAY	$ 22,144.67	35
35075	J L CUNNINGHAM	313 DOCK WAY	$ 7.70	40
35080	G A BUCKLES	307 DOCK WAY	$447,765.48	35
35085	J F SITAR	301 DOCK WAY	$ 200.00	2
35090	D U WILDE	330 CARNATION	$ 884.53	
35095	A W FEVERSTEIN	324 CARNATION	$ 19.25	35
35100	R W FORSTROM	318 CARNATION		10.03
35105	S J FRYCKI	312 CARNATION	$ 5,569.53	35
35110	H E KAPLOWITZ	306 CARNATION	$ 94,944.55	10.03

67 ITEMS LISTED.

APPENDIX A

THE ACCOUNT FILE USED IN EXAMPLES

As an aid to understanding the numerous examples in this manual which deal with 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 CURR-BALNC BILL-RATE (CR)

PAGE 1                                                    15:07:41  21 AUG 1976

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

11000       M H KEENER         100 ANCHOR PL     $  2,246.78       10.03
11015       L K HARMAN         118 ANCHOR PL     $      8.60          30
11020       J T O'BRIEN        124 ANCHOR PL     $306,755.54          30
11025       P R BAGLEY         130 ANCHOR PL     $ 82,045.33       10.03
11030       F E CABRON         101 BEGONIA       $     20.50       10.03
11035       R S MARCUS         107 BEGONIA       $ 23,911.14       10.03
11040       E G MCCARTHY       113 BEGONIA       $    334.56          30
11045       F R DRESCH         119 BEGONIA       $  1,199.46       10.03
11050       J R MARSHECK       125 BEGONIA                          30
11055       W H KOONS          131 BEGONIA       $958,343.75       10.03
11060       F T NATORI         131 BAY STREET    $     34.86        3333
11065       C V RANDALL        125 BAY STREET    $    552.13       10.03
11070       A A ALTHOFF        119 BAY STREET    $     22.60       10.03
11075       T F LINDSEY        113 BAY STREET    $     13.10       10.03
11080       E M AWAD           107 BAY STREET    $  2,937.35       10.03
11085       A B SEGUR          101 BAY STREET    $    224.55-         30
11090       J W JENKINS        130 AVOCADO       $  2,224.84          30
11095       J B STEINER        124 AVOCADO       $      3.83          30
11100       E F CHALMERS       118 AVOCADO       $     17.50          40
11105       C C GREEN          112 AVOCADO                          30
11110       D L WEISBROD       106 AVOCADO       $    484.84          30
11115       D R MASTERS        100 AVOCADO       $      9.20          30
21780       E W AWAD           107 BAY STREET    $  9.932.22       10.03
23000       H T LEE            200 BAY STREET    $ 12,332.11          35
23005       W B THOMPSON       206 BAY STREET    $ 11,265.21       10.03
23010       W E MCCOY          212 BAY STREET    $ 28,464.64          35
23015       R M COOPER         218 BAY STREET    $    385.56          35
23020       S L UNGERLEIDER    224 BAY STREET    $    983.34       10.03
23025       D C BINGAMAN       230 BAY STREET    $     18.70          35
23030       L J DEVOS          201 CARNATION     $    484.94          30
23035       G A BORDEN         207 CARNATION     $  9,663.72          35
23040       P B SCIPMA         213 CARNATION     $123,432.22          35
23045       P F KUGEL          219 CARNATION     $ 99.422.34          35
23050       E G MCCARTHY JR    225 CARNATION     $     44.88          35
23055       S M NEWMAN         231 CARNATION     $  7,452.92          35
23060       S I SZABO          231 COVE STREET   $ 54,668.13          35
23065       J A WOSK           225 COVE STREET   $  8,563.36          35
23070       L R MARCHANT       219 COVE STREET   $    345.12           1
23075       M J SADOSKI        213 COVE STREET   $  1,124.75          35
23080       J W YOUNG          207 COVE STREET   $     89.32       10.03
23090       W J HIRSCHFIELD    230 BEGONIA       $     20.45          35
```

DECIMAL	HEX	CHARACTER	SPECIAL USE IN REALITY
48	30	0	
49	31	1	
50	32	2	
51	33	3	
52	34	4	
53	35	5	
54	36	6	
55	37	7	
56	38	8	
57	39	9	
58	3A	:	
59	3B	;	
60	3C	<	
61	3D	=	
62	3E	>	
63	3F	?	
64	40	@	
65	41	A	
66	42	B	
67	43	C	
68	44	D	
69	45	E	
70	46	F	
71	47	G	
72	48	H	
73	49	I	
74	4A	J	
75	4B	K	
76	4C	L	
77	4D	M	
78	4E	N	
79	4F	O	
80	50	P	
81	51	Q	
82	52	R	
83	53	S	
84	54	T	
85	55	U	
86	56	V	
87	57	W	
88	58	X	
89	59	Y	
90	5A	Z	
91	5B	[	
92	5C	\	
93	5D	]	
94	5E	¬	
95	5F	_	
123	7B	{	
124	7C	:	
125	7D	}	
126	7E	~	
127	7F	DEL	
251	FB	SB	Start Buffer
252	FC	SVM	Secondary Value Mark
253	FD	VM	Value Mark
254	FE	AM	Attribute Mark
255	FF	SM	Segment Mark

APPENDIX B

LIST OF ASCII CODES

This appendix presents a list of ASCII codes used in the Reality system.

DECIMAL	HEX	CHARACTER	SPECIAL USE IN REALITY
0	0	NUL	Null prompt character
1	1	SOH	Cursor home on CRT Terminal
2	2	STX	
3	3	ETX	
4	4	EOT	
5	5	ENQ	
6	6	ACK	Cursor forward on CRT Terminal
7	7	BEL	Bell on CRT Terminal
8	8	BS	
9	9	HT	
10	A	LF	Cursor down on CRT Terminal
11	B	VT	Vertical address on CRT Terminal
12	C	FF	Screen erase on CRT Terminal
13	D	CR	Carriage return on CRT Terminal
14	E	SO	
15	F	SI	
16	10	DLE	Horizontal address on CRT Terminal
17	11	DC1	
18	12	DC2	
19	13	DC3	
20	14	DC4	
21	15	NAK	Cursor back on CRT Terminal
22	16	SYN	
23	17	ETB	
24	18	CAN	
25	19	EM	
26	1A	SUB	Cursor up on CRT Terminal
27	1B	ESC	
28	1C	FS	
29	1D	GS	
30	1E	RS	
31	1F	US	
32	20	SPACE	
33	21	!	
34	22	"	
35	23	#	
36	24	$	
37	25	%	
38	26	&	
39	27	'	
40	28	(	
41	29	)	
42	2A	*	
43	2B	+	
44	2C	,	
45	2D	-	
46	2E	.	
47	2F	/	

(THIS PAGE LEFT BLANK INTENTIONALLY)

APPENDIX C

SPECIAL CONTROL CHARACTERS

This appendix describes special control characters. Control characters are represented with a superscript of "c" for control and "s" for shift (e.g., O^{cs} represent control-shift-O, H^{c} represent control-H, etc.).

CONTROL CHARACTER	FUNCTION
H^{c}	Backspace: deletes last character typed in and allows user to re-enter character.
X^{c}	Cancel: deletes the entire line currently being typed in.
R^{c}	Retype: causes entire line currently being typed in to be re-typed.
O^{cs}	Line Continuation: If typed as the last character on a line, will allow continuation of the line onto the next physical line. This character must be immediately followed by a carriage return or line feed.

Microdata Corporation

17481 RED HILL AVENUE, IRVINE, CALIFORNIA 92714 • TELEPHONE: 714/540-6730 • TWX: 910-595-1764

PRINTED IN USA