MCDONNELL DOUGLAS

ENGLISH
Reference Manual

1400

ENGLISH
Reference Manual

February 1988

McDonnell Douglas Computer Systems Company 87-1400
Boundary Way
Hemel Hempstead
Hertfordshire
ENGLAND HP2 7HU McDonnell Douglas Information Systems Limited

Chapter 1 How to Use This Manual

Overview 1-3
Purpose of This Manual 1-3
Conventions 1-3
Use of Examples 1-4

Chapter 2 Using ENGLISH

Overview 2-3
Data Retrieval 2-3
Report Generation 2-3
Other Capabilities 2-4
General Form of ENGLISH Sentence 2-5
General Rules of ENGLISH 2-8

Chapter 3 ENGLISH Verbs

Overview 3-3
Summary of Verbs 3-3
BSELECT 3-5
  Multivalued Attributes 3-5
COPY-LIST 3-7
COUNT 3-9
DELETE-LIST 3-10
EDIT-LIST 3-11
  Multivalued Attributes 3-11
ESEARCH 3-12
FORM-LIST 3-14
GET-LIST 3-15
HASH-TEST 3-16
I-DUMP 3-17
  System Delimiters 3-17
ISTAT 3-18
LIST 3-19
  Sub-items 3-19
LIST-ITEM 3-21
LIST-LABEL 3-22
  Multivalued Attributes 3-23
S-DUMP 3-25
  System Delimiters 3-25
SAVE-LIST 3-26
SELECT 3-27
SORT 3-29
  Multivalued Attributes 3-29
  Associative Attributes 3-29
SORT-ITEM 3-31
SORT-LABEL 3-32
SORT-LIST 3-33
SSELECT 3-34
ST-DUMP 3-35
STAT 3-36
SUM 3-37
T-DUMP 3-38
T-LOAD 3-39

Chapter 4 **Data Selection**

Overview 4-3
Item Lists 4-3
  Sublists 4-3
Selection Criteria 4-4
String Searching 4-6
Relational Operators and Logical Connectives 4-8
Output Specification 4-10
  Multivalued Attributes 4-10
  Print Limiting 4-10
  Associative Attributes 4-10
TAPE Modifier 4-13
Throwaway Connectives 4-13

Chapter 5 **Report Presentation**

Overview 5-3
Sort-Order 5-3
Headings and Footings 5-6
  Heading 5-6
  Footing 5-6
  Heading and Footing Options 5-6
  Expanded Print 5-7
Generating Totals 5-9
  Labelling the Total 5-9
Breaking on Attribute Values 5-11
  Generating Sub-totals 5-12
Miscellaneous Modifiers 5-14

Chapter 6 **Dictionary Files**

Overview 6-3
Attribute Definition Items 6-4
  Structure 6-4
Default Attributes 6-6
Item Size 6-6
Dictionary File Creation 6-10
Attribute Definition Item Creation 6-10

Chapter 7 **Correlatives and Conversions**

Overview 7-3
Multiple Correlatives/Conversions 7-3
Correlative or Conversion? 7-4
  BREAK-ON Modifier 7-4
  TOTAL Modifier 7-5
Correlative and Conversion Codes 7-6
Algebraic Functions: A Code 7-7
  Embedded Decimal Points: 'An' Format .. 7-7
  'An; expression' Format 7-8
  Operands 7-8
  Conversions 7-9
  Operators 7-10

Special Functions 7-12
Concatenation: C Code 7-14
Date Formatting: Date Codes 7-17
Associative Attributes: D1 and D2 Codes . 7-20
Mathematical Functions: F Code 7-22
Operands 7-23
Operators 7-23
F Code Stack 7-24
F Code Special Operands and 'Fn' Format 7-26
Embedded Decimal Points: 'Fn;' Format . 7-28
Summary of F Code Stack Operations 7-28
Group Extraction: G Code 7-31
Mask Character: MC Codes 7-32
Mask Decimal: MD Code 7-34
Mask Field: MF Code 7-36
Mask Packed: MP Code 7-39
Time Formatting: MT Code 7-40
Mask Hexadecimal: MX Code 7-41
Text Extraction: T Code 7-42
File Translation: Tfile Code 7-43
Sublists: V Code 7-47

Appendix A ASCII, Hexadecimal and Decimal Table A-1

Index Index-1

Chapter 1
How to Use This Manual

Overview 1-3
Purpose of This Manual 1-3
Conventions 1-3
Use of Examples 1-4

Overview

The ENGLISH Retrieval Language provides a simple yet powerful means of accessing the system database. It enables the output of all selected information according to a user specification and to a largely user-specified format, thus enabling reports to be produced.

Purpose of This Manual

To explain how to use ENGLISH and also how to set up the necessary elements to enable ENGLISH to be used.

Command Language

The entry of commands for retrieval of information from your database is described in chapters 2 to 5 of this manual. Another manual, the Beginner's Guide to ENGLISH, provides an introduction to ENGLISH with practical examples which you may find useful, especially if you are unfamiliar with the use of computers.

Dictionaries

Dictionaries are described in chapters 6 and 7 of this manual. As a user of the command language you do not need to know how to create or design dictionaries. However, you may find it useful to know something about what they can do in order to get the best out of the command language.

Conventions

The following conventions apply to this manual:

Convention	Meaning
TEXT	Boldface text represents your input. . Characters printed in UPPER CASE must be entered; type them as shown. . Characters or words printed in lower case are parameters that you supply (for example, when you see file-name, item-id, or data, fill in the actual name of your file, the actual item-id, or your own data). New terms appear in boldface when first defined. Commands appear in boldface when first mentioned.
{ }	Braces indicate optional parameters.

{ }...	If an ellipsis (three dots) follows the terminating brace, then the enclosed word or parameter may be omitted or repeated an arbitrary number of times.
[ ]	Brackets specify that you must enter one of the enclosed parameters.

After you type a command or answer a system prompt, press the RETURN key unless otherwise instructed.

Use of Examples

Many of the examples in this document use the Hotel database which was set up for use with the Beginner's Guide to the ENGLISH Retrieval Language. This database comprises the ROOMS and the GUESTS files, both of which have item-ids that are room numbers.

Dates are displayed on your screen in American (MM/DD/YY) or International (DD/MM/YY) format depending on your system/terminal setting. Similarly the currency sign displayed on your screen may be $, £ or other. In this manual dates are shown in international format and costs are shown predominantly in $ as these are the most widely recognised forms.

Chapter 2
Using ENGLISH

Overview 2-3
Data Retrieval 2-3
Report Generation 2-3
Other Capabilities 2-4
General Form of ENGLISH Sentence 2-5
General Rules of ENGLISH 2-8

Overview

ENGLISH is a terminal command language for the retrieval of data from a database. Data can be presented as reports, labels, counts, lists and sums which may be output to terminal, printer or tape, or saved for editing or later use.

DATA RETRIEVAL

Within a file the pieces of data to be selected can be specified in a number of ways. If you know exactly the data you require you can simply specify the item-ids (record names) and attribute names (fields) you have selected. However, ENGLISH also allows you to specify conditions that an item must satisfy in order to be selected. You might want to retrieve all items satisfying one or more of the following conditions:

. Those with specific attribute values (for example, in the HOTEL database example used throughout this manual, rooms with King or Queen size beds).

. Those with a particular attribute (or item-id) within a range of values (for example, rooms costing between 60 and 70 dollars a night).

. Those with a particular attribute (or item-id) starting with, containing, or ending with a particular string of characters (for example, rooms with guest names starting with 'Fen').

. Those without a value in a particular attribute (for example, rooms without a guest).

REPORT GENERATION

Selected data is automatically output in the form of a report if you use a SORT or LIST verb. The format of the report can be modified to suit your requirements in a number of ways including:

. By specification of headings and footings.

. By generation of totals of numerical attributes.

. By grouping items with similar attribute values.

. By suppression of default information such as time, date, item-id and page numbers (which otherwise appear in every report).

. By sorting output on the numerical or alphabetical order of one or more attributes.

This is only a brief summary of some of the facilities most often used in producing reports.

OTHER CAPABILITIES

ENGLISH has powerful processing capabilities which enable generation of statistics from information in the database, printing of address and other labels, counting, sorting and summing of data and generation of lists and reports which can be saved in various ways for later use.

GENERAL FORM OF ENGLISH SENTENCE

A verb must always be the first word of any ENGLISH sentence. The remaining elements, however, may be specified in almost any order.

The general form of an ENGLISH sentence can be given as:

verb {**DICT**} **file-name** {**item-list**} {**sel-criteria**} {**sort-order**} {**o/p-spec**} {**o/p-modifier**} {(**options**)}

where:

verb

is any of the ENGLISH verbs, each of which indicates a particular operation.

For example, LIST, SORT, SELECT, SUM.

DICT

specifies that the verb is to operate on the dictionary rather than the data file.

file-name

identifies the file containing the data to be accessed.

For example, the hotel ROOMS file contains information about each of the rooms in the hotel, such as room size, rate, bed type.

item-list

contains one or more item-ids, each enclosed in single quote marks. This limits processing to these items only. An item-list may also be created by an item-list selection criterion containing relational operators and logical connectives. If item-list is omitted, all items are implied, unless an item list is supplied by an immediately preceding SELECT, SSELECT, or equivalent command.

For example, the item-list

'117''354''535''127'

applied to the ROOMS file specifies rooms 117, 354, 535 and 127.

sel-criteria

limit processing to items satisfying these criteria. If item-list is also included, processing is restricted to selected items that also satisfy the selection criteria.

For example, the sel-criterion

WITH BED-CODE = "WB"

applied to the ROOMS file selects rooms that have a waterbed.

sort-order

modifies the order of output of items.

For example,

BY-DSND LEAVE-DATE

applied to the ROOMS file outputs selected items in descending order of LEAVE-DATE.

o/p-spec

comprises the names of one or more attributes that are to be output. If omitted, any attributes defined as 'default attributes' by the dictionary file are output.

For example, the o/p-spec

GUEST-NAME LEAVE-DATE RATE

applied to the ROOMS file specifies that these three attributes are to be output for each selected item.

o/p-modifier

generally changes the output in some way. More than one modifier may be included, anywhere in the sentence.

For example,

HEADING "GUEST LIST"

puts the heading 'GUEST LIST' at the top of every page of the report.

options

also qualify the output in some way. The following options apply to most verbs; additional options are specified with the appropriate verbs. The closing bracket is optional if the option(s) is specified at the end of the sentence.

Option	**Explanation**
P	Directs the report to the spooler (normally to be printed)
N	Suppresses automatic paging at a terminal.

GENERAL RULES OF ENGLISH

The following general rules apply to the use of ENGLISH:

- ENGLISH sentences are entered at the TCL colon prompt (:) or the TCL+ prompt (+:).

- The first word of any ENGLISH sentence must be an ENGLISH verb.

- A sentence is usually terminated by one press of the RETURN key. However, a sentence longer than 140 characters may be created by entering a segment mark (CTRL _) followed by RETURN after entering 140 characters or fewer. This can be repeated as necessary.

- Each sentence must include just one file name. The modifier 'DICT' may be included anywhere in the sentence (normally just preceding the file-name) to specify operation on the dictionary rather than the data file.

- Any number of attributes may be named in a sentence. Generally attributes included in a report are shown across the page in the order they are specified in the ENGLISH sentence. If, however there is insufficient room they are shown down the page. Attributes must be defined in the dictionary of the referenced file.

- Each element in the sentence must be followed by a blank, if not followed by a quote or double-quote sign.

- Specific item-ids, and values used for selection if item-ids, are enclosed within single quotes.

- Specific values for comparison with attributes are enclosed within double quotes, and apply to the previous attribute name.

Overview 3-3
Summary of Verbs 3-3
BSELECT 3-5
  Multivalued Attributes 3-5
COPY-LIST 3-7
COUNT 3-9
DELETE-LIST 3-10
EDIT-LIST 3-11
  Multivalued Attributes 3-11
ESEARCH 3-12
FORM-LIST 3-14
GET-LIST 3-15
HASH-TEST 3-16
I-DUMP 3-17
  System Delimiters 3-17
ISTAT 3-18
LIST 3-19
  Sub-items 3-19
LIST-ITEM 3-21
LIST-LABEL 3-22
  Multivalued Attributes 3-23
S-DUMP 3-25
  System Delimiters 3-25
SAVE-LIST 3-26
SELECT 3-27
SORT 3-29
  Multivalued Attributes 3-29
  Associative Attributes 3-29
SORT-ITEM 3-31
SORT-LABEL 3-32
SORT-LIST 3-33
SSELECT 3-34
ST-DUMP 3-35
STAT 3-36
SUM 3-37
T-DUMP 3-38
T-LOAD 3-39

Overview

This section describes the ENGLISH verbs which specify the type of action to be taken. A summary of these verbs is given in Table 3-1 and this is followed by detailed descriptions of the verbs in alphabetical order.

SUMMARY OF VERBS

Table 3-1: ENGLISH Verbs

Verb	Function
LIST, SORT	Generate formatted output. LIST simply lists the selected output, while SORT orders the output in ascending order of item-id. A BY modifier may be included with either verb to sort the selected items in descending order of item-id or ascending/descending order of a specified attribute.
LIST-ITEM, SORT-ITEM	List the attributes of the specified items. SORT-ITEM orders the output in ascending order of item-id. A BY modifier may be included as with LIST and SORT.
LIST-LABEL, SORT-LABEL	These are analogous to LIST and SORT except that they arrange the output into a label format (suitable for address labels).
COUNT	Outputs a count of the number of items meeting the specified conditions.
SUM, STAT	Generate a total sum for the specified attribute. STAT also generates an average and a count of the number of items.
SELECT, SSELECT	Form a list of item-ids of items which satisfy the specified conditions (SSELECT forms a sorted list). This list is then made available to the next entered statement.
ESEARCH	Searches selected items for any occurrence, or non-occurrence, of a string or strings and forms a SELECT list of those items which satisfy the search criteria.
BSELECT	Forms a list of data satisfying specified conditions. This list is then made available to the next entered statement.

Table 3-1: ENGLISH Verbs (cont'd)

Verb	**Function**
SAVE-LIST, GET-LIST, SORT-LIST, EDIT-LIST, COPY-LIST, DELETE-LIST	Used to save, retrieve, sort, edit, copy and delete item-lists created using the SELECT, SSELECT, BSELECT, ESEARCH and FORM-LIST verbs.
FORM-LIST	Functions like GET-LIST except that an item in a user-specified file is the source of the item list.
T-DUMP, I-DUMP	Dump files to magnetic tape or the terminal.
T-LOAD	Loads files from magnetic tape.
ISTAT, HASH-TEST	ISTAT produces file hashing statistics and an optional histogram for selected items in a file. HASH-TEST is similar except that it bases the results on a user-specified test modulo.

BSELECT

Function	BSELECT forms a list of data specified by an item-list and attribute name(s). Any processing that was specified when the attribute name(s) was defined is carried out. The list is then made available for use by the next statement entered at the special '>' prompt.
Syntax	**BSELECT file-name {item-list} {sel-criteria} {o/p-spec}**
Command Class	ENGLISH verb.

Description

The number of items selected is displayed followed by a special prompt character '>'. The statement entered at this prompt can use the newly-formed list and may be one of the following:

- Where the data selected is actually a set of item-ids for a second file, an ENGLISH statement can be entered without an item-list.
- A catalogued DATA/BASIC program name. The selected items are available to the DATA/BASIC program via the READNEXT statement (see the DATA/BASIC Reference Manual for further details).
- Where the data selected is actually a set of item-ids for a second file, a TCL-II statement, such as COPY, can be entered without an item-list (see the TCL Commands Reference Manual for details of TCL-II statements).
- SAVE-LIST to save the list under a user-specified name so that it can be recalled for use as required.

When a BSELECT is processed within a PROC the selected items may be accessed by a variety of PROC commands (see the PROC Reference Manual for further details).

Multivalued Attributes

Each value (and sub-value) within the selected data is treated as a separate attribute and becomes a separate list element.

Example

```
:BSELECT ROOMS WITH BED-CODE = "Q" GUEST-NAME
 LEAVE-DATE

6 ITEMS SELECTED
>SAVE-LIST QBED

[241] 'QBED' CATALOGED; 1 FRAMES USED

:EDIT-LIST QBED
TOP
.L99
001  Lynch
002  21/04/87
003  Hennessey
004  30/04/87
005  Irving
006  24/04/87
EOI  006
.EX
'QBED' EXITED
:
```

COPY-LIST

Function COPY-LIST copies a saved item-list to another list name (and/or account name). It can also be used to copy the item-list to a normal file item.

Syntax **COPY-LIST {name {account-name}} {(options}**

Command Class TCL-I verb.

Description When the verb has been entered the system prompts:

TO:

at which you should enter a new list name and/or a different account name in the form:

{name {account-name}}

Alternatively, if the item-list is to be copied to become an item in a file, you must enter the file and item in the form:

({DICT} file-name) {item-id}

where if an item-id is not specified then an item-id of the same name as the list being copied is assumed. The size of the item-list in this case must not exceed 32K.

The options which may be used with the COPY-LIST verb are the same as those available with the COPY verb, which are:

Option	Explanation
D	Delete the source list after it has been copied. (Not valid with P or T.)
L	Suppress line numbers. (Valid only with T.)
N	Suppress auto paging. (Valid only with T.)
O	Overwrite destination file items with source file items if the item-ids are the same.
P	Copy to printer.
S	Suppress 'item-id' NOT COPIED message.
T	Copy to terminal.

X	Output in hexadecimal format. (Valid only with T or P.)

Example 1

```
:COPY-LIST VACANT
TO: EMPTY
```

Copies the item-list called VACANT to a new list called EMPTY; both lists then exist on the system.

Example 2

```
:COPY-LIST EMPTY
TO: (TEST.FILE) SP.ROOMS
```

Copies the item-list called EMPTY to an item called SP.ROOMS in TEST.FILE.

Example 3

```
:COPY-LIST VACANT
TO: EMPTY

[241]'EMPTY' CATALOGED; 1 FRAMES USED
```

Example 4

```
:COPY-LIST EMPTY (T

EMPTY
001 186
002 329
003 321
004 179
005 600
:
```

COUNT

Function	COUNT outputs a count of the number of items in a file meeting any conditions specified. The maximum count is 2,147,483,647.
Syntax	**COUNT** {**DICT**} **file-name** {**item-list**} {**sel-criteria**} {(**options**}

<u>Option</u>	<u>Explanation</u>
C{n}	Running counters of the number of items processed and items selected are to be displayed as selection progresses. In order that the ENGLISH process is not slowed down too much, the counters are only updated every 500 items processed. This number can be overridden by a number specified immediately following the C.
P	Copy to printer.

Command Class	ENGLISH verb.
Sub-items	If, when the file was set up, an item was specified as having sub-items (via a sublist) then these can be included in the count by preceding the file-name with the word WITHIN. One, and only one, item-id (that of the master item) must then be specified.

Example 1

```
:COUNT ROOMS

29 ITEMS COUNTED.
```

Example 2

```
:COUNT ROOMS WITH NO GUEST

5 ITEMS COUNTED.
```

Example 3

```
:COUNT GUESTS WITH BILL-TOTAL > "100"

18 ITEMS COUNTED.
```

Example 4

```
:COUNT ROOMS LE '400' WITH GUEST (C

18 ITEMS SELECTED             29 ITEMS PROCESSED
18 ITEMS COUNTED
```

Example 5

```
:COUNT WITHIN ASSEMBLIES 'A200'

8 ITEMS COUNTED
```

DELETE-LIST

Function	DELETE-LIST deletes a saved item list. Storage frames are returned to the system overflow space.
Syntax	**DELETE-LIST {name {account-name}}**
Command Class	TCL-I verb.
Description	A list generated and saved from another account can only be deleted if the deleting account has SYS2 privileges (as specified when the account was set up).

Example

```
:DELETE-LIST VACANT
[242] 'VACANT' DELETED
```

EDIT-LIST

Function	EDIT-LIST allows an item list saved by the SAVE-LIST verb to be modified. The EDITOR prompt '.' is displayed and any of the EDITOR commands can be used to add, delete or change item-ids.
Syntax	**EDIT-LIST** {**name** {**account-name**}} {(**Z**}

where:

Z

suppresses the initial 'TOP' message.

Command Class	TCL-I verb.
Multivalued Attributes	For lists created from multivalued data using an exploded sort, each line in the list contains an item-id followed by a value mark and a number. This number represents the position of the sorted value within the multivalue.

Example

```
:EDIT-LIST VACANT
TOP
.G4
004 179
.I             <------- EDITOR Insert Command
004+ 117
004+
.FS
TOP
.L99
001 186
002 329
003 321
004 179
005 117
006 600
EOF 6
.FI

[241]'VACANT' CATALOGED; 1 FRAMES USED
```

ESEARCH

Function	ESEARCH searches selected items in a file for any occurrence, or non-occurrence, of one or more character strings. It forms a SELECT list (list of item-ids) of those items which satisfy the specified search criteria. This list is then made available for use by the next statement entered at the special '>' prompt. (Note that item-ids are not searched.)
Syntax	**ESEARCH file-name {item-list} {sel-criteria} {(options}**

Option	Explanation
A	AND. Item must contain all specified strings in order to be selected.
I	List item-ids as they are selected.
L	Save the line numbers of lines containing the string(s). This results in a SELECT list comprising, for each occurrence of a specified string, an item-id followed by a value mark and the number of the attribute. This type of list would typically be used as input to DATA/BASIC.
N	NOT. Select those items already selected by the combination of 'item-list' and 'sel-criteria' (else all items in the file) which do not contain the specified string(s).
S	Suppress formation of a select list and simply display the item-ids of items as they are selected.

Command Class ENGLISH verb.

Description There is a prompt for the string(s) to be searched for:

```
STRING :AS
STRING :IT
  .
  .
  .
```

until just RETURN is entered in response. There is a limit on the total length of the input strings of 500 bytes. Unless the A or N option is present, if any one (or more) of the specified strings is found in an item it is selected.

The number of items selected is displayed followed by a special prompt character '>'. The statement entered at this prompt can be any of the following:

- An ENGLISH statement entered without an item-list.
- A catalogued DATA/BASIC program name. The selected items are available to the DATA/BASIC program via the READNEXT statement (see the DATA/BASIC Reference Manual for further details).
- A TCL-II statement, such as COPY, entered without an item-list (see the TCL Commands Reference Manual for details of TCL-II statements).
- SAVE-LIST to save the list of items under a user-specified name so that it can be recalled for use as required.

When an ESEARCH is processed within a PROC the selected items may be accessed by a variety of PROC commands (see the PROC Reference Manual for further details).

Example

```
:ESEARCH GUESTS (S

STRING: Henn
STRING:
143

1 ITEMS SELECTED
:
```

FORM-LIST

Function FORM-LIST forms an item-list from the contents of an item in a file. (Such an item is formed via the EDITOR, DATA/BASIC, PROC or the COPY-LIST verb.)

Syntax **FORM-LIST {DICT} file-name item-id {(n}**

where:

n

specifies that the list of item-ids formed is to start from attribute n instead of the first attribute.

Command Class TCL-II verb.

Example

```
:FORM-LIST TEST TRL

6 ITEMS SELECTED

>
```

GET-LIST

Function	GET-LIST retrieves a previously saved item-list and displays the '>' prompt. The effect of using this verb is therefore the same as if a SELECT or equivalent command had just been entered.
Syntax	**GET-LIST {name {account-name}}**
Command Class	TCL-I verb.
Description	If a list generated and saved by another user on another account is required then that account name must be specified. If 'name' is omitted then the list is saved as the default list for the current account, overwriting any previous default list.

Example

```
:GET-LIST VACANT

5 ITEMS SELECTED
>
```

HASH-TEST

Function

HASH-TEST produces file hashing statistics, and an optional histogram, for selected items in a file on the basis of a user-specified test modulo.

Syntax

HASH-TEST {**DICT**} **file-name** {**item-list**} {**sel-criteria**} {(**options**}

Option	**Explanation**
H	Output histogram
P	Copy to printer

Command Class

ENGLISH verb.

Description

After the HASH-TEST statement has been entered the system displays the prompt:

```
TEST MODULO:
```

The value entered here is used to produce the statistics/histogram.

Example

```
:HASH-TEST TEST (H

TEST MODULO: 7
FILE=ACCOUNT MODULO=7 SEPAR=1
BYTES ITMS
02413 010*>>>>>>>>>>
02710 011*>>>>>>>>>>>
02121 009*>>>>>>>>>
02598 011*>>>>>>>>>>>
01619 007*>>>>>>>
02462 010*>>>>>>>>>>
02660 010*>>>>>>>>>>

ITEM COUNT = 68, BYTE COUNT = 16583, AVG.
BYTES/ITEM = 243.8
AVG. ITEMS/GROUP = 9.7, STD DEVIATION = 1.3, AVG.
BYTES/GROUP = 2369
```

For further information see the manual Managing Accounts and Files.

I-DUMP

Function I-DUMP dumps dictionary or data files to the terminal.

Syntax **I-DUMP {DICT} file-name {item-list} {sel-criteria}**

Command Class ENGLISH verb.

System Delimiters System delimiters are displayed as follows:

Delimiter	Display
Attribute Mark	^
Value Mark	]
Sub-value Mark	\

Example

```
:I-DUMP ROOMS '179''186'

179^S,K^72.00^
186^S,K^72.00^

2 ITEMS DUMPED
:
```

ISTAT

Function ISTAT produces file hashing statistics, and an optional histogram, for selected items in a file.

Syntax **ISTAT {DICT} file-name {item-list} {sel-criteria} {(options}**

Option	Explanation
H	Output histogram.
P	Output to printer.

Command Class ENGLISH verb.

Example **:ISTAT TEST (H**

```
FILE= ACCOUNT MODULO= 3 SEPAR=1
BYTES ITMS
05379 023 *>>>>>>>>>>>>>>>>>>>>>>
05391 022 *>>>>>>>>>>>>>>>>>>>>>
05813 023 *>>>>>>>>>>>>>>>>>>>>>>

ITEM COUNT = 68, BYTE COUNT = 16583, AVG.
BYTES/ITEMS = 243.8
AVG. ITEMS/GROUP = 22.6, STD DEV = .5,
AVG. BYTES/GROUP = 5527.6
```

For further information see the manual Managing Accounts and Files.

LIST

Function LIST outputs information from a file according to the specification.

Syntax **LIST {DICT} file-name {item-list} {sel-criteria} {sort-order} {o/p-spec} {o/p-mod}...{(options}**

Command Class ENGLISH verb.

Description Output is in the form of columns across the page wherever possible. If, however, the total width of the columns is greater than the defined page width then a non-columnar output is generated down the page.

Sublists If, when the file was set up, an item was specified as having a sub-set of related items (via a sublist) then these can be included in the report by preceding the file-name with the word 'WITHIN'. One, and only one, item-id (that of the master item) must then be specified.

In a columnar listing a column five characters wide with a heading 'LEVEL' contains the level of item. In non-columnar listings the level number precedes each item.

(For further information on sublists see the section CORRELATIVES AND CONVERSIONS).

Example 1

```
:LIST ROOMS WITH ROOM-CODE "S" ROOM-TYPE RATE
GUEST-NAME LEAVE-DATE

PAGE 1

ROOMS Room.......   Rate..  Current....  Leave...
      Type                  Guest        Date

140   Single Occ.   68.00   Lynch        21/04/87
186   Single Occ.   72.00
143   Single Occ.   68.00   Hennessey    30/04/87
122   Single Occ.   64.00   Anderson     30/05/87
179   Single Occ.   72.00

5 ITEMS LISTED
:
```

Example 2

```
:LIST ROOMS WITH ROOM-CODE "S" ROOM-CODE
ROOM-TYPE BED-CODE BED-TYPE RATE GUEST-NAME
LEAVE-DATE
```

```
PAGE 1

ROOMS: 140
Room Code S
Room Type Single Occ
Bed Code Q
Bed Type Queen
Rate 68.00
Current Guest Lynch
Leave Date 21/04/87

ROOMS: 186
Room Code S
   :
   :
```

Example 3

```
:LIST WITHIN ASSEMBLIES 'A200' DESC SUB.ASSEM
STOCK
```

```
PAGE 1

LEVEL ASSEMBLIES DESCRIPTION...  SUB.ASSEM. STOCK

1     A200       SERVO           A201       74
                                 A202
                                 A203
2     A201       DC MOTOR        A210       35
                                 A211

3     A210       DC MOTOR PLTFM             23
3     A211       DC MOTOR P.U.              31
2     A202       SERVO BOARD                17
2     A203       SERVO HOUSING   A212       18
                                 A213
3     A212       HOUSING SEALS              32
3     A213       HOUSING PLATES             20

8 ITEMS LISTED
```

In this example items at level 2 are sublist items of the level 1 item and items at level 3 are sublist items of the preceding level 2 item.

LIST-ITEM

Function	LIST-ITEM lists the attributes, one per line, of the specified items in a file. The attributes are output in the same form as they are stored, without any correlatives or conversions being applied.
Syntax	**LIST-ITEM** {**DICT**} **file-name** {**item-list**} {**sel-criteria**} {**sort-order**} {**o/p-mod**}... {(**options**}
Command Class	ENGLISH verb.
Example	**:LIST-ITEM ROOMS**

```
   117
001 D,K
002 72.00
003 6321
004 6321

   194
001 D,D
002 64.00
003 6330
004 6330
   :
   :
   :
   :
```

LIST-LABEL

Function

LIST-LABEL is similar to LIST except that each item selected is output in the form of a label rather than a line in the report.

Syntax

LIST-LABEL {DICT} file-name {item-list} {sel-criteria} {sort-order} {o/p-spec} {o/p-mod}...{(options}

Command Class

TCL-II verb.

Description

After the command has been entered the system prompts for details of how the data is to be arranged.

```
COL,ROW,SKIP,INDNT,SIZ,SPACE(,C):
```

where:

COL

is the number of labels across the page

ROW

is the number of print lines per label. This must be a minimum of one for each attribute specified plus one for the item-id if ID-SUPP is not used.

SKIP

is the number of lines to skip between labels.

INDNT

is the number of spaces to indent the data on the left.

SIZ

is the maximum width of each label (anything longer than this will be truncated).

SPACE

is the number of spaces to skip between labels

C

is optional. If present specifies that null attributes are to be ignored and existing data is to be compressed. If not specified null attributes are treated as all blanks and a blank line is left in the label.

Note that the total page width implied by

```
INDNT + COL(SIZ + SPACE)
```

must not be greater than the page width.

If INDNT is non-zero the system prompts:

HEADER:

for a side-heading for the first line of label output; this is repeated for each line of label output. If headings are not required press just RETURN at each of the prompts.

Multivalued Attributes

Each multivalue in a multivalued attribute is treated as if it were a separate attribute and is output as a separate line of the label.

Note: If you execute LIST-LABEL or SORT-LABEL from a PROC and stack the parameters within that PROC then the prompts for COL, ROW, SKIP..... and HEADER will not be printed. This allows you to print labels on a slave printer.

Example 1

:LIST-LABEL GUESTS '117''119''144''147' NAME ADDRESS CITY STATE ID-SUPP COL-HDR-SUPP

```
COL,ROW,SKIP,INDNT,SIZ,SPACE(,C) :2,4,2,0,25,8,C

Loretta Rizzo                    Barry R. Scott
10 Webster St.                   90 Alpine St.
Harrington                       Harrison
TX                               CT

Mr. & Mrs. H. Irving             Loretta T. Janson
20 Thorpe Road                   23 Glenborn Av.
Lexington                        Los Angeles
FL                               CA
```

If ID-SUPP is not specified then item-ids are output as the first line of the label.

If COL-HDR-SUPP is not specified then the page number, time and date are printed at the top of each page. If COL-HDR-SUPP is specified, a continuous format without page breaks is produced.

Example 2

```
:LIST-LABEL GUESTS '117''119''144''147' NAME
ADDRESS CITY STATE ID-SUPP COL-HDR-SUPP

COL,ROW,SKIP,INDNT,SIZ,SPACE(,C) :2,4,3,10,25,5,C
HEADER :NAME
HEADER :ADDRESS
HEADER :CITY
HEADER :STATE

NAME      Loretta Rizzo          Barry P. Scott
ADDRESS   10 Webster St.         90 Alpine St.
CITY      Harrington             Harrison
STATE     TX                     CT

NAME      Mr. & Mrs H. Irving    Loretta T. Janson
ADDRESS   20 Thorpe Rd.          23 Glenborn Av.
CITY      Lexington              Los Angeles
STATE     FL                     CA
```

S-DUMP

Function S-DUMP sorts and dumps dictionary or data files to the terminal.

Syntax **S-DUMP {DICT} file-name {item-list} {sel-criteria}**

Command Class ENGLISH verb.

System Delimiters System delimiters are displayed as follows:

Delimiter	**Display**
Attribute Mark	^
Value Mark	]
Sub-value Mark	\

Example **:S-DUMP ROOMS > '500'**

```
535^P,WB^164.00^7142^7142^
600^ST,WB^104.00^
2 ITEMS DUMPED.
```

SAVE-LIST

Function If you anticipate that you will need a particular item list (created using SELECT, BSELECT, ESEARCH, SEARCH, SSELECT or FORM-LIST) more than once then you can save it. This will avoid you having to repeat the time consuming process of creating a list more than once.

Syntax **SAVE-LIST {name}**

Command Class TCL-I verb.

Description You must enter this command at the '>' prompt immediately after a SELECT, SSELECT, BSELECT, ESEARCH or equivalent statement.

Any existing list of the same name in the account in use is automatically overwritten by the new one. If 'name' is omitted then the list is saved as the default list for the current account, overwriting any previous default list.

A pointer to be saved list is stored in the POINTER-FILE as an item with an item-id 'account-name*L*name' (account-name is the name of the account from which the SAVE-LIST command was issued).

Example

```
:SSELECT ROOMS WITH NO GUEST

>SAVE-LIST VACANT

[241]'VACANT' CATALOGED; 1 FRAMES USED
```

SELECT

Function

SELECT forms a list of item-ids of items that satisfy the specified conditions. This list is then made available for use by the next statement entered at the special '>' prompt as described below.

Syntax

SELECT {**DICT**} **file-name** {**item-list**} {**sel-criteria**} {**sort-order**} {(**options**}

Option	Explanation
C{n}	Running counters of the number of items processed and items selected are to be displayed as selection progresses. In order that the ENGLISH process is not slowed down too much, the counters are only updated every 500 items processed. This number can be overridden by a number specified immediately following the C.
P	Copy to printer.

Command Class

ENGLISH verb.

Description

C specifies that running counters of the number of items processed and items selected are to be displayed as selection progresses. In order that the ENGLISH process is not slowed down too much, the counters are only updated every 500 items processed. This number can be overridden by a number immediately following the C.

The number of items selected is displayed followed by a special prompt character '>'. The statement entered at this prompt can use the newly-formed item list and may be one of the following:

- An ENGLISH statement entered without an item-list.
- A catalogued DATA/BASIC program name. The selected items are available to the DATA/BASIC program via the READNEXT statement (see the DATA/BASIC Reference Manual for further details).
- A TCL-II statement, such as COPY, entered without an item-list (see the TCL Commands Reference Manual for details of TCL-II statements).

- SAVE-LIST to save the list of items under a user-specified name so that it can be recalled for use as required.

When a SELECT is processed within a PROC the selected items may be accessed by a variety of PROC commands (see the PROC Reference Manual for further details).

Multivalued Attributes

For lists created from multivalued data using an exploded sort, each line in the list contains an item-id followed by a value mark and a number. This number represents the position of the sorted value within the multivalue.

Example 1

```
:SELECT ROOMS WITH BED-CODE "Q"

3 ITEMS SELECTED
>LIST GUESTS NAME

PAGE 1
GUESTS   Guest Name

140      Susan P. Lynch
143      William Hennessey
144      Mr. & Mrs. H. Irving

3 ITEMS LISTED
```

Example 2

```
:SELECT ROOMS WITH NO GUEST

5 ITEMS SELECTED
>SAVE-LIST EMPTY

[241]'EMPTY' CATALOGED; 1 FRAMES USED

:EDIT-LIST EMPTY
 TOP
 .L99
 001 186
 002 329
 003 321
 004 179
 005 600
 EOI 005
 .EX
 'EMPTY' EXITED
```

SORT

Function	SORT is similar to LIST except that, in the absence of a 'sort-order', it sorts the output into ascending order of the ASCII value of item-id, character by character, from left to right, (assuming left justified item-ids).
Syntax	**SORT {DICT} file-name {item-list} {sel-criteria} {sort-order} {o/p-spec} {o/p-mod}...{(options}**
Command Class	ENGLISH verb.
Description	If a descending sort of item-id is required then a BY-DSND sort-order modifier must be used followed by an attribute name which has been set up to describe attribute 0 (item-id).
Multivalued Attributes	It is sometimes convenient to store more than one value in a single attribute. For example, a hotel guest incurs multiple charges - for room, restaurant, bar, telephone, and so on. Each of the attributes BILL-CODE, BILL-DATE, BILL-AMOUNT and BILL-DESC are, therefore, multivalued with each value comprising a separate charge.
	If a SORT.... BY BILL-DATE is carried out then the items are output in ascending order of the dates of the first charge, with second and subsequent dates being listed under the first date.
	If each of the multivalues (and subvalues) is to be treated as a separate value then an exploded sort must be performed using the BY-EXP modifier.
Associative Attributes	The asterisks (*) under the headings 'Bill Date' and 'Amount' indicate that the dictionary defines the BILL-DATE and BILL-AMOUNT attributes as associated with the BILL-CODE attribute.
	The BILL-CODE attribute is the master attribute and must be included in any output-spec containing secondary associative attributes. If this is not done then the secondary attributes are not output.
	Also, as shown in the following examples, associative attributes are output in a pre-defined order which is not necessarily the order specified in the ENGLISH statement.
	(For further information see ASSOCIATIVE ATTRIBUTES in the section CORRELATIVES AND CONVERSIONS).

Example 1

```
:SORT ROOMS WITH NO GUEST ROOM-CODE ROOM-TYPE
 BED-TYPE RATE

PAGE 1

ROOMS  Room  Room.......  Bed.....  Rate....
       Code  Type         Type

179    S     Single Occ.  King         72.00
186    S     Single Occ.  King         72.00
321    DL    Deluxe       King         82.00
329    DL    Deluxe       WaterBed    104.00
600    ST    Suite        WaterBed    104.00

5 ITEMS LISTED
:
```

Example 2

```
:SORT GUESTS '117''119''144' BY BILL-DATE
 BILL-CODE BILL-DESC BILL-DATE BILL-AMOUNT

GUESTS  Bill  Bill Date  Amount..  Description...
        Code
              *          *
117        2   17/04/87    $62.00  Room
           6   20/04/87    $17.95  Dinner
          18   20/04/87    $8.76   Miscellaneous

144        4   17/04/87    $12.95  Breakfast
          15   20/04/87    $22.50  Telephone
          16   20/04/87    $12.05  Newspapers

119        2   21/04/87    $56.00  Room
          15   21/04/87    $23.98  Telephone
          13   21/04/87    $17.95  Bar (2)

3 ITEMS LISTED
```

Example 3

```
:SORT GUESTS '117''119''144' BY-EXP BILL-DATE
 BILL-CODE BILL-DESC BILL-DATE BILL-AMOUNT

GUESTS  Bill  Bill Date  Amount..   Description..
        Code
              *          *
117        2   17/04/87    $62.00   Room
144        4   17/04/87    $12.95   Breakfast
117        6   20/04/87    $17.95   Dinner
117       18   20/04/87    $8.76    Miscellaneous
144       15   20/04/87    $22.50   Telephone
144       16   20/04/87    $12.05   Newspapers
119        2   21/04/87    $56.00   Room
119       15   21/04/87    $23.98   Telephone
119       13   21/04/87    $17.95   Bar (2)
```

SORT-ITEM

Function	SORT-ITEM sorts and lists the attributes, one per line, of the specified items in a file. The attributes are output in the same form as they are stored, without any correlatives or conversions being applied.
Syntax	**SORT-ITEM {DICT} file-name {item-list} {sel-criteria} {sort-order}{o/p mod}...{(options}**
Command Class	ENGLISH verb.
Example	**:SORT-ITEM ROOMS**

```
   117
001 D,K
002 72.00
003 6321
004 6321

   194
001 D,D
002 72.00
003 6330
004 6330
  :
  :
```

SORT-LABEL

Function	SORT-LABEL is similar to LIST-LABEL except a SORT is also performed. (See LIST-LABEL)

SORT-LIST

Function	SORT-LIST performs a SORT on an item list previously saved by a SAVE-LIST statement.
Syntax	**SORT-LIST** {**name** {**account-name**}} If a list generated and saved by another user on another account is to be sorted then that account name must be specified. If 'name' is omitted then the default list for the current account is sorted.
Command Class	TCL-I verb.
Example	:**SORT-LIST VACANT** :

SSELECT

Function	SSELECT is similar to SELECT except that a SORT is also performed. (See SELECT).

ST-DUMP

Function	ST-DUMP sorts and dumps dictionary or data files to magnetic tape.
Syntax	**ST-DUMP {DICT} file-name {item-list} {sel-criteria} {(options}**

Option	Explanation
I	List item-ids as they are dumped.
T	Inhibit the writing of a tape label.

Command Class	ENGLISH verb.
Description	An EOF (end-of-file) mark is written to the tape at the end of the dump.

Example

```
:ST-DUMP ROOMS < '300'

15 ITEMS DUMPED.
```

==

STAT

Function	STAT generates a total sum, an average, and a count for the specified attribute.
Syntax	**STAT {DICT} file-name {item-list} attribute-name {sel-criteria} {(options}**
Command Class	ENGLISH verb.

Example 1

```
:STAT GUESTS BILL-TOTAL

STATISTICS OF BILL TOTAL:
TOTAL = $3,639.99 AVERAGE = $151.6662  COUNT = 24
```

Example 2

```
:STAT GUESTS BILL-TOTAL WITH BILL-TOTAL > "200"

STATISTICS OF BILL TOTAL:
TOTAL = $977.36 AVERAGE = $244.3400  COUNT = 4
```

SUM

Function	SUM generates a total sum for the specified attribute
Syntax	**SUM** {**DICT**} **file-name** {**item-list**} **attribute-name** {**sel-criteria**} {(**options**}
Command Class	ENGLISH verb.

Example 1

```
:SUM GUESTS BILL-TOTAL

TOTAL OF BILL TOTAL IS : $3,639.99
```

Example 2

```
:SUM ROOMS '318''365''329' RATE

TOTAL OF RATE IS : $312.00
```

T-DUMP

Function	T-DUMP dumps dictionary or data files to magnetic tape.
Syntax	**T-DUMP {DICT} file-name {item-list} {sel-criteria} {(options}**

Option	**Explanation**
I	List item-ids as they are dumped.
T	Inhibit the writing of a tape label.

Command Class	ENGLISH verb.
Description	An EOF (end-of-file) mark is written to the tape at the end of the dump.

Example

```
:T-DUMP ROOMS < '300'

15 ITEMS DUMPED
```

T-LOAD

Function	T-LOAD selectively loads dictionaries or data files from tape.
Syntax	**T-LOAD** {**DICT**} **file-name** {**item-list**} {**sel-criteria**} {(**options**}

Option	Explanation
O	Overwrite existing items if they have the same item-id.
S	Suppress listing of item-ids.

Command Class	ENGLISH verb.
Example	**:T-LOAD ROOMS WITH ROOM-CODE "S"**

Chapter 4
Data Selection

Overview 4-3
Item Lists 4-3
  Sub Lists 4-3
Selection Criteria 4-4
String Searching 4-6
Relational Operators and Logical
Connectives 4-8
Output Specification 4-10
  Multivalued Attributes 4-10
  Print Limiting 4-10
  Associative Attributes 4-10
TAPE Modifier 4-13
Throwaway Connectives 4-13

Overview

This section describes ways of limiting the data to be processed. These comprise item-lists, selection criteria and output specifications.

ITEM LISTS

An item list limits the items processed. If omitted, all items in the file are processed.

A simple item list comprises any number of item-ids, each enclosed in single quotes.

Another way to specify the items to be processed is to use relational operators and logical connectives to construct criteria for the selection of items. These criteria must immediately follow the file name and item-ids must be enclosed in single quotes.

Processing is faster when a simple item-list is used because only the specified items are accessed. In the case of a complex item-list (which contains relational operators and logical connectives) all items in the file are accessed for examination.

The precedence of logical connectives is AND then OR. The default logical connective is OR.

Sublists

If an item has been set up with sub-items (related items) then these can be specified in a LIST or COUNT sentence by preceding the file-name by the word WITHIN. The item-id of the main item (master item) only, must then be specified in the ENGLISH sentence.

See also STRING SEARCHING and RELATIONAL OPERATORS AND LOGICAL CONNECTIVES in this section, and SUBLISTS: V CODE in the section Correlatives and Conversions.

Example 1

LIST ROOMS BETWEEN '215' AND '245'

Specifies items with item-ids greater than 215 but less than 245.

Example 2

LIST ROOMS < '172' OR > '386'

Specifies items with item-ids less than 172 or greater than 386.

Example 3

LIST TST < 'A' OR > 'B' AND < 'C' OR > 'D' AND < 'E'

Specifies all items with ids less than A, or with ids greater than B but less than C, or with ids greater than D but less than E.

SELECTION CRITERIA

Selection criteria allow you to limit the items considered for output by specifying conditions which must be met by one or more attributes.

Syntax

WITH {NO} {EVERY} attribute-name {{op} value-list}

where:

WITH (or its synonym IF)

must be the first word.

EVERY (or its synonym EACH)

means that every value in a multivalued attribute must satisfy the specified condition in order for the item to be selected. If this is not specified then any one (or more) value(s) satisfying the condition causes the item to be selected.

value-list

specifies the conditions to be met by the attribute. It is formed in the same way as an item-list except that double quotes must surround the actual values. If a 'value-list' is omitted then the presence of a value in the specified attribute is required.

NO (or its synonym NOT)

specifies that the absence of a value in the attribute is required. NO and EVERY are mutually exclusive.

Another way to specify which items are to be considered is by making use of relational operators and logical connectives.

If a relational operator is not specified an 'equal to' operator is assumed.

Two or more selection-criteria may be joined by logical connectives to form the complete selection-criteria. In this case the AND connective has a higher precedence than the OR connective. It is imperative that when two or more selection-criteria are joined that each attribute name is preceded by the word WITH (or IF). If a logical connective is omitted, OR is assumed.

A complete selection-criterion may consist of up to nine 'AND clauses' where an 'AND clause' is made up of any number of individual selection-criteria joined by AND connectives; an 'AND clause' is terminated by 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 RATE = "72" AND WITH ROOM-CODE = "D") OR
(WITH RATE = "104" AND WITH ROOM-CODE = "ST")
```

See also STRING SEARCHING and RELATIONAL OPERATORS AND LOGICAL CONNECTIVES in this section.

Example 1

```
LIST ROOMS WITH ROOM-CODE "S"
```

selects all single rooms.

Example 2

```
LIST ROOMS WITH ROOM-CODE "D" OR "DL"
```

selects all double and deluxe rooms.

Example 3

```
LIST ROOMS WITH GUEST-NAME
```

selects all rooms with a guest.

Example 4

```
LIST ROOMS WITH NO GUEST-NAME
```

selects all rooms with no guest.

Example 5

```
LIST ROOMS WITH ROOM-CODE NE "D"
```

selects all non-double rooms, that is, single, penthouse, suite and deluxe rooms.

Example 6

```
LIST ROOMS WITH ROOM-CODE "D" OR "DL" AND WITH NO
GUEST-NAME
```

Selects all double and deluxe rooms without guests.

Example 7

```
LIST ROOMS WITH ROOM-CODE "D" OR WITH ROOM-CODE
"S" AND WITH BED-CODE "Q"
```

Selects all double rooms and those single rooms that have a Queen-size bed.

String Searching

You can also specify that in order for an item to be selected it's item-id, or the values of a particular attribute, must start with, contain, or end with a specified sequence of characters. There are two facilities which allow this and which may be combined if required, one using the up-arrow (^) character and the other using left and right square brackets ([and]) as follows:

^	is used as an 'ignore' or mask character. It specifies that the character in the corresponding position in the attribute value or item-id is to be ignored (that is, can be any character). You can only use this facility with left justified attributes.
[	placed to the left of a string of characters (within double quotes) signifies that the attribute value or item-id can be any number of characters ending with the string.
]	placed to the right of a string of characters (within double quotes) signifies that the attribute value or item-id can be any number of characters starting with the string.
[]	a combination of the above signifies that the attribute value or item-id can be any sequence of characters containing the string.

In order to search on an item-id an operator must be specified because an "=" operator is not assumed. If this is not specified then a message of the form '[19' NOT ON FILE is output. (This does not apply if item-id is defined and used in the search as an attribute.)

Example 1

```
:LIST ROOMS WITH GUEST-NAME "[son" GUEST-NAME

PAGE 1

ROOMS Current Guest..

142   Madison
122   Anderson
365   Ferguson
147   Janson

4 ITEMS LISTED
```

Example 2

```
:LIST ROOMS WITH GUEST-NAME "Men]" GUEST-NAME

PAGE 1

ROOMS Current Guest..

309   Mendell

END OF LIST
```

Example 3

```
:LIST ROOMS WITH GUEST-NAME "[gus]" GUEST-NAME

PAGE 1

ROOMS Current Guest..

365   Ferguson

END OF LIST
```

Example 4

```
:LIST ROOMS WITH GUEST-NAME "^^^^^son" GUEST-NAME

PAGE 1

ROOMS Current Guest..

122   Anderson
365   Ferguson
```

Example 5

```
:LIST ROOMS WITH GUEST-NAME "^ost^^" GUEST-NAME

PAGE 1

ROOMS Current Guest..

428   Postma

END OF LIST
```

Example 6

```
:LIST ROOMS = '11^' ROOM-CODE AVAILABLE

PAGE 1

ROOMS    Room  Available
         Code

117      D       21/04/87
119      D       22/04/87

2 ITEMS LISTED
```

Relational Operators and Logical Connectives

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

Relational operators comprise:

= or EQ	equal to
> or GT or AFTER	greater than
< or LT or BEFORE	less than
>= or GE	greater than or equal to
# or NE or NOT or NO	not equal to or null attribute value
BETWEEN	between but not equal to

If a relational operator is not specified, EQ is assumed (except when string searching on item-id).

A relational condition is resolved by comparing every item-id or attribute value in the items selected by an item-list, with those in the ENGLISH sentence. For left justified attributes this is done on the basis of ASCII character code, character by character from left to right.

Logical connectives comprise:

AND	both connected parts must be satisfied (true)
OR	either connected part must be satisfied (true)

The precedence of logical connectives is AND then OR.

If a logical connective is not specified, OR is assumed.

Example 1

LIST ROOMS > '217'

selects items with item-ids greater than 217.

Example 2

LIST ROOMS WITH ROOM-CODE "S"

selects single rooms.

Example 3

LIST ROOMS WITH NO GUEST-NAME

selects rooms with no guest.

Example 4	**LIST ROOMS = '217' > '326'** selects room 217 and those greater than 326.
Example 5	**LIST ROOMS WITH ROOM-CODE "D" AND WITH BED-CODE "K"** selects double rooms with a king-size bed.

OUTPUT-SPECIFICATION	Item lists and selection criteria limit the items selected, whereas output specifications indicate the attributes (of the selected items) to be output. If an output specification is not specified then the attributes defined by the file's dictionary as 'default' attributes are output.
Multivalued Attributes	If a multivalued attribute is specified in the output-spec then each multivalue is output on a separate line.
Print Limiting	Output can be limited to specific multivalues by following an attribute name in the output-spec by a print limiting clause comprising logical/relational operators and values enclosed in double quotes. Values which do not satisfy the print limiting are 'blanked out'.
Associative Attributes	In the above examples the asterisks (*) under the headings 'Bill Date' and 'Amount' indicate that the dictionary defines the BILL-DATE and BILL-AMOUNT attributes as associated with the BILL-CODE (master) attribute. If a print limiting clause is specified immediately following a master attribute then this limits output of not only the master attribute but also the secondary (associative) attributes.

Example 1

```
:LIST ROOMS < '300' ROOM-TYPE RATE GUEST-NAME

PAGE 1

ROOMS   Room.......  Rate....  Current Guest
        Type

117     Double Occ.     72.00  Rizzo
194     Double Occ.     64.00  Hynes
140     Single Occ.     68.00  Lynch
 :          :              :      :
 :          :              :      :
```

Example 2

```
:LIST GUESTS '117''119' BILL-CODE BILL-DATE
BILL-AMOUNT BILL-DESC

PAGE 1

GUESTS  Bill   Bill Date Amount.. Description..
        Code
               *         *
117        2    17/04/87   $62.00 Room
           6    20/04/87   $17.95 Dinner
          18    20/04/87   $8.76  Miscellaneous
119        2    21/04/87   $56.00 Room
          15    21/04/87   $23.98 Telephone
          13    21/04/87   $17.95 Bar (2)

2 ITEMS LISTED
:
```

Example 3

```
:LIST GUESTS '117''119' BILL-CODE BILL-DATE
BILL-AMOUNT > "20" BILL-DESC

PAGE 1

GUESTS  Bill   Bill Date Amount.. Description..
        Code
               *         *
117        2    17/04/87   $62.00 Room
           6    20/04/87          Dinner
          18    20/04/87          Miscellaneous
119        2    21/04/87   $56.00 Room
          15    21/04/87   $23.98 Telephone
          13    21/04/87          Bar (2)

2 ITEMS LISTED
:
```

Example 4 **:LIST GUESTS < '200' BILL-CODE "15" BILL-DATE BILL-AMOUNT**

```
PAGE 1

GUESTS  Bill    Bill Date  Amount..
        Code
                *          *

147       15     20/04/87     $3.87
117
119       15     21/04/87    $23.98
122
140
142
194
143       15     18/04/87    $12.95
144       15     20/04/87    $22.50

9 ITEMS LISTED
:
```

TAPE modifier

This indicates that retrieval is from the tape file positioned on the tape drive rather than from a disc file. The required attribute definition items are found on the system in the dictionary of the specified file.

If a dictionary file is specified, the attribute definition items are retrieved from the Master Dictionary (MD) of the account in use.

The TAPE modifier is only valid with the LIST, SELECT, COUNT, SUM, STAT, I-STAT, and LIST-LABEL verbs.

THROWAWAY CONNECTIVES

Throwaway connectives do not affect the meaning of an ENGLISH sentence but they do make it more readable.

For example,

```
SORT ROOMS
```

could be input as

```
SORT THE ROOMS FILE
```

where 'THE' and 'FILE' are throwaway connectives.

A number of throwaway connectives are defined as standard on the system. Others may be created by copying any of the standard throwaway connective items in the master dictionary (MD) to a new item of the required name.

The standard set of throwaway connectives comprises:

A, AN, ARE, ANY, FILE , FOR, IN, ITEMS, OF, OR, THE

In common with modifiers, operators and logical connectives, these are reserved words which may not be used as attribute names if contained in the MD of the account.

Overview 5-3
Sort-Order 5-3
Headings and Footings 5-6
  Heading 5-6
  Footing 5-6
  Heading and Footing Options 5-6
  Expanded Print 5-7
Generating Totals 5-9
  Labelling the Total 5-9
Breaking on Attribute Values 5-11
  Generating Sub-totals 5-12
Miscellaneous Modifiers 5-14

Overview

As well as specifying the items and attributes to be output you also have a number of choices in the way the information is presented. These choices comprise:

Sort-order -	the order of output of items.
Headings and Footings -	text to appear at the top and bottom of every page.
Breaks -	a break of a few lines when a specified attribute changes in value.
Totals -	a total value of a specified attribute, with or without a user-specified label. Also sub-totals when a break-on-attribute-value is specified.

SORT-ORDER

In the absence of any 'sort-order' the order of output of items for LIST, LIST-ITEM and other verbs which do not perform a sort, is according to any item-list. If an item-list is not specified, items are output in the order in which they are stored in the file.

For verbs such as SORT, SORT-ITEM and similar verbs which perform a sort, the order of output is, for left justified item-ids, in ascending order of the ASCII value of item-id (character by character, from left to right).

The order of output can be changed by adding a 'sort-order' clause to the ENGLISH statement. A 'sort-order' takes one of the following forms:

Sort-Order	Order of Output
BY att-name	Ascending order of ASCII value of specified attribute (character by character, left to right)
BY-DSND att-name	Descending order of ASCII value of specified attribute (character by character, left to right)
BY-EXP att-name	As BY att-name, except that if the specified attribute is multivalued, each multivalue is treated as a separate value and is output in its appropriate position.

BY-EXP-DSND att-name	As BY-DSND att-name, except that if the specified attribute is multivalued each multivalue is treated as a separate value and is output in its appropriate position.

If more than one item has a similar value for the attribute to be sorted on, then these items are arranged in ascending order of item-id.

Multiple sort order modifiers may be freely mixed; the priority order is from left to right so that in the example

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

the main sort is in ascending order of attribute QUAN. For each group of items with the same QUAN value a further sort is performed to arrange the items in ascending order of attribute PRICE. For each group of items with the same price a final sort arranges items into ascending order of item-id.

Example 1

:SORT GUESTS '117''119''144' BY BILL-DATE BILL-CODE BILL-DESC BILL-DATE BILL-AMOUNT

```
GUESTS Bill Bill Date Amount.. Description...
       Code *             *

117       2 17/04/87  $62.00   Room
          6 20/04/87  $17.95   Dinner
         18 20/04/87   $8.76   Miscellaneous
144       4 17/04/87  $12.95   Breakfast
         15 20/04/87  $22.50   Telephone
         16 20/04/87  $12.05   Newspapers
119       2 21/04/87  $56.00   Room
         15 21/04/87  $23.98   Telephone
         13 21/04/87  $17.95   Bar (2)

3 ITEMS LISTED
```

Example 2

```
:SORT ROOMS WITH ROOM-CODE "D" BY-DSND
LEAVE-DATE LEAVE-DATE BED-TYPE RATE

PAGE 1

ROOMS    Leave...... Bed........Rate....
         Date        Type

147      02/05/87    King        72.00
194      30/04/87    Double      64.00
144      27/04/87    Queen       68.00
142      26/04/87    King        72.00
119      22/04/87    Double      64.00
117      21/04/87    King        72.00

6 ITEMS LISTED
:
```

Example 3

```
:SORT ROOMS WITH RATE < "100" BY RATE BY
AVAILABLE RATE AVAILABLE ROOM-TYPE

ROOM TYPE

PAGE 1

ROOMS   Rate....   Available  Room ...........
                              Type

119        64.00   22/04/87   Double Occ.
194        64.00   30/04/87   Double Occ.
122        64.00   03/05/87   Double Occ.
140        68.00   21/04/87   Double Occ.
144        68.00   27/04/87   Double Occ.
143        68.00   01/05/87   Single Occ.
 :          :          :          :
 :          :          :          :
 :          :          :          :
```

HEADINGS AND FOOTINGS

In the absence of any modifiers, reports always include, at the top of each page, the page number, time and date and, at the end of the report, a line of the form 'n ITEMS LISTED'.

You may optionally specify a heading (text to appear at the top of every page) and/or a footing (text to appear at the bottom of every page).

Heading

The specified heading appears at the top of every page of output and the system generated page number, time and date heading, and items listed message are suppressed.

Syntax

The general form of a HEADING specification, which may be placed anywhere in an ENGLISH sentence, is:

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

Footing

Any specified footing appears at the bottom of every page of output. A user-specified footing does not suppress the system generated page number, time and date, or 'n ITEMS LISTED' message.

Syntax

The general form of a FOOTING specification, which may be placed anywhere in an ENGLISH sentence, is:

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

Heading and Footing Options

There are a number of options which, if included in a HEADING or FOOTING specification, are replaced by appropriate data or cause appropriate action when the sentence is executed.

The options, which must be enclosed in single quotes, comprise:

Option	Explanation
B	Break. If a BREAK-ON modifier with a 'B' option is also included in the sentence then the value of the first BREAK-ON attribute on the page is inserted; otherwise this has no effect.
C{n}	Centre. Centres text within the heading/footing. If a number n is specified then text is entered within a line of length n.
D	Date. Inserts the current system date.
F	File. Inserts the file-name.
L	Line. Specifies that a new line is to be started.
N	No page. Defeats automatic paging of output.
P	Page. Inserts the current page number.
PP	Page Justify. Inserts the current page number right justified in a field of four blanks.
T	Time. Inserts the current system time and date.
''	Two successive single quotes print a single quote mark.

Expanded Print

An expanded print capability is available on some printers. This causes designated text to be output with each character the width of two normal characters.

Headings and footings may be printed in expanded print by preceding the text string with the ASCII 'SO' character (CTRL N). The string that follows, up to a RETURN, will be output in expanded print.

Example 1

```
:LIST ROOMS WITH GUEST-NAME GUEST-NAME HEADING
"GUEST LIST"

GUEST LIST
ROOMS    Current Guest

117      Rizzo
194      Hynes
535      Evans
 :          :
 :          :
```

Example 2

```
:LIST ROOMS WITH GUEST-NAME GUEST-NAME FOOTING
"GUEST LIST"

PAGE 1

ROOMS    Current Guest

117      Rizzo
194      Hynes
 :          :
 :          :
411      Gallagher
401      Palmer
478      Kolman
GUEST LIST

24 ITEMS LISTED
```

Example 3

```
:LIST ROOMS WITH GUEST-NAME GUEST-NAME HEADING
"'L'GUEST-LIST 'DL'" FOOTING "'L' PAGE 'PL'"

GUEST LIST 21 JUL 1987

ROOMS    Current Guest

117      Rizzo
194      Hynes
535      Evans
 :          :
 :          :
211      Lewis

PAGE 1
```

GENERATING TOTALS

The TOTAL modifier generates a total for an attribute with numerical values.

Syntax

The general form of the TOTAL modifier is:

TOTAL attribute-name

Example

:LIST GUESTS LAST-NAME TOTAL BILL-TOTAL

```
GUESTS   Last Name.  Bill Total

401      Palmer         $149.25
147      Janson         $122.82
 :         :               :
 :         :               :
365      Ferguson       $159.70

***                   $3,639.99

24 ITEMS LISTED
```

Labelling the Total

The GRAND-TOTAL modifier allows user-specified text to replace the three asterisks (***) printed in the item-id column.

Syntax

The general form of the GRAND-TOTAL modifier is:

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

Option	Explanation
L	Line. Suppresses the blank line preceding the GRAND-TOTAL line. This option overrides the 'U' option if both are specified.
P	Page. Causes the GRAND-TOTAL line to be output on a new page.
U	Underline. Causes an underline of the final TOTAL attribute value.

Example

```
:LIST GUESTS NAME TOTAL BILL-TOTAL GRAND-TOTAL
"SUM OWING 'U'"

GUESTS   Guest Name..........Bill Total

401      Sharon R. Palmer       $149.25
147      Loretta T. Janson      $122.82
 :
 :
365      Marilyn T. Ferguson    $159.70
                              ---------
SUM OWING                     $3,639.99

24 ITEMS LISTED
:
```

BREAKING ON ATTRIBUTE VALUES

The BREAK-ON modifier causes a break of three lines when the value of the specified attribute changes. Three asterisks (***) are displayed in the BREAK-ON attribute column.

Syntax

The general form of the BREAK-ON modifier, which should be placed at the appropriate point in the 'o/p-spec', is:

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

where:

text

is any text to replace the three asterisks printed on the breakline.

Option	Explanation
B	Break. Specifies that the first value of this attribute name on the page is to replace the 'B' option specified in the HEADING or FOOTING. Only the first 24 characters of the attribute are used.
D	Data. Suppresses the break data line entirely if there was only one line of data since the last break.
L	Line. Suppresses the blank line preceding the break data line. This option overrides the 'U' option if both are specified.
P	Page. Causes a new page to be started after the data associated with this break has been output.
R	Rollover. Causes one or more control-break lines occurring at the end of a page to output on the same page. Without this option page rollover occurs after the first control-break at the end of the page is printed.
U	Underline. Causes underlining of all TOTAL fields.
V	Value. Causes the value of the control-break attribute to be inserted at this point in the BREAK-ON label.

Up to 15 control-breaks are allowed in one sentence; precedence is left to right.

If the BREAK-ON attribute data is not required in the output then the column width for that attribute should be set to zero (attribute 010 in the attribute definition item).

Generating Sub-totals

If a TOTAL modifier is used in the same sentence as a BREAK-ON modifier then subtotals are printed whenever a break occurs in addition to a final total.

Multiple TOTAL modifiers may be specified.

Example 1

:SORT GUESTS BY ARRIVAL-DATE BREAK-ON ARRIVAL-DATE NAME

```
PAGE 1

GUESTS Arrival. Guest Name..........
       Date

144    15/04/87 Mr. & Mrs. H. Irving
211    15/04/87 David M. Lewis
222    15/04/87 Michael T. O'Brien
318    15/04/87 Janis M. Petrillo
354    15/04/87 D. Taylor

          ***

117    16/04/87 Loretta Rizzo

          ***

140    17/04/87 Susan P. Lynch
 :        :           :
```

Example 2

```
:SORT GUESTS BY ROOM-TYPE BREAK-ON ROOM-TYPE
LAST-NAME TOTAL BILL-TOTAL GRAND-TOTAL
"INCOMINGS"
```

```
PAGE 1

GUESTS Room Type....... Last Name... Bill Total

211    Deluxe           Lewis          $241.68
289    Deluxe           Mendell        $195.87
 :       :                :               :
 :       :                :               :
428    Deluxe           Postma         $152.50

       ***                           $1,146.96

117    Double Occ.     Rizzo            $88.71
 :       :                :               :
 :       :                :               :
194    Double Occ.     Hynes            $72.35

       ***                             $568.36
 :       :                :               :
 :       :                :               :
222    Suite           O'Brien         $117.59
 :       :                :               :
 :       :                :               :
478    Suite           Kolman          $160.00

       ***                           $1,384.74

INCOMINGS                            $3,639.99

24 ITEMS LISTED.
```

MISCELLANEOUS MODIFIERS

There are a number of other modifiers that perform specific actions:

Modifier	**Function**
COL-HDR-SUPP	Suppresses output of the page number, time and date heading, column headings and the 'n ITEMS LISTED' message. This essentially produces a continuous format without page breaks.
HDR-SUPP or SUPP	Suppresses output of the page number, time and date heading, and the 'n ITEMS listed' message.
ID-SUPP	Suppresses output of item-ids.
DBL-SPC	Inserts a blank line between items.
DET-SUPP	Suppresses detail lines when used with TOTAL or BREAK-ON modifiers.
LPTR	Directs output to the spooler for subsequent printing. (Produces the same result as the P option.)
NOPAGE	Suppresses automatic paging at a terminal. (Produces the same result as the N option).

Chapter 6
Dictionary Files

Overview 6-3
Attribute Definition Items 6-4
  Structure 6-4
Default Attributes 6-6
Item Size 6-6
Dictionary File Creation 6-10
Attribute Definition Item Creation 6-10

Overview

In order to access data within a data file using ENGLISH the fields or 'attributes' of each item within that file must be defined to the system so that each attribute can be called up by name. This is done via a dictionary file containing attribute definition items.

The hierarchy of files is as follows:

```
                      ------------
                     |   SYSTEM   |
                     |            |
                      ------------
                           |
                           |
                           v
 ------------         ------------                          ------------
|   MASTER   |       |   MASTER   |                        |   MASTER   |
| DICTIONARY | . . . | DICTIONARY | . . . . . . .          | DICTIONARY |
|            |       |            |                        |            |
 ------------         ------------                          ------------
    /   |   \       /      |      \                          /   |   \
   /    |    \     /       |       \                        /    |    \
  /     |     \   /        |        \                      /     |     \
                 /         |         \
                /          v          \
    ------------      ------------      ------------
   | DICTIONARY |    | DICTIONARY |    | DICTIONARY |
   |    FILE    |    |    FILE    |    |    FILE    |
   |            |    |            |    |            |
    ------------      ------------      ------------
         |                 |                 |
         |                 |                 |
         v                 v                 v
    ------------      ------------      ------------
   |    Data    |    |    Data    |    |    Data    |
   |    File    |    |    File    |    |    File    |
   |            |    |            |    |            |
    -----------       ------------      ------------
```

File SYSTEM also known as the System Dictionary, contains an item for each account on the system. This item points to a file identified as M/DICT or MD, known as the Master Dictionary, which is unique to the corresponding account. The MD contains items which point to the DICT files defined on that account and each DICT file contains a single item which points to the associated data file.

ATTRIBUTE DEFINITION ITEMS

The simplest and most usual type of attribute definition item defines a single data attribute. The item-id of the item is then the name by which the attribute is known, and this can be included in an ENGLISH sentence applying to the corresponding data file.

A definition item doesn't necessarily have to define a single data attribute; it may link together two or more data attributes which need to be displayed together in a report.

Furthermore any number of definition items can be associated with a data attribute, with each item specifying different processing to be carried out.

Structure

An Attribute Definition Item comprises ten attributes, as follows:

<u>Attribute</u>	<u>Content</u>
1	D/CODE. Defines the type of item. An attribute definition item contains one of the following three codes, each having a different effect as indicated: A If a column heading is not specified in attribute 3 then the definition item-id is used. S If a column heading is not specified in attribute 3 no column heading is shown. X Defined attribute is not output as default but maintains order of default attributes. (See DEFAULT ATTRIBUTES in this chapter)
2	A/AMC (Attribute/Attribute Mark Count) attribute. Usually contains the number of the data item attribute being defined. However, when a concatenation of two or more attributes is being defined this contains a zero if the concatenation is to be performed unconditionally. It contains an attribute number if the concatenation is to be performed only if that attribute is non-null, and otherwise is to output a null value.

3 Column heading for the attribute on output. If the heading covers more than one line then the end of one line and the start of the next is specified by a value mark (CTRL])

If this attribute is null, the column heading is determined as described under 1 above.

4 to 6 Reserved

7 V/CONV. Contains any conversion specification defining processing to be done on each attribute value immediately before output. Multiple conversion codes can be specified separated by value marks (CTRL]); multiple codes are processed on a left to right basis. (See CORRELATIVES AND CONVERSIONS.)

8 V/CORR. Contains any correlative specification, defining processing to be done on each attribute value before sorts/selects (and any conversions) processed. Multiple correlative codes can be specified separated by value marks (CTRL]); multiple codes are processed on a left to right basis. (See CORRELATIVES AND CONVERSIONS).

9 V/TYP. Specifies how the attribute values are to be positioned within the column on output. A code value must be specified and it may be one of the following:

L Left justified. If value greater than column width (specified in attribute 10), output continues on next line and takes up as many lines as necessary.

R Right justified.

I Left justified, but lines after the first are indented one space.

T 'Text data', left justified, but folded at blanks if possible.

U Left justified, but entire value printed on one line ignoring column boundaries.

10	V/MAX. Defines the column width within which attribute values are to be printed. A numeric value (number of characters) must be specified. Note that if the value given here is less than the width of the column heading the width specified here is overridden.

DEFAULT ATTRIBUTES

Default attributes are attributes that are automatically included in a report when none have been specified. The most commonly required attributes are normally specified as defaults to obviate the need for re-typing the same names each time a report is required. If default attributes are not defined then, in the event of output attributes not being specified in an ENGLISH sentence, item-ids only are output.

Default attributes are defined via attribute definition items and are those which have item-ids which are sequential integers, that is 1,2,3,4. Default attributes thus often have two definitions associated with them - one with the attribute name as item-id and one with a number as item-id. When output attributes are not specified the attributes defined by sequential integer item-ids are output, in numerical order.

As stated, default attributes have item-ids which are sequential integers. If there are items 1, 2, 4 and 5 then only attributes 1 and 2 are considered as being default. If items 1, 2, 3, 4 and 5 have been defined but only 1, 2, 4 and 5 are to be output then, simply change the D/CODE of item 3 from A or S to X. (Items with D/CODES of X maintain sequential order but are not output.)

In order to inhibit the output of default attributes (so that item-ids only are output) the modifier ONLY must precede the file-name.

ITEM SIZE

An attribute definition item can be set up to define an item size attribute, this can be used to LIST or SORT items conditionally on their size, in number of bytes, as follows:

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

Example 1 A file PERSONNEL contains items that have employees last names as attribute 1, and first names as attribute 2. A definition item with an item-id of LAST.NAME is used to define the first attribute and an item called FIRST.NAME the second, as illustrated below:

```
ATTRIBUTE
DEFINITION
ITEMS
        --- FIRST.NAME
       |
       |     ---LAST.NAME
       |    |
       |    |
DATA   |     --> 001 JOHNSON ---> 001 HOGG --->
ITEMS   ------> 002 ANDREW  ---> 002 JACK --->
                 :                :
                 :                :
                 :                :
```

These definitions enable the system to interpret ENGLISH statements such as

LIST PERSONNEL WITH LAST.NAME "JOHNSON" FIRST.NAME

Example 2 A definition for file PERSONNEL with item-id FULL.NAME specifies that full names are a concatenation of attributes 002 and 001 separated by a space.

LIST PERSONNEL WITH FULL.NAME "CAROL STEVENS"

lists the particulars of CAROL STEVENS.

Another definition allows a part of each data attribute to be identified.

Example 3 The personnel file of another company has a different arrangement of data. In this file attribute 1 contains the full names of employees. Three definitions are set up as follows:

Item-id	Defines Data
FULL.NAME	Attribute 1
FIRST.NAME	Attribute 1 up to first space character.
LAST.NAME	Attribute 1 from first space character on.

Example 4 **:LIST ROOMS**

```
PAGE 1

ROOMS  Room  Room........ Bed..... Room..........
       Code  Type         Type     Rate

117    D     Double Occ.  King       72.00......
194    D     Double Occ.  Double     64.00......
535    P     Penthouse    WaterBed  164.00......
140    S     Single Occ.  Queen      68.00......
119    D     Double Occ.  Double     64.00......
318    ST    Suite        WaterBed  104.00......
 :      :     :            :          :
 :      :     :            :          :
 :      :     :            :          :
```

:LIST ONLY ROOMS

```
PAGE 1

ROOMS

117
194
535
140
119
318
:
:
```

Example 5 **:SORT STOCK WITH SIZE > "300" SIZE**

```
PAGE 1

Account... Size...

23060           596
23075           317
23080           318
35085           404

4 ITEMS LISTED.
```

DICTIONARY FILE CREATION

A dictionary file is usually created at the same time as the data file with which it is associated. For example,

```
CREATE-FILE TEST (1 11
```

ATTRIBUTE DEFINITION ITEM CREATION

Once you have created the dictionary and data files you can create attribute definition items (and if required, data items) using the EDITOR (see the EDITOR Reference Manual). For example,

```
ED DICT TEST NAME
```

enters the EDITOR to allow creation of a definition item for attribute NAME.

Chapter 7
Correlatives and Conversions

Overview 7-3
Multiple Correlatives/Conversions 7-3
Correlative or Conversion? 7-4
  BREAK-ON Modifier 7-4
  TOTAL Modifier 7-5
Correlative and Conversion Codes 7-6
Algebraic Functions: A Code 7-7
  Embedded Decimal Points: 'An' Format .. 7-7
  'An; expression' Format 7-8
  Operands 7-8
  Conversions 7-9
  Operators 7-10
  Special Functions 7-12
Concatenation: C Code 7-14
Date Formatting: Date Codes 7-17
Associative Attributes: D1 and D2 Codes . 7-20
Mathematical Functions: F Code 7-22
  Operands 7-23
  Operators 7-23
  F Code Stack 7-24
  F Code Special Operands and 'Fn' Format 7-26
  Embedded Decimal Points: 'Fn;' Format . 7-28
  Summary of F Code Stack Operations 7-28
Group Extraction: G Code 7-31
Mask Character: MC Codes 7-32
Mask Decimal: MD Code 7-34
Mask Field: MF Code 7-36
Mask Packed: MP Code 7-39
Time Formatting: MT Code 7-40
Mask Hexadecimal: MX Code 7-41
Text Extraction: T Code 7-42
File Translation: Tfile Code 7-43
Sublists: V Code 7-47

Overview

Correlatives and conversions are used to process data. Conversions are specified in attribute 7 and correlatives in attribute 8 of an attribute definition item.

Correlative codes are applied to data to produce a new (intermediate) format which is used for sort/selection purposes.

Conversion codes are applied to selected data just prior to output but after any sorts/selections have been carried out; their purpose, therefore, is to convert data into the format required in the output report. In addition, inverses of many of the conversion codes are applied to appropriate values specified in ENGLISH enquiry sentences so that they can be compared with internal values.

```
 ----------           ------------           ---------
|  STORED   | corr. |INTERMEDIATE| conv. | OUTPUT  |
|  FORMAT   |------ |   FORMAT   |----->| FORMAT  |
|           |       |            |      |         |
 ----------           ------------           ---------
                      Used by sorts
                     and selections

           ------------    inverse      ----------
          |INTERMEDIATE|  conversion  | VALUE IN |
          |   FORMAT   |<-------------| ENGLISH  |
          |            |              | SENTENCE |
           ------------                 ----------
```

Note that conversion codes are applied only to those values that have been selected for output. Correlative codes are applied to all values (as indicated by a file-name, item-list and output-spec before sort/selection takes place. Correlatives typically require more processing time than conversions as a result. Consequently, it is more efficient to carry out any processing as a conversion than as a correlative except when that processing must be carried out before sorting and/or selection.

MULTIPLE CORRELATIVES/ CONVERSIONS

Multiple conversion/correlative codes can be specified and these must be separated by value marks (CTRL]). Multiple codes are processed on a left to right basis, each code acting on the result produced by the previous code.

```
 --------                     ---------                   --------
| STORED |correl1,correl2..| INTER-   |conv 1,conv 2..| OUTPUT |
| FORMAT |----------------->| MEDIATE  |--------------->| FORMAT |
|        |                  | FORMAT   |                |        |
 --------                     ---------                   --------
```

CORRELATIVE OR CONVERSION?

Most of the codes described in this section can be used either as correlatives or conversions In deciding which is appropriate, some thought must be given to the intermediate and output formats required.

As an example consider date handling. Dates are normally stored as integers which represent numbers of days (plus or minus) from 31 December 1967 (so that 21 December 1967 is stored as -10, 5 January 1968 is stored as 5, and so on). Holding dates in this format enables accurate sorts and selects to be carried out and takes up less storage space. Dates are converted to the internal format via a conversion code used in DATA/BASIC before storage. Reciprocal processing is needed to convert them back into the required output form for inclusion in reports. In this case the intermediate format, used in sorts and selects, must be the same as the stored format and so a correlative is not needed; a date code conversion simply converts the internal format just prior to output. A reverse conversion is applied to dates specified in ENGLISH sentences to produce values suitable for internal comparison.

As another example consider a FULL-NAME attribute definition which is the concatenation of two attributes, FIRST-NAME and LAST-NAME. If enquiries such as 'LIST FILE WITH FULL-NAME= "Andrew Lockwood"' are to be made then, since an inverse concatenation code is not applied, the intermediate format must comprise the concatenated names. The concatenation code should, therefore, be specified as a correlative.

Correlatives are processed just prior to a sort or selection and conversions are processed just prior to output. There are, however, other instances when it is necessary to be aware of what processing has actually taken place - for example, in the use of BREAK-ON and TOTAL modifiers.

BREAK-ON Modifier

A BREAK-ON modifier causes a break when the intermediate value changes. As a conversion converts all identical intermediate values to the same output value, the user normally gets the required results. However, when different intermediate values convert to the same output value, the output appears to contain unnecessary breaks. If this is possible, the use of a correlative instead of a conversion should be considered.

TOTAL Modifier

A TOTAL is produced as follows:

1. Any correlatives are applied to individual attribute values.

2. The values produced are summed.

3. Any conversions are applied to this summed value.

CORRELATIVE AND CONVERSION CODES The correlative and conversion codes are summarised in Table 7-1.

Table 7-1 Correlative/Conversion Codes

Code	Description
A	Algebraic. Perform mathematical and logical operations on values.
C	Concatenation. Concatenate attributes/literal values
D etc.	Date. Convert dates for output.
D1	Define primary. Define primary associative attribute linked with set of secondary associative attributes.
D2	Define secondary. Define secondary associative attribute.
F	Function. Obsolescent; see A code.
G	Group. Extract contiguous segment(s) of a value.
MC etc.	Mask Character. Extract and convert characters.
MD	Mask Decimal. Convert and scale numbers.
MF	Mask Field. Perform literal insertions and currency formatting.
MP	Mask Packed. Convert packed decimals for output.
MT	Mask Time. Convert times for output.
MX	Mask Hexadecimal. Convert character strings to their hexadecimal ASCII equivalents.
T	Text extraction. Extract a contiguous string of characters from a value.
Tfile	File translation. Translate values using a second file.
Uxxxx	'User exit'. Execute assembler-coded routines to process value(s).
V	Sublist. Specified in attribute 8 of the DL/ID to indicate a sublist attribute.

ALGEBRAIC FUNCTIONS: A CODE

The A code allows algebraic expressions involving attributes to be specified. Such functions are usually defined as correlatives for the following reasons:

. Any selection or sort normally needs to be done on processed values.

. Reverse processing is not carried out on relevant values in ENGLISH sentences.

Syntax

A{n};expression

where:

n

is a scaling factor in the range 1 to 6. which multiplies each value by that power of ten before carrying out rounding, resulting in a scaled integer, (n=0 is only valid if ';expression' is omitted).

You must specify n if values containing embedded decimal points are to be handled. The decimal point is then recognised and the value converted to an integer (equal to the original value, multiplied by 10 to the power n, rounded to the nearest integer).

expression

comprises operands, operators, conditional statements, conversions and special functions, as described below.

If ';expression' is omitted, the A/AMC attribute (attribute 2) of the attribute definition item containing the A code must contain an attribute number. If ';expression' is included the A/AMC attribute must be zero.

Embedded Decimal Points: 'An' Format

The 'An' correlative converts an internally stored number(s) with a decimal point (typically an ALL-generated numeric field) to a scaled integer. In line two (A/AMC) of this attribute definition item, you must enter the attribute number containing the data to be processed by this correlative (that is, not a zero). An 'MD' or 'MF' conversion may subsequently be used to print the value with a decimal point, comma, etc.

For example, consider attribute values as follows:

37.65	273.95	219.84

In order for any processing to be carried out on these values they must be converted to integers. This is achieved by a correlative of A2, which multiplies each value by 100. The intermediate integer values so produced can then be processed.

A conversion of MD2 is necessary in order to produce decimal values for output. An inverse conversion is also applied to appropriate values in an ENGLISH sentence.

'An; expression' Format

This correlative evaluates the expression and then converts the resulting value containing an embedded decimal point to a scaled integer. When using this correlative, you should enter a zero on line two (A/AMC) of the definition item if this attribute is to be referenced by name in other functions.

OPERANDS

Attribute Numbers

An attribute number (AMC) can be specified to indicate the attribute to be used. For example, code A;6 references attribute 6.

Certain attribute numbers have special meanings:

0	Item-id.
9999	Item size in bytes.
9998	Number of the item within the file.

Attribute Names

An attribute defined in the same dictionary can be extracted via the N function using the general format:

N(attribute-name)

The named attribute is accessed and any correlatives in attribute 008 are applied to the value before it is returned to the A code for any further processing.

Correlatives in the attribute definition item being accessed may also contain an A code referencing another attribute, and so on. (This recursive capability is an important advantage of the A code over the F code which it replaces.)

For example, code A;N(QTY) retrieves QTY value and processes any correlatives in QTY attribute definition item.

Literals

Any literal string or numeric constant may be specified enclosed in double quotes. (If the double quotes are omitted from a numeric constant then an attribute number is implied.)

For example, A;"20"+N(QTY)

Special Operands

Six special operands give the current values of system parameters:

Operand	Description
NI	Item count.
NV	Multivalue counter.
NS	Sub-multivalue counter.
ND	Number of detail lines since last control break.
D	System date (internal format).
T	System time (internal format).

Any of the special operands may be preceded by a minus sign to change the sign of the value.

For example, A; NV.

Repeat Characters

Any operand may be followed by an 'R'. This specifies that a single value should be repeated so that there will be the same number of values as a multivalued attribute used elsewhere in the calculation.

'RR' is similar except that it repeats at the multi-subvalue level.

CONVERSIONS

An operand may be followed by any conversion expression(s) enclosed in parentheses. (If more than one conversion is specified, they must be separated by sub-value marks, CTRL], and the whole expression enclosed in parentheses).

OPERATORS

Arithmetic

Four operators each connect two operands:

Operator	Value Returned
+	Sum of operands.
-	Difference of operands.
*	Product of operands.
/	Quotient (an integer value) of operands.

String

The : operator concatenates of the results of two expressions.

For example, A;5+6:3 adds attributes 5 and 6 and concatenates the result with attribute 3.

Relational

Six relational operators each connect two expressions to produce the result 1 (if the condition is true) or 0 (if the condition is false).

Operator	Meaning
<	Less than.
>	Greater than.
<=	Less than or equal to.
>=	Greater than or equal to.
=	Equal to.
#	Not equal to.

For example, A;IF N(ACT)+N(PRI)*".07">"0" THEN "CREDIT" ELSE "DEBIT"

where the sum of attributes ACT and PRI is multiplied by .07; depending on the value of this the word "CREDIT" or "DEBIT" is assigned.

Operator Precedence

Any number of levels of parenthesis may be used to determine the order of operations. In the absence of parentheses the order of decreasing priority is:

```
* and /
+ and -
relational operators
: (concatenation)
```

Two operators with the same priority are processed from left to right.

For example, 1+2*3<4 evaluates as (1+(2*3))<4; 4/5*6 evaluates as (4/5)*6

Conditional Statement

The IF statement gives the A code conditional capabilities. Its general form is:

IF expression THEN statement ELSE statement.

where:

expression

must evaluate to 1 or 0.

statement

is any statement generally resulting in a string or numerical value.

The IF statement may be nested, as in "IF 1 THEN IF 2 THEN 3 ELSE 4"; however every IF statement must evaluate to a single value.

The logical connectives AND and OR bind together conditional statements. The AND connective specifies that both connected parts must be true; the OR connective specifies that either one or both connected parts must be true. If a connective is not specified, OR is assumed.

The words IF, THEN, ELSE, AND and OR must be followed by at least one blank.

```
A;IF N(OBJ)="4" OR N(PRED)="7" THEN "VALID" ELSE
"(NO COMMENT)"
```

where if either part of the IF expression is true, "VALID" is assigned; if both parts are false "(NO COMMENT)" is assigned.

SPECIAL FUNCTIONS

Remainder: R

The remainder function takes two expressions as operands and returns the remainder of the first divided by the second. Its general form is:

R(expression,expression)

For example,

```
A;R(N(COST),"5")
```

returns the remainder when COST is divided by 5.

Summation: S

The summation function evaluates an expression and then adds together all the multivalues. Its general form is:

S(expression)

For example,

```
S(N(RATE)R*N(HOURS))
```

multiplies multivalued HOURS by single valued RATE (repeated) and then sums the mutlivalued results.

Substring: [n,n]

The substring function extracts a substring from a string of characters. The substring is defined by a start character number and a number of characters. Its general form is:

["start-char-no.","no.-of-chars"]

For example,

```
N(NAME)["2","5"]
```

returns a string of five characters starting at position 2.

Note: If quote marks are omitted then attribute numbers are implied so that N(NAME)[2,5] would retrieve a substring from NAME where the starting position is found in attribute 2 and length in attribute 5.

Example

Consider a STOCK file containing a data item for each part number. Each data item contains a price, two quantities which represent stocks of the part in two different departments, and numerous other pieces of information.

Dictionary items are set up to define the price attribute (PRICE) and each of the two quantities (QTY.P and QTY.Q).

In addition a total quantity (QTY) and value (VALUE) items are set up. QTY adds together QTY.P and QTY.Q and VALUE multiplies QTY by PRICE.

The STOCK dictionary and data items are as follows:

Dictionary Items

```
   VALUE                 PRICE   QTY                    QTY.P   QTY.Q
001 A                    A       A                      A       A
002 0                    1       0                      2       3
003 Value                Price   Qty                    Qty.P   Qty.Q
004
005
006
007 MD2,£                MD2,£
008 A;N(PRICE)*N(QTY)            A;N(QTY.P)+N(QTY.Q)
009 R                    R       R                      R       R
010 10                   10      5                      5       5
```

Data Items

```
    1125        1126       1127        1128
001  1269        157        2763        21165
002  12         0          3           5
003  6          9          17          2
 :
```

```
:LIST STOCK QTY.P QTY.Q QTY PRICE VALUE

PAGE 1

STOCK Qty.P Qty.Q Qty.. Price..... Value.....

1125      12     6    18     £12.69    £228.42
1128       5     2     7    £211.65  £1,481.55
1126       0     9     9      £1.57     £14.13
1127       3    17    20     £27.63    £552.60

4 ITEMS LISTED.
```

CONCATENATION: C CODE

Attributes and/or literal values can be concatenated using the C code as a correlative or conversion.

Syntax

C{;}n{*n}

where:

;	is optional and ignored
*	is the character to be inserted between the concatenated attributes and/or literals. A semicolon (;) is a reserved character that means no separation character is to be used. Any non-numeric (except a minus sign or a system delimiter) is valid, including a blank.
n	is any attribute mark count (AMC), or any literal enclosed in single quotes.

If the A/AMC (line 2) of the attribute definition item containing the C code is non-zero, and if that data attribute contains a null, then the C code is ignored and a null value is output. If the A/AMC is zero then the C code is always processed.

The inverse of a C conversion code cannot be applied. Thus values in an ENGLISH sentence are compared directly with intermediate values.

Example 1 Consider a TEST file with dictionary and data items as follows:

Dictionary Items

```
     CAT1         CAT2
001   A            A
002   0            0
003
004
005
006
007   C2;3;1       C2;'55'=1/4
008
009   L            L
010   10           10
```

Data Items

```
     A123         A456
001   ABC          AAA
002   DEF          BBB
003   GHI          CCC
004   JKL          DDD
 :     :            :
 :     :            :
```

:LIST TEST 'A123''A456' CAT1 CAT2

```
PAGE 1

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

A123       DEFGHIABC       DEF55=ABC/JKL
A456       BBBCCCAAA       BBB55=AAA/DDD

2 ITEMS LISTED
```

Example 2 Consider dictionary and data items as follows:

Dictionary Items

```
     CAT3         CAT4
001   A            A
002   0            2
003
004
005
006
007   C2;3         C2;3
008
009   L            L
010   10           10
```

Data Items

```
     B123         B456
001   ABC          AAAA
002   DEF
003   GHI          BBBB
 :     :            :
 :     :            :
```

```
:LIST TEST 'B123''B456' CAT3 CAT4

PAGE 1

TEST...... CAT3...... CAT4......

B123       DEFGHI     DEFGHI
B456       BBBB

2 ITEMS LISTED.
```

DATE FORMATTING: DATE CODES

In order to save space and simplify sorts and selects, dates are normally stored as integers which represent numbers of days (plus or minus) from December 31, 1967. The Date codes (which are almost always specified as conversions) convert internally stored dates back into suitable forms for output.

When a date is specified in an ENGLISH sentence the inverse conversion is performed before comparison with intermediate values. A date specified in any valid format is converted into the internal format; it does not have to be in the same format as is output.

If the year is not specified then the current year is used. If the year is input as two digits (for example, 29 or 73), then the twentieth century is used if the year is in the range 30 to 99 (inclusive), the twenty-first century if in the range 0 to 29 (inclusive).

Syntax

There are a number of Date codes which can be applied:

D{n}{*m}{s}

where:

n is an optional single digit in the range 0 to 4. This specifies the number of digits to be printed in the year field on output. If n is omitted, four digits are assumed.

***m** is optional and specifies the number, of concatenated segments separated by a nonnumeric separator, *, that are to be skipped before the date portion of an attribute is encountered.

* is the nonnumeric separator. This cannot be a system delimiter or ';'. m is a single numeric digit.

s is an optional nonnumeric character to be used as the separator between the day, month and year on output. (Output is in the form ddsmmsyy{yy} or mmsddsyy{yy} where 's' represents a space, depending on your system/ terminal setting.

DD	Returns just the day of the month.
DJ	Returns the Julian day of the year as a number from 1 to 365 (366 in a leap year).
DM	Returns just the month as a number from 1 to 12.
DMA	Returns the name of the month.
DQ	Returns the quarter as a number from 1 to 4.
DW	Returns the day of the week as a number from 1 to 7 denoting Monday to Sunday inclusive.
DWA	Returns the name of the day of the week.
DY{n}	Returns just the year. If the optional 'n' is present, and in the range 0 to 4, it will return the rightmost 'n' digits of the year. If 'n' is not present, or is in the range 5 through 9, the year defaults to 4 digits.

In addition to the above codes which act on dates stored in the internal format, there is also a DI code which can be used on dates stored in the external format:

DI	Returns the date in internal format. This can be used as a correlative or a conversion.

As previously stated, the internal date is defined as the number of days (plus or minus) from December 31, 1967. The conversion to the internal format is achieved using the ICONV DATA/BASIC function. The following list illustrates the internal format:

Date	Internal Value
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

Examples

D Code	Internal Value	Output Value
D	2704	27 MAY 1975
D/	2704	27/05/1975
D-	2707	27-05-1975
DO	2704	27 MAY
DO/	2704	27/05
D2*	2704	27*05*75
D	-13732	27 MAY 1930
D/	-13732	27/05/1930
D-	-13732	27-05-1930
DO/	19141	27/05
D2*	19141	27*05*30
D%1	ABC%2704	ABC%27 MAY 1975
D%1/	ABC%2704	ABC%27/05/1975
D%1-	ABC%2704	ABC%27-05-1975
DO%1	ABC%2704	ABC%27 MAY
DO	ABC%2704	ABC%2704
DD	2704	27
DJ	2704	147
DM	2704	5
DMA	2704	MAY
DQ	2704	2
DW	2704	2
DWA	2704	TUESDAY
DY	2704	1975
DY2	2704	75
DI	27 MAY 1975	2704

ASSOCIATIVE ATTRIBUTES: D1 AND D2 CODES

Attributes can be logically grouped with a single primary attribute. Such secondary attributes are only output if the primary attribute is also specified for output. Also, they are output in a pre-defined sequence and are marked as being related to a primary by an asterisk under the column heading.

The primary attribute is indicated by a D1 correlative in the attribute definition item, secondary attributes by D2 correlatives.

If more than one correlative is being specified then any D1 or D2 correlative must be specified first. D1 correlatives defined for attributes that also have F correlatives are ignored.

If the output of a primary attribute is suppressed by a print limiting clause then the output of that attribute's associated secondary attributes is also suppressed.

Syntax

The general form of a D1 correlative is:

D1;amc{;amc}...

where:

amc

is the attribute mark count of a defined secondary associative attribute. All the secondary attributes must be included in the order in which they are to appear on output. However, the amc of the primary attribute must be numerically less than amcs of the secondary attributes.

The general form of a D2 correlative is:

D2;amc

where:

amc

is the attribute mark count of the primary associative attribute.

Example

In file TEST the attributes CODE, UNITS and POUNDS are secondary attributes linked to the DATE primary attribute. The DATE, CODE, UNITS and PRICE correlatives are now shown together with a listing of TEST file item '5330':

Attribute	**Correlative**
DATE	D1;21;22;23
CODE	D2;20
UNITS	D2;20
PRICE	D2;20

```
:LIST TEST '5330' DATE CODE PRICE UNITS

PAGE 1

TEST  DATE.......  CODE.. UNITS..PRICE..
                   *      *      *
5330  07 APR 1987  P                9.50
      18 MAR 1987  B                9.50
      17 MAR 1987  T                2.00
      13 MAR 1987  R        2721    7.50
      05 FEB 1987  P                9.20
      15 JAN 1987  B                9.20
      14 JAN 1987  T                2.00
      10 JAN 1987  R        2696    7.20

END OF LIST
```

Note that although the attributes are specified in the order DATE CODE PRICE UNITS they are output in the order specified in the D1 correlative.

MATHEMATICAL FUNCTIONS: F CODE

The F code has been replaced in normal use by the easier to use A code. It is included here for reference by existing users only.

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

Syntax

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

where:

n is described in the topic F CODE SPECIAL OPERANDS

element may be any of the following:

- A numeric AMC specifying an attribute value to be pushed onto the stack, optionally followed by an "R" (Repeat code), optionally followed by any conversion specification(s) enclosed in parentheses.
- A constant of the form Cn where "n" is a numeric or string constant to be pushed onto the stack.
- A D which specifies the current date is to be pushed onto the stack.
- A T which specifies the current time is to be pushed onto the stack.
- A special two-character operand designating a particular system counter.
- An operator which specifies an operation to be performed on the top two entries in the stack.

Operands

Operands always cause a single push onto the stack, with existing values (if any) moved down one position in the stack.

The following operands are available:

Operand	Description
amc{R}{(conversion)}	Numeric AMC, optional repeat code, optional conversion specification(s).
Cn	Numeric or string constant.
D	System date (internal format).
T	System time (internal format).
NI	Item counter.
ND	Number of detail lines since last BREAK on a Break data line. This has a value of 1 on any detail lines, and is equal to the item counter on a grand-total line. This operand is used to get averages, for example, within the control- break structure.
NV	Multivalue counter (columnar listing only).
NS	Sub-multivalue counter (columnar listing only).

Operand specification is further described in the topics F CODE STACK and F CODE SPECIAL OPERANDS.

Operators

The relational operators compare STACK2 to STACK1. After the operation, STACK1 contains either a 1 or 0, depending upon whether the result is true or false respectively. (For example, if the F code was F;C3;C3;= then STACK1 would contain a 1.)

The following operators are available:

Operator	Function
*	Multiplies the top two entries in the stack.
/	Divides STACK1 by STACK2.
R	Returns the remainder to top of stack.
+	Adds the top two entries in the stack.
-	Subtracts STACK2 from STACK1.
S	Places the total sum of all STACK1 multi-values at the top of the stack.
"	Places a duplication of STACK1 onto the stack.
-	Exchanges the top two positions in the stack.
^	Pops the stack.
:	Concatenates STACK1 with STACK2.
[]	Retrieves substring of STACK3. STACK2 specifies starting column, and STACK1 specifies number of characters.
=	"Equal to" relational operator.
<	"Less than" relational operator.
>	"Greater than" relational operator.
#	"Not equal to" relational operator.
[	"Equal to or less than" relational operator.
]	"Equal to or greater than" relational operator.

F Code Stack

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

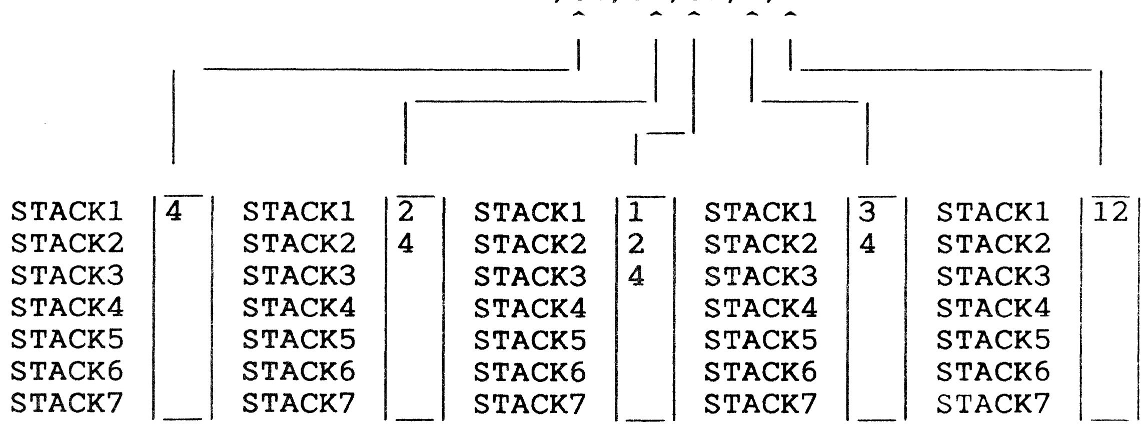
```
STACK1 -> |________|
STACK2 -> |________|
STACK3 -> |________|
STACK4 -> |________|
STACK5 -> |________|
STACK6 -> |________|
STACK7 -> |________|
```

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

The F code comprises any number of operands or operators in reverse Polish format, separated by semicolons. When the function processor encounters an operand specification (for example, 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) is the value retrieved.

Example 1 The operation "(1+2)*4=12" could be done with an F code as follows:

F;C4;C2;C1;+;*



Example 2 As a further example consider a PARTS file with dictionary and data items as follows:

Dictionary Item

```
  CODE
001 A
002 22
:
:
008 F;2;3;*;C*;:;1;C3;C2;[];:
009 L
010 10
```

Data Item

```
  1137
001 S*632-19
002 32
003 21
```

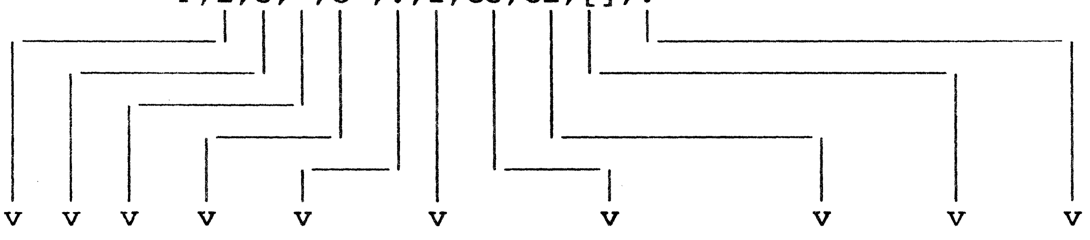
:LIST PARTS '1137' CODE

```
PAGE 1
PARTS..... CODE......
1137       632*672

1 ITEMS LISTED
```

In this example the F code is processed in the following way:

```
F;2;3;*;C*;:;1;C3;C2;[];:
```



ST1	32	21	672	*	*672	S*632-19	3	2	63	63*672
ST2		32		672		*672	S*632-19	3	*672	
ST3							*672	S*632-19		
ST4								*672		
ST5										
ST6										
ST7										

F Code Special Operands and 'Fn' Format

An F code operand may be multivalued, may contain a conversion specification(s), or may be a special two-character operand specifying one of several counters. The optional 'n' parameter converts numeric fields stored with decimal points (typically ALL-generated numeric fields) to integer values that can be handled by ENGLISH.

Operands

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

Example 1

```
F;10;11(T*SALES;X;;3);*
```

Places the data from attribute 10 onto the stack; picks up ID from attribute 11, translates it from dictionary of file 'SALES' and places it onto the stack; and multiplies the two values to give the result.

Example 2

```
F;D(D2/]G2/1);3(D2]G2/]);-
```

Computes the difference in the year fields of the system date and the date stored in attribute 3. D2/] converts the date from internal format to MM/DD/YYYY; G2/1] then isolates the year section of the date.] is actually a value-mark, CTRL].

Example 3

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

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

Embedded Decimal Points: 'Fn;' Format	The 'Fn;' correlative performs the desired function(s) on numbers stored internally with decimal points (typically ALL-generated numeric fields) and converts the resulting value to a scaled integer; this value can then be used by ENGLISH. The 'n' parameter is a scaling factor in the range 1 to 6, which multiplies each value by that power of ten before carrying out rounding. The following illustrates the difference between ENGLISH-stored and ALL-stored numbers:

	Stored by	
Actual Value	**ENGLISH**	**ALL**
100.23	10023	100.23
566.10000	566100000	566.1

For example,

```
F3;15;13;+
```

If attribute 15 contains 100.34 and attribute 13 contains 6.022, the resulting value is 106362 (a scaled integer). Note that numbers with varying decimal places are handled correctly, with either rounding or zero padding.

Summary of F Code Stack Operations

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

Element	Description	Action
amc	attribute (with conversion)	Push corresponding attribute value, after optional conversion, onto pushdown stack (maximum seven conversion levels): attribute value -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
Cn	constant	Push numeric or string constant "n" onto stack: "n" -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
D	date	Push numeric value representing current system date (internal form) onto stack: date -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 lost

Element	Description	Action
+	add	STACK1 + STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
-	subtract	STACK1 - STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
*	multiply	STACK1 * STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
/	divide	STACK1 / STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
R	remainder	remainder (STACK1/STACK2) -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2
S	sum	summation (STACK1) -> STACK1 Prior to this operation, STACK1 may be multivalued; this operator sums all those multivalues into a single value.
"	duplicate	STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
-	exchange	STACK1 <--> STACK2
^	POP	STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 -> STACK2 -> STACK1 -> lost
:	concatenate	STACK1:STACK2 -> STACK1, STACK7 -> STACK6 -> STACK5 -> STACK4 -> STACK3 STACK2
[]	extraction	STACK3[STACK2,STACK1] -> STACK3 (STACK2 = starting position, STACK1 = length), STACK4 -> STACK2 STACK5 -> STACK3, STACK6 -> STACK4, STACK7 -> STACK5
T	time	Push numeric value representing current system time (internal format) onto stack: time -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
NI	item counter	Push numeric value representing current item counter onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost

Element	Description	Action
ND	detail line counter	Push numeric value representing number of detail lines since the last control-break onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
NV	multivalue counter	Push numeric value representing current multivalue counter onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
NS	sub-multivalue counter	Push numeric value representing current sub-multivalue counter onto stack: counter -> STACK1 -> STACK2 -> STACK3 -> STACK4 -> STACK5 -> STACK6 -> STACK7 -> lost
=	equal	1) If STACK1 = STACK2 then 1 -> STACK1 2) If STACK1 # STACK2 then 0 -> STACK1 3) STACK7 -> STACK6 -> STACK5 -> STACK4 - > STACK3 -> STACK2
#	not equal	1) If STACK1 # STACK2 then 1 -> STACK1 2) If STACK1 = STACK2 then 0 -> STACK1 3) STACK7 -> STACK6 -> STACK5 -> STACK4 - > STACK3 -> STACK2
<	less than	1) If STACK1 < STACK2 then 1 -> STACK1 2) If STACK1 not < STACK2 then 0 -> STACK1 3) STACK7 -> STACK6 -> STACK5 -> STACK4 - > STACK3 -> STACK2
>	greater than	1) If STACK1 > STACK2 then 1 -> STACK1 2) If STACK1 not > STACK2 then 0 -> STACK1 3) STACK7 -> STACK6 -> 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) STACK7 -> STACK6 -> 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) STACK7 -> STACK6 -> STACK5 -> STACK4 - > STACK3 -> STACK2

GROUP EXTRACTION: G CODE

One or more contiguous segments of an attribute value may be extracted using the G code as a conversion or a correlative. The attribute value may consist of any number of segments separated by a common non-numeric separator (which may not be a minus sign or a system delimiter although a space character is valid).

The inverse of a G conversion code cannot be applied to values in ENGLISH sentences. Values are therefore compared directly with appropriate intermediate values.

Syntax

G{m}*n

where:

m	is the number of segments to skip; if omitted zero is assumed and retrieval begins with the first segment.
*	is the segment separator.
n	is the number of segments to be retrieved.

Examples

G Code	Attribute Value	Value Retrieved
G#1	ABC#DEF#GHI#JKL	ABC
G1#2	ABC#DEF#GHI#JKL	DEF#GHI
G2#1	ABC#DEF#GHI#JKL	GHI
G1A1	123A5555A22	5555
G2A1	123A5555A22	22

MASK CHARACTER: MC CODES

The MC codes perform character extractions and conversions. They may be used as conversions or correlatives.

The inverse of any MCDX and MCXD code only, is applied to values in an ENGLISH sentence.

The MC codes are as follows:

Code	Function
MCA	Extracts only the alpha characters.
MC/A	Extracts only the non-alpha characters.
MCN	Extracts only the numeric characters.
MC/N	Extracts only the non-numeric characters.
MCB	Extracts only the alpha and numeric characters.
MC/B	Extracts characters which are neither alpha nor numeric.
MCC;x;y	Changes all occurrences of string 'x' to string 'y'.
MCL	Converts all upper case characters to lower case.
MCU	Converts all lower case characters to upper case.
MCT	(Text). Converts all upper case characters to lower case starting with the second character in each word. Will also force the first character to upper case if necessary.
MCP	Converts all non-printable characters (X'00'-X'1F',X'80'-X'FF') to tildes (~).
MCPN	Same as MCP but whenever a non-printable character is converted to a tilde, the tilde is followed by the two character hex representation of the overwritten character.
MCDX	Converts a decimal value to its equivalent hex value.
MCXD	Converts a hex value to its equivalent decimal value.

Examples

MC Code	Stored Value	Converted Value
MCA	ABC123DEF	ABCDEF
MC/A	ABC123DEF	123
MCB	ABC;123/DEF	ABC123DEF
MCC;A1B;C2D	A1B567A1B	C2D567DC2D
MCU	abcDEF	ABCDEF
MCPN	ABCxDEFxGH where x=CTRL E	ABC~05DEF~05GH
MCDX	152	98

MASK DECIMAL: MD CODE	In order to save space and simplify sorts and selects, numbers are stored as integers. The MD code (which is almost always specified as a conversion) inserts decimal points, commas and/or currency symbols for output.
	An inverse MD conversion is applied to decimal values specified in an ENGLISH sentence to produce integer values for comparison with intermediate values.
Syntax	**MD{n}{m}{Z}{,}{£ or $}{i*}{c}**
	where:

n	is a single numeric digit defining the number of digits to be output (printed or displayed) following the decimal point. If n is not specified, 0 is assumed and no decimal point is output.
m	is a single numeric digit that defines the position of the decimal point. The decimal point is inserted so that m digits follow it. If m>n then the digits following the decimal point are rounded off. Although m is normally optional, if the 'i*' option is used and the Z, and £ (or dollar) options omitted, then m must be specified in order to prevent ambiguity.
Z	specifies the suppression of leading zeros. A zero is always output preceding the decimal point for values less than 1 and greater than -1.
,	causes commas to be inserted between every thousandths position of the value.
£ or $	causes a pound or dollar sign to precede the output value.
i*	causes the value to be overlaid on a field of i characters, * specifies the filler character and may be any nonnumeric (typically an asterisk or a blank to cause currency symbols to align).

c is a credit indicator and can 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 (credit) to follow negative values; two blanks to follow positive or zero values.
- < causes negative values to be enclosed in angle brackets (< >).

Examples

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

MASK FIELD: MF CODE

The MF code is normally used as a conversion and performs literal insertions and currency formatting.

Syntax

MF{s}{Z}{'j'}{i*}{c}P[{m{(n)}}{Y}]

The picture clause (the parameters between the square brackets) may be repeated; the brackets are not entered.

The elements of the MF conversion are as follows:

s	is an optional single numeric digit defining the scaling factor (as a power of 10) to allocate the assumed decimal point. Scaling is only performed on numeric data, in a right justified picture clause. (For example if s=3, the decimal point will be placed to the left of the third digit from the right ---1234 becomes 1.234.)
Z	is an optional literal that specifies a null will be printed if the stored data equals zero.
'j'	is an optional character (or characters) placed between single quotes that will be appended to the left of the converted value. This is typically used to append a pound, dollar, or other currency sign to the value.
i*	is an optional parameter that causes the converted value to be overlaid in a field of I characters. * specifies the fill character which may be any non-numeric character. If i* is used, s, Z, or 'j' must first be specified. If none of these three is desired, a zero must be entered directly after the MF code.
c	is an optional parameter used as a credit indicator. It may be one of the following:

-	Causes a negative sign to follow negative values and a blank to follow zero or positive values.
C	Causes the letters CR to follow negative values and two blanks to follow zero or positive values.

D	Causes the letters DB to follow negative values and two blanks to follow zero or positive values.
<	Causes negative values to be preceded by a < sign and followed by a > sign, and a single blank to follow zero or positive values.

P	is the code for the 'picture' clause which consists of the following elements:
m	is an optional parameter indicating the data type. It may be one of the following:
9	Numeric symbol. If a 9 is specified directly after the P code the picture clause is right justified.
A	Alphabetic character. If an A is specified directly after the P code the picture clause is left justified.
X	Alphanumeric character. If an X is specified directly after the P code the picture clause is right justified.

Note: The justification applies only within the picture clause itself. The converted value is output according to the justification specified in the attribute definition item.

(n)	is an optional one or two digit number which causes the 9, A or X to be repeated the specified number of times.
Y	is an optional parameter for character insertions. It may be one of the following:

.	Inserts a decimal point.
,	Inserts a comma.
"literal"	Inserts the literal specified.

If 'Y' is specified directly after the 'P' code, the parameter will be bypassed until a character symbol is found. This character symbol will be used to determine if the picture clause is left or right justified.

Taking into account left or right justification of the picture clause and of the data, characters in the leftmost positions of the picture clause and data are checked for matches. The next characters are then checked, and so on. If there are any mismatches the conversion is ignored and the data is output as an unconverted value. If the data value consists of fewer characters than are specified in the picture clause, the remaining clause symbols are ignored (that is, are assumed to match).

The inverse of any MF conversion code is NOT applied to values in an ENGLISH sentence.

Example

MF Code	Stored Value	Converted Value
MFP99	12	12
MFP9(3)	123	123
MFP99.99	0	0.00
MF<P9(6)	-123456	<123456>
MF'LRA'CP9,9(3).99	-123456	LRA1,234.56CR
MF'LRA'CP9,9(3).99	-1234567	-1234567
MF'LRA'CP9,9(3).99	-123	LRA1.23CR

MASK PACKED: MP CODE

The MP code (which is almost always specified as a conversion) converts packed decimal numbers into normal representation.

The inverse of the conversion is performed on decimal numbers specified in ENGLISH sentences to produce a value suitable for comparison with intermediate values.

Syntax

MP

Note: Packed decimal digits should always be unpacked for output. Packed values are not displayed on terminals in a recognisable format, also, many of these characters are terminal control characters.

TIME FORMATTING: MT CODE

In order to save space and simplify sorts and selects, times are normally stored as integers which represent numbers of seconds from midnight. The MT code (usually specified as a conversion) converts internally stored integer values back into an external form.

The inverse of the conversion is applied to values in an ENGLISH sentence. Illegal values are converted to null and AM/PM notation is ignored unless the H option is used.

Syntax

MT{H}{S}

where:

H	is optional and specifies 12-hour external format; AM is assumed if AM/PM is not specified. If omitted, 24-hour format is assumed.
S	is optional and specifies that seconds are required. If omitted, only hours and minutes are output.

Note: 12:00AM is considered midnight and 12:00PM is considered noon.

Negative values are output as nulls and other illegal values convert to 00:00.

Examples

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

MASK HEXADECIMAL: MX CODE

The MX code converts any character string to its hexadecimal ASCII equivalent. Each character in the string is converted to two hexadecimal ASCII characters. This code may be used as a conversion or a correlative.

A reverse MX conversion is applied to any values used in an ENGLISH statement for comparison with appropriate attributes. These values must consist exclusively of hexadecimal characters (0 to 9, A to F inclusive) for the reverse conversion to be applied.

Syntax **MX**

Examples

Stored Value	**Converted Value**
ABC	414243
ABC#	41424323
T	54
%T	2554
XYZ	58595A
56	3536
12	3132

TEXT EXTRACTION: T CODE

The T code extracts a contiguous string of characters from an attribute value.

The inverse of any T conversion code cannot be applied to values in ENGLISH sentences. The T code must, therefore, be processed as a correlative if such a comparison needs to be done.

Syntax

T{m,}n

where:

m is the optional starting column number from the left. If this is not specified then 1 is assumed and extraction starts with the first character from the left or right depending upon whether left or right justification is specified in the dictionary item.

n is the number of characters to be retrieved.

Examples

T Code	Attribute Value	Value Retrieved
T3,2	ABCDEFG	CD
T3,5	ABCDEFG	CDEFG
T2	CA92631	CA (left just.) 31 (right just.)
T9	ABCDEFG	ABCDEFG
T8,1	65439679	9

The T code can be used to save space in file items by allowing an attribute (or item-id) to contain different fixed length values. This can result in significant space savings for large files (and decreased processing time due to smaller items).

FILE TRANSLATION: Tfile CODE

The Tfile code, which can be used as a correlative or a conversion, allows more than one file to be accessed using a single ENGLISH statement. This avoids having to duplicate data.

The value to be translated (defined by the attribute definition item containing the Tfile code) is the item-id of an item in the defined translation file (the second file to be accessed). The number of the attribute to be retrieved from this item is specified in the Tfile code.

Syntax

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

where:

*****

is optional and indicates that the translation file to be accessed is a dictionary file.

file

is the name of the translation file to be accessed.

c

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

V	Conversion item must exist on file, and the specified attribute must have a value for conversion. If not then a message of the form '[708] tst CANNOT BE CONVERTED' is output.
C	If conversion item does not exist, or if specified attribute has no value, then the original value (which was to have been translated) is output.
X	If conversion item does not exist, or if specified attribute has no value, then use null value.
I	Input verify: functions as a C for output translation and a V for input translation.

O	Output verify: functions as a V for output translation and a C for input translation.

input-amc

For input translation only. After locating the translation file the value of the attribute with the amc specified in input-amc is retrieved. This is the item-id of an item in the original file. If this parameter is null, no input translation takes place.

output-amc

For output translation only. This is the number of the attribute to be accessed in the items of the translation file. If this parameter is null, no output translation takes place.

Note that in the above description 'output translation' refers to a translation to an intermediate or output value. 'Input translation' (which is only used when the code is specified as a conversion and appropriate translation items have been specified) refers to a translation from a value specified in an ENGLISH sentence to an intermediate value.

The additional items must have item-ids of the actual data values being translated to. Within each of these items one attribute (which is identified via 'input-amc' in the Tfile code) contains the translation value used in the translation file.

Example 1

Consider an ordering system. A CUSTOMERS file contains details of customers. The item-id of an item in this file is the customer account number, attribute 1 contains the customer name, attribute 2 contains the customer address, and so on.

A second file called ORDERS contains details of orders received. The item-id of an item in this file is the order number; attribute 1 contains the part number required and attribute 2 contains the number of parts required. Among other pieces of information required to be held in this file are details of the customer. Rather than having to repeat details already held in the CUSTOMERS file every time an order is made, it is much simpler to hold just the customer account number in, say, attribute 3 of the ORDERS data items.

This can then be used to translate to the CUSTOMERS file whenever customer details are required to be output.

The ORDERS CUST.NAME and CUST.ADDRESS dictionary items and sample ORDERS and CUSTOMERS data items are now shown.

```
                ORDERS                    |          CUSTOMERS
                                          |
Dictionary Items                          |
                                          |
CUST.NAME               CUST.ADDRS        |
001 A                   A                 |
002 3                   3                 |
003 CUSTOMER            ADDRESS           |
004                                       |
005                                       |
006                                       |
007                                       |
008 TCUSTOMERS;C;;1     TCUSTOMERS;C;;2   |
009 T                   T                 |
010 15                  15                |
                                          |
Data Items                                |
                                          |
A3725                   A3726             |      5872
001 A167                A219              |  001  CEP
002 3                   1                 |  002  35, High St..
003 5872                4176              |   :    :
 :   :                  :                 |   :    :
 :   :                  :                 |   :    :
                                          |
                                          |      4176
                                          |  001  Key Products
                                          |  002  2, River Way..
                                          |   :    :
                                          |   :    :
```

```
:LIST ORDERS CUST.NAME CUST.ADDRS PART.NO NO.REQ

ORDERS   CUSTOMER...... ADDRESS........ PT.NO NO.REQ

A3725    CEP            35, High St.,    A167       3
                        Harlow, Essex
A3726    Key Products   2, River Way,    A219       1
                        Ware, Herts

2 ITEMS LISTED
```

Example 2 Using the ordering system of the previous example, in order to be able to input sentences such as 'LIST ORDERS WITH CUST.NAME = "CEP", it is necessary to have a data item called CEP in the CUSTOMERS file. An attribute within this item must contain the customer account number and this attribute be identified via the input-amc of the CUST.NAME dictionary item.

SUBLISTS: V CODE

The V code is a special code which can only be specified in attribute 8 of the DL/ID (Data Level Identifier) of a file. It is used to designate an attribute within the data items as containing a sublist of related items.

In order to invoke the sublist feature, the file specified in a LIST or COUNT statement is immediately preceded by the word WITHIN, and only one item-id (that of the master item) must be specified. This causes the selection of the master item and all sublist items. Furthermore, if a sublist item itself has a sublist then those sublist items are also selected. Up to 20 sublevels may be specified in this way.

The LIST verb outputs a report consisting of a detail line for each item selected. Each detail line is preceded by a level number. Lines are listed so that every level of a sublist is shown before going to the breakdown of the next sublist.

The COUNT verb gives a count of the total number of items selected, including all intermediate items.

Syntax

V;;sub-list-AMC

where:

sub-list-AMC

is the attribute number containing the sublist.

Example

Consider an ASSEMBLIES file with each item containing details of parts related to a servo motor. These parts are as follows:

```
LEVEL 1                  Servo
                ----------|-------------
                |         |             |
                |         |             |
LEVEL 2     D.C.Motor  Servo Board  Servo Housing
               /\                       /\
              /  \                     /  \
             /    \                   /    \
LEVEL 3  D.C.Motor  D.C.Motor    Housing  Housing
         Platform   Power Unit    Seals   Plates
```

where LEVEL 1 item is made up of LEVEL 2 items, and LEVEL 2 items are made up of LEVEL 3 items as indicated.

Data items for the servo (level1), D.C. Motor (level2) and D.C. Motor Platform (level3) are as follows:

```
A200
 001 SERVO
 002 A201]A202]A203 <- Item-ids of D.C. Motor,
                       Servo Board and Servo
                       Housing separated by value
                       marks CTRL ].

A201
 001 D.C. MOTOR
 002 A210]A211 <- Item-ids of D.C. Motor Platform
 003 35             and D.C. Motor Power Unit

A210
 001 D.C. MOTOR PLATFORM
 002
 003 23
```

Dictionary items for the file are as follows:

```
     DL/ID      DESC         SUB.ASS  STOCK
001   D          S             S        S
002   30099      1             2        3
003   1          DESCRIPTION   SUB.ASS  STOCK
004
005
006
007
008   V;;2
009   L          L             L        R
010   10         16            7        5
```

Note that attribute 008 of the DL/ID item indicates that attribute 2 of each of the data items contains any sublist.

```
:LIST WITHIN ASSEMBLIES 'A200' DESC SUB.ASSEM
STOCK

PAGE 1

LEVEL ASSEMBLIES  DESCRIPTION..... SUB.ASS  STOCK

1           A200  SERVO               A201     74
                                      A202
                                      A203
2           A201  D.C.MOTOR           A210     35
                                      A211
3           A210  D.C.MOTOR PLTFM              23
3           A211  D.C.MOTOR P.U.               31
2           A202  SERVO BOARD                  17
2           A203  SERVO HOUSING       A212     18
                                      A213
3           A212  HOUSING SEALS                32
3           A213  HOUSING PLATES               20

8 ITEMS LISTED
```

Appendix A
ASCII, Hexadecimal and Decimal Table

DECIMAL	HEXADECIMAL	ASCII	DECIMAL	HEXADECIMAL	ASCII
000	00	NUL	053	35	5
001	01	SOH	054	36	6
002	02	STX	055	37	7
003	03	ETX	056	38	8
004	04	EOT	057	39	9
005	05	ENQ	058	3A	:
006	06	ACK	059	3B	;
007	07	BEL	060	3C	<
008	08	BS	061	3D	=
009	09	HT	062	3E	>
010	0A	LF	063	3F	?
011	0B	VT	064	40	@
012	0C	FF	065	41	A
013	0D	CR	066	42	B
014	0E	SO	067	43	C
015	0F	SI	068	44	D
016	10	DLE	069	45	E
017	11	DC1	070	46	F
018	12	DC2	071	47	G
019	13	DC3	072	48	H
020	14	DC4	073	49	I
021	15	NAK	074	4A	J
022	16	SYN	075	4B	K
023	17	ETB	076	4C	L
024	18	CAN	077	4D	M
025	19	EM	078	4E	N
026	1A	SUB	079	4F	O
027	1B	ESC	080	50	P
028	1C	FS	081	51	Q
029	1D	GS	082	52	R
030	1E	RS	083	53	S
031	1F	US	084	54	T
032	20	SPACE	085	55	U
033	21	!	086	56	V
034	22	"	087	57	W
035	23	#	088	58	X
036	24	£ or $	089	59	Y
037	25	%	090	5A	Z
038	26	&	091	5B	[
039	27	'	092	5C	\
040	28	(	093	5D	]
041	29	)	094	5E	^
042	2A	*	095	5F	_
043	2B	+	096	60	`
044	2C	,	097	61	a
045	2D	-	098	62	b
046	2E	.	099	63	c
047	2F	/	100	64	d
048	30	0	101	65	e
049	31	1	102	66	f
050	32	2	103	67	g
051	33	3	104	68	h
052	34	4	105	69	i

ASCII, Hexadecimal and Decimal Table

DECIMAL	HEXADECIMAL	ASCII	DECIMAL	HEXADECIMAL	ASCII
106	6A	j	159	9F	
107	6B	k	160	A0	
108	6C	l	161	A1	
109	6D	m	162	A2	
110	6E	n	163	A3	
111	6F	o	164	A4	
112	70	p	165	A5	
113	71	q	166	A6	
114	72	r	167	A7	
115	73	s	168	A8	
116	74	t	169	A9	
117	75	u	170	AA	
118	76	v	171	AB	
119	77	w	172	AC	
120	78	x	173	AD	
121	79	y	174	AE	
122	7A	z	175	AF	
123	7B	{	176	B0	
124	7C	|	177	B1	
125	7D	}	178	B2	
126	7E	~	179	B3	
127	7F	DEL	180	B4	
128	80		181	B5	
129	81		182	B6	
130	82		183	B7	
131	83		184	B8	
132	84		185	B9	
133	85		186	BA	
134	86		187	BB	
135	87		188	BC	
136	88		189	BD	
137	89		190	BE	
138	8A		191	BF	
139	8B		192	C0	
140	8C		193	C1	
141	8D		194	C2	
142	8E		195	C3	
143	8F		196	C4	
144	90		197	C5	
145	91		198	C6	
146	92		199	C7	
147	93		200	C8	
148	94		201	C9	
149	95		202	CA	
150	96		203	CB	
151	97		204	CC	
152	98		205	CD	
153	99		206	CE	
154	9A		207	CF	
155	9B		208	D0	
156	9C		209	D1	
157	9D		210	D2	
158	9E		211	D3	

DECIMAL	HEXADECIMAL	ASCII	DECIMAL	HEXADECIMAL	ASCII
212	D4		234	EA	
213	D5		235	EB	
214	D6		236	EC	
215	D7		237	ED	
216	D8		238	EE	
217	D9		239	EF	
218	DA		240	F0	
219	DB		241	F1	
220	DC		242	F2	
221	DD		243	F3	
222	DE		244	F4	
223	DF		245	F5	
224	E0		246	F6	
225	E1		247	F7	
226	E2		248	F8	
227	E3		249	F9	
228	E4		250	FA	
229	E5		251	FB	SB
230	E6		252	FC	SVM
231	E7		253	FD	VM
232	E8		254	FE	AM
233	E9		255	FF	SM

A

A code, 7-7etc.
 IF statement, 7-11
 operands, 7-8
 operators, 7-10
 remainder function, 7-12
 substring function, 7-12
 summation function, 7-12
A/AMC, 6-4
addition, 7-10
address labels, 3-22
algebraic functions, 7-7
ALL-generated numerics, 7-7
An format, 7-7
AND connective, 4-8
ASCII code, A-1
ASCII conversion, 7-41
associative attributes, 3-29,4-10,7-20etc.
attribute
 name, 7-8
 number, 6-4,7-8
attribute definition item, 6-4
 creation, 6-10
 structure, 6-4
automatic paging, 2-7

B

BEFORE operator, 4-8
BETWEEN operator, 4-8
BREAK-ON modifier, 5-11,7-4
BSELECT verb, 3-5
BY modifier, 3-29
BY-DSND modifier, 3-29
BY-EXP modifier, 3-29
BY-EXP-DSND modifier, 3-29

C

C code, 7-14
COL-HDR-SUPP modifier, 3-23, 5-14
column headings, 6-5
column width, 6-6
columnar output, 3-19
concatenation, 7-4,7-14
conventions, 1-3
conversions, 6-5,7-3etc.
 multiple, 6-5,7-3
conversions cont.
 summary, 7-6
COPY-LIST verb, 3-7
correlatives, 6-5,7-3etc.
 multiple, 6-5,7-3
 summary, 7-6
COUNT verb, 3-9

D

D code, 7-17
D/CODE, 6-4,6-6
D1 and D2 codes, 7-20
data file, 6-3
data storage, 7-3
DATA/BASIC, 3-5,3-12,3-27
date handling, 7-4,7-17
DBL-SPC modifier, 5-14
DD code, 7-18
default attributes, 6-6
DELETE-LIST verb, 3-10
delimiters, 3-17,3-25
DET-SUPP modifier, 5-14
DI code, 7-18
DICT, 2-5,2-8
dictionaries, 6-3
dictionary file, 6-3
 creation, 6-10
division, 7-10
DJ code, 7-18
DM code, 7-18
DMA code, 7-18
DQ code, 7-18
DW code, 7-18
DWA code, 7-18
DY code, 7-18

E

EACH modifier, 4-4
EDIT-LIST verb, 3-11
embedded decimal points, 7-7,7-28
equal to operator, 4-8,7-10
ESEARCH verb, 3-12
EVERY modifier, 4-4
expanded print, 5-7
exploded sort, 3-30
extra space, 2-8

F

F code, 7-22
file dictionary, 6-3
 creation, 6-10
file hashing statistics, 3-16,3-18
file translation, 7-43
file-name, 2-5
footings, 5-6
FORM-LIST verb, 3-14

G

G code, 7-31
GET-LIST verb, 3-15
GRAND-TOTAL modifier, 5-9
greater than operator, 4-8, 7-10
greater than/equal to operator, 4-8,7-10
group extraction, 7-31

H

HASH-TEST verb, 3-16
hashing statistics, 3-16, 3-18
HDR-SUPP modifier, 5-14
headings and footings, 5-6
 options, 5-6
hexadecimal code, A-1
histogram, 3-16,3-18

I

I-DUMP verb, 3-17
ID-SUPP modifier, 3-23,5-14
IF modifier, 4-4
IF THEN ELSE statement, 7-11
ISTAT verb, 3-18
item number, 6-4
item size, 6-6
item type, 6-4
item-list, 2-5,4-3

J

justification, 6-6

L

labelling totals, 5-9
less than operator, 4-8,7-10
less than/equal to operator, 7-10
list
 copying, 3-7
 creation, 3-5,3-12,3-14, 3-27,3-34
 deletion, 3-10
 editing, 3-11
 retrieval, 3-15
 saving, 3-26
 sorting, 3-33
 storage, 3-26
LIST verb, 3-19
LIST-ITEM verb, 3-21
LIST-LABEL verb, 3-22
logical connectives, 4-3, 4-4,4-8
long sentences, 2-8
LPTR modifier, 5-14

M

mask field conversion, 7-36
mask hexadecimal, 7-41
master dictionary, 6-3
mathematical functions, 7-7, 7-22
MC/A code, 7-32
MC/B code, 7-32
MC/N code, 7-32
MCA code, 7-32
MCB code, 7-32
MCC code, 7-32
MCDX code, 7-32
MCL code, 7-32
MCN code, 7-32
MCP code, 7-32
MCPN code, 7-32
MCT code, 7-32
MCU code, 7-32
MCXD code, 7-32
MD code, 7-34
MF code, 7-36
modifiers, 5-3etc.
modulo, 3-16
MP code, 7-39
MT code, 7-40
multiplication, 7-10

multivalued attributes,
3-11,3-23,3-27,3-29,4-10,
7-21
MX code, 7-41

N

N (NOPAGE) option, 2-7
NO modifier, 4-4
NO operator, 4-8
NOPAGE modifier, 5-14
not equal to operator, 4-8,
7-10
number conversion/scaling,
7-34

O

ONLY, 6-6
operands, 7-8
options, 2-7
OR connective, 4-8
output-modifier, 2-7
output-modifiers, 5-3etc.
output-specification, 2-6,
4-10

P

P (printer) option, 2-7
packed decimal, 7-39
precedence
logical connectives, 4-3,
4-8
operators in A code, 7-11
print limiting, 4-10
PROC, 3-5,3-13,3-27

R

READNEXT, 3-5,3-13,3-27
relational operators, 4-3,
4-4,4-8
report presentation, 5-3etc.
rules of ENGLISH, 2-8

S

S-DUMP, 3-25
SAVE-LIST verb, 3-26
SELECT verb, 3-27
selection-criteria, 2-6,4-4
sentence-construction, 2-5
separation, 3-16
SORT verb, 3-29
SORT-ITEM verb, 3-31
SORT-LABEL verb, 3-32
SORT-LIST verb, 3-33
sort-order, 2-6,3-29,5-3
SSELECT verb, 3-34
ST-DUMP verb, 3-35
STAT verb, 3-36
statistics
file hashing, 3-16,3-18
string searching, 3-12,4-6
sub-items, 3-9,4-3
sub-totals, 5-12
sublists, 3-9,3-19,4-3,
7-47etc.
subtraction, 7-10
SUM verb, 3-37
SUPP modifier, 5-14
system dictionary, 6-3

T

T code, 7-42
T-DUMP verb, 3-38
T-LOAD verb, 3-39
TAPE modifier, 4-13
text extraction, 7-42
Tfile code, 7-43
throwaway connectives, 4-13
time formatting, 7-40
TOTAL modifier, 5-9,7-5
totals
labelling, 5-9

V

V code (WITHIN), 7-47
V/CONV, 6-5
V/CORR, 6-5
V/MAX, 6-6
V/TYP, 6-5
verbs, 3-3etc.

W

WITH modifier, 4-4

WITHIN connective, 3-9,3-19,
 4-3,7-47

[

['end with' string, 4-6
[]'contain'string, 4-6

]

] 'start with' string, 4-6

^

^ mask character, 4-6