SMA STANDARD

SMA/RETRIEVAL Language Specification

SMA: 401

January 1988

SPECTRUM
MANUFACTURERS
ASSOCIATION

sma

SMA:4Ø1

SMA/RETRIEVAL LANGUAGE SPECIFICATION - DRAFT 3.5

-- NOTICE --

Published by
SPECTRUM MANUFACTURERS ASSOCIATION
974Ø Appaloosa Rd., Suite 1Ø4
San Diego, CA 92131

SMA:401
SMA/RETRIEVAL LANGUAGE SPECIFICATION - DRAFT 3.5

Foreword: This document provides a set of syntax definitions for the SMA/RETRIEVAL language. This language, provided by all Spectrum Manufacturers Association members, is a report generating process which enables quick and easy preparation of various listings and queries from the Data Base. The language uses a limited natural language sentence format for accessing the Data Base. This document is intended to serve as a guide to the preparation of SMA/RETRIEVAL language sentences. For details on any specific system, the user should refer to the manufacturer's reference manual.

RELEASE COPY

Spectrum Manufacturers Association
January 1988

RELEASE

SMA:401
SMA/RETRIEVAL LANGUAGE SPECIFICATION - DRAFT 3.5

The SMA Executive Board wishes to thank the following individules and organizations for their contributions to the preparation of this document:

D. Harman,	Systems Management, Inc.
C. Saunders,	Fujitsu Microsystems of America, Inc.
K. Hoppe,	Altos Computer Systems
I. Sandler,	CIE Systems
H. Eggers,	Mcdonnell Douglas Computer Systems Co.
C. Wilson,	General Automation, Inc.

Spectrum Manufacturers Association
January 1988

SMA/RETRIEVAL LANGUAGE SPECIFICATION - DRAFT 3.5

CONTENTS

Spectrum Manufacturers Association
January 1988

CONTENTS

1.Ø Scope

1.1 Specification Objectives:
It is the objective of this document to provide the user with a defined syntax of the SMA/Retrieval Language to enable preparation of SMA/Retrieval statements that can be moved from one system to another with maximum portability.

1.2 Inclusions:
This document includes all the commonly available verbs with syntactical representation to clearly define the results produced by each.

1.3 Exclusions:
Extensions to the SMA/Retrieval language have been implemented or proposed by several SMA members. This document excludes any such items which have not been adopted by the majority of the SMA member companies. For further details on any differences between manufacturer's systems, see the manufacturers documentation.

2.Ø Definitions

2.1 Names and Symbols:

The names for verbs and predefined words used in this document are used by all the SMA manufacturers. They can be easily changed for different languages without affecting the documented functionality.

The use of quotes (") and single quotes (') is required in the forms shown below.

The use of braces ({ }) means the included string is optional.

The use of ellipsis (...) means the preceding information can be repeated.

2.2 Structural Terms:

Item-Id -	A character string (maximum 48 characters) which uniquely identifies an item within a file. The item-id may include any characters except system delimiters. The use of blanks, single quotes, quotes, commas, and backslash characters may require special considerations.
Item Body -	The variable length character structure which makes up the information content of the item. It is composed of any number of attributes, values, and subvalues.
SM -	Segment Mark: Delimiter character used to mark the end of a data structure.
AM -	Attribute Mark: Delimiter character used to mark the end of an attribute. The last AM in an item is implied by the presence of a segment mark.
VM -	Value Mark: Delimiter character used to mark the end of a value. The last VM in an attribute is implied by an attribute or segment mark.

2.2 Structural Terms (continued):

SVM -	Sub-value Mark: Delimiter character used to mark the end of a subvalue. The last SVM in a value is implied by a value, attribute, or segment mark.
AMC -	Attribute Mark Count: The positional count, from 1, that locates a specific attribute within the body of an item. An AMC value of zero is used to locate the item-id.
VMC -	Value Mark Count: The positional count, from 1, that locates a specific value within a multivalued attribute.
SVMC -	Sub-value Mark Count: The positional count, from 1 that locates a specific subvalue within a multisubvalued structure.
BM -	Buffer Mark: Delimiter character used to mark the beginning or ending of special character strings.

3.0 SMA/Retrieval Language Sentence

3.1 SMA/Retrieval Language Sentence General Form:

The SMA/Retrieval Language invokes processing of data from a file according to specified criteria by the use of a single sentence.

The general form of the sentence takes up one logical line.

```
    verb   {file.modifiers} file.name
           {item.list/item.id.selection.criteria}
           {USING {DICT} file.name}
           {data.definition.item.selection.criteria}
           {sort.criteria}
           {output.criteria}
           {modifiers} {(option,...)}
```

Only the verb and file name are mandatory. Blanks are used as separators between the parts of the sentence.

The SMA/Retrieval Language comes with a standard set of verbs, modifiers, and relational operaters. These words are defined as items in the user's Master Dictionary (MD). The user may define any number of synonyms for these words, and remove the supplied entries, thereby creating his own semantics for the language. Thus the SMA/Retrieval language can be tailored to fit any language.

3.2 Verb:

A verb specifies what processing will be performed on the file. The verb must be the first word in the sentence.

3.3 File.Modifier:

Certain modifiers are valid as modifiers of the file name, and if present, must precede the file name.

3.3.1 DICT:

The file modifier DICT stipulates that the dictionary section of the named file contains the data to be operated on.

3.3.2 ONLY:

The file modifier ONLY stipulates that only the item-ids of the items shall be output, suppressing any existing "default" output.criteria.

3.4 File.Name:

The file name stipulates the file to be operated on by the verb. It also implies the dictionary to be used as the source of data.definition.items, unless the USING clause appears later in the statement.

3.5 USING:

The USING connective stipulates that the following named file is to be used as the source of data.definition.items during processing instead of the dictionary of the file to be processed. If the modifier "DICT" precedes the file.name then the dictionary of the file is to be used, otherwise the data portion of the file is to be used as the source.

3.6 Item.Lists

Either an Explicit or an Implicit item list is used for retrieving items from a data file for processing. If no explicit list of item-ids is specified, and an implicit.list is not active, all items in the file will be used by the verb. Only those items within the list, that are present in the file, will be used by the verb.

3.6.1 Explicit.Item.List:

The explicit.item.list is a list of item-ids, each of which is surrounded by single quotes ('), to be operated on by the particular verb.

If present, an explicit list takes precedence over an implicit list, and causes the implicit list to be ignored.

3.6.2 Implicit.Item.List

An implicit.item.list is a list which has been activated by a previous command such as SELECT or GET-LIST.

3.7 Selection.Criteria:

Items may be stipulated as members of the set output by the inclusion of the selection.criteria in the command.

A selection criterion may be either an item.id.selection.criteria or a data.definition.item.selection.criteria.

A selection criterion is made up of a relational operator and a value.string.

3.8 Item-id Selection.Criteria:

Selection.criteria are placed on the item-id by following the file.name with one or more relational.connective and character.string pairs, ANDed together as desired. There must be at least one relational.connective in the sequence of pairs, otherwise the collection of 'Item-id' and "character.string" elements will be treated as a list of explicit Item-ids.

Any 'Item-id' or "Character.string" elements not preceeded by a relational.connective will be taken to be "equal"tests.

The forms 'Item-id' and Character.string" have the same status in the list immediately following the file.name and preceeding the first data.definition.item and are called Item.id.values.

The Item-id selection.criteria form is:

```
      relational.connective item.id.value
         {{AND/OR} {relational.connective} item.id.value}...
      where:
            item.id.value is

                 'Item-id'
            or
                 "Character.string"
```

An item must pass the item-id selection.criteria before any other criteria are considered. Therefore, item-id selection.criteria are implicitly ANDed with all other selection.criteria, and an explicit AND is illegal after item-id selection.criteria.

3.8.1 ORing Item.Id.Values:

Several item.id.values in succession are taken to be ORed with implicit equal relational.connectives; that is, if an item-id matches any of the item.id.values, the item passes this criterion.

3.8.2 ANDing Item.Id.Values:

The AND evaluation connective may be placed between item.id.values. In this case, an item-id must pass all the tests stipulated by the relational.connective-item.id.value pairs for the item to pass this criterion. Note that use of the implicit equal relational.connective will define a criterion that can not be passed by any item-id, as an item-id can not be "equal" to two different values.

3.9 Data.Definition.Item Selection.Criteria:

Items may be selected based on the generated contents of stipulated data.definition.items.

Stipulation of data.definition.item selection criteria requires the use of the WITH connective.

The form of the data.definition.item selection criterion is

```
      WITH {EACH} {NO} data.definition.item.name
         {value.string.criteria} {{AND/OR} WITH {EACH} {NO}
         data.definition.item.name {value.string.criteria}}...
```

The WITH connective may be associated with the EACH and/or with the NO modifier. The effect of the inclusion of the EACH modifier is that each multi-value of the attribute stipulated by the data definition item must pass the value.string criterion. The effect of the NO modifier is to pass data.definition.items whose value is null if there is no value.string.criterion, or to pass those data.definition.items which <u>do not</u> pass the value.string criteria if one is present.

A command may contain at least 9 ORed data group selections.

3.9.1 ORing Data.Definition.Items:

A sequence of data.definition.item selection criteria will be implicitly ORed together if they are not explicitly ANDed together. If an item passes any one of ORed data.definition.item selection criteria it will be accepted.

3.9.2 ANDing Data.Definition.Items:

Inserting an AND between the end of a data.definition.item selection criterion and the beginning of another has the effect of ANDing the criteria together. Any number of data.definition.item selection criteria may be ANDed together to form a group. An item must pass all of the ANDed criteria in order to be accepted.

There may be as many as 9 of these groups of more than one data.definition.item selection criteria which have been ANDed together. These groups are implicitly ORed together.

3.1Ø Value.String.Criteria:

The form of a value.string.criterion is:

```
   {relational.connective} "Character.string" {{AND/OR}
      {relational.connective} "Character.string"} ...
```

If there is no relational.connective preceeding a "Character.string", then the test defaults to "equal". A data value will pass an ORed group of value.string criteria if it passes any one. A data value will pass an ANDed group of value.string criteria if it passes all entries in the group. The logic of evaluations with NOT follows as with the NOT evaluation for Item-id Selection Criteria, that is, the AND connective must be used to exclude more than one value.

3.1Ø.1 Relational Connectives:
The allowable relational.connectives are:

mnemonic	symbol	meaning
EQ	=	equal
NE	#	not equal
LT	<	less than
LE	<=or=<	less than or equal
GT	>	greater than
GE	>=or=>	greater than or equal

The lack of a relational.connective defaults to the EQ (=) relational.connective. Either the mnemonic or symbol may be used.

3.1Ø.2 Character String:
A character.string is a sequence of characters enclosed in quotes (").

Selecting part of a value.string can be performed by including any or all of the three following reserved characters in the character.string.

character	meaning
[	accept any leading characters or nulls.
]	accept any trailing characters or nulls.
^	accept any single character.

To use "[" in this special way it must be the first character in the character.string. To use "]" in this special way it must be the last character in the character.string.

The character "^" may occur anywhere in the string, and any number of times. Each case of "^" will match any single character in that location in the string.

These special "wild card" character meanings are not evaluated on elements that have EXTERNAL Operations defined, which are used as input conversions (see SMA:3Ø1).

3.10.3 ORing Value.Strings:

Several value.strings in succession are taken to be ORed; that is, if any one matches the referenced data from the item, the item passes this criterion.

3.10.4 ANDing Value.Strings:

The AND evaluation connective may be placed between value.strings. In this case, the referenced data must pass each of the tests stipulated by the relational connective-value string pair.

3.11 Sort.Criteria:

A sort criterion clause is made up of a sort activation connective and an associated data.definition.item specifier. The sort criterion applies to the sorting verbs; SORT, SSELECT, SREFORMAT, S-DUMP, SORT-ITEM, and SORT-LABEL.

If a sort.criteria is not specified on a sorting verb, the output of the command will be in the ascending order sorted sequence of the item-ids. Justification is determined by the File Definition Item (see SMA:301).

3.11.1 Sort Connectives:

The sort connectives for single-valued attributes are of the form:

BY element	- ascending order
BY-DSND element	- descending order

where "element" is a data.definition.item.

3.11.2 Exploding Sort Connectives:

The sort connectives for exploding multi-valued attributes, where each value is treated as an independent item by the sort.criteria, are of the form:

BY-EXP element {explosion.limiter}	- ascending order
BY-EXP-DSND element {explosion.limiter}	- descending order

where "element" is a data.definition.item and explosion.limiter is of value.string.critera form (see section 3.10).

3.11.3 Sort Evaluation:
The evaluation of a sort sequence criterion may be either alphabetic or numeric. It is considered alphabetic and left adjusted if the justification is not explictly declared to be 'right adjusted'. It will be evaluated from left to right.

The sort sequence is considered numeric if the justification is declared to be 'right adjusted'. The data will be evaluated in terms of numeric magnitude, including the sign. If value.string is alphanumeric, the numeric portions are sorted right-to-left, and the non-numeric portion is sorted left-to-right.

3.11.4 Multiple Key Sort:
Any number of sort criteria may be included in a statement. Each will contribute to the value used for sort purposes, with the first being used as the highest order sort value, and continues, in order, toward the end of the command, with the item-id always being used as the lowest-order sort value.

3.12 Output.Criteria:
Any data.definition.item not preceded by a selection connective or a sort connective is an output criterion.

The form of the output.criteria is:

```
{TOTAL} data.definition.item {print.limiter}

or

BREAK-ON data.definition.item {break.option.string}
```

The effect of an output.criteria is to emit the value.string generated by the data.definition.item into the output stream generated by the command.

If no output.criteria is specified and a set of default data.definition.items exist, they will be used for the output criteria. Default data.definition.items are those whose item.ids are sequential integers beginning with 1.

3.12.1 Print.Limiter Criteria:
Print.limiters are used with output.criteria to select certain values from multi-valued attributes for output. Output of values is limited to those values that meet specified criteria. Dependent values in associative data sets will be suppressed if the value they depend on is not output. A print.limiter criteria is of the same form as a value.string.criteria (see section 3.1Ø).

3.12.2 BREAK-ON Connective:

The output.criteria may be preceded by the BREAK-ON connective. This causes the stipulated output.criteria values in successive items to be monitored for change. When an item is encountered which contains a different value from the previous item, a control break is said to have occurred. When a control break is detected, then special actions are taken to indicate the detection in the output, including the inclusion of the break.option.string.

The form of the break.option.string is:

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

The permissible BREAK-ON options are:

Options	Meaning
B	BREAK. Insert the value of the data.definition.item in the page heading.
D	DATA. Suppresses the break line entirely if there was only one detail line since the last control-break occurred.
L	LINE. Suppresses the blank line before the break data line. This option is ignored when the 'U' option (below) is used.
N	Reset the page number to one on this break.
P	PAGE. Cause a page break after this break line has been output.
R	ROLLOVER. Inhibit page break until all data associated with this break has been output.
U	UNDERLINE. Causes the underlining of all specified TOTAL fields.

3.12.2 BREAK-ON Connective (continued):

V	VALUE. Causes the value of the control break to be inserted at this point in the BREAK-ON line.
''	Two successive single quotes are used to insert a single quote mark in the text.

The control break process prints the leading and trailing text, if any, and the value of the data before the break if the V option is specified.

If there is neither text nor a V option specified, the default output is three asterisks (***).

3.12.3 Multiple Control Breaks:
Control breaks are hierarchically ordered, with the first control break criteria as the highest break level, and continuing toward the end of the command.

The use of control breaks assumes a sort sequence based on the data.definition.items specified as control breaks and in the same hierarchy.

There are at least 15 control break levels.

3.12.4 TOTAL Connective:
If an output.criteria is preceded by the TOTAL connective, then the data elements are summed for the specified items to be output.

There is one running total kept for each control break output.criteria, and one for the complete operation.

Totals are output for each output.criteria preceded by a TOTAL connective at each control break and at the end of the output.

Non-numeric data is taken to be zero for the totaling process.

3.12.5 GRAND-TOTAL Modifier:

The GRAND-TOTAL modifier may be used with the TOTALs and/or BREAK-ONs to specify special formatting on the grand-total line.

The GRAND-TOTAL connective form is:

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

where text is output at the completion of the command, and the 'options' enclosed in single quotes specify the following:

Option	Meaning
L	Line suppress. Suppress the line before the grand-total line.
P	Page break. Force a page break before the grand-total line is displayed.
U	Underline. Display a line of equal-signs (=) in the totaled output.criteria column before displaying the grand-total line.
''	Two successive single quotes are used to insert a single quote mark in the text.

The grand-total literal string will be displayed, left-justified, starting in the first column.

3.13 Modifiers:

Modifiers change the form of the output format. Note that several of the modifiers can alternatively be specified as Options.

The admissible modifiers are:

Mnemonic	Meaning
COL-HDR-SUPP	Suppress the default heading, the column headings, and the end-of-file message.
DBL-SPC	Place a blank line between items.
DET-SUPP	Suppress detail lines.
HDR-SUPP	Suppress the default heading and the end-of-file message. The column headings are not suppressed.

3.13 Modifiers (continued):

HEADING	Provide page heading format.
ID-SUPP	Suppress output of default item-id column.
FOOTING	Provide page footing format.
LPTR	Send the output to the spooler.
NOPAGE	Do not pause at the bottom of each page when displaying to a terminal.
TAPE	Acquire data from a T-DUMP tape.

3.13.1 Heading and Footing Modifiers:

The HEADING and FOOTING modifiers must be followed by a string surrounded by quotes ("), of the form:

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

Each group of options must be surrounded by single quotes ('). The effect is the text associated with the HEADING modifier will appear as the heading on each page of output, as operated on by the options, and the text associated with the FOOTING modifier will appear as the footing on each page of output, as operated on by the options.

3.13.2 Heading and Footing Options:

The allowable options for the HEADING and FOOTING modifiers are:

Option	Meaning
B	Break. Insert the value causing the control break if the "B" option has been specified in a BREAK-ON connective literal.
C	Center. Causes the HEADING or FOOTING line to be centered on the output page.
D	Date. Insert the current date, dd mmm yyyy, in the heading at this point.

3.13.2 Heading and Footing Options (continued):

F	File name. Insert the name of the file being LISTed or SORTed.
L	New line. Specifies a new line in the HEADING or FOOTING.
P	Page number. Insert the current page number, right justified, in a field of four blanks.
T	Time and Date. Insert the time and date, hh:mm:ss dd mmm yyyy.
''	Two successive single quotes are used to insert a single quote mark in the heading text.string.

3.14 Options:

The options must be last in the sentence, are enclosed in parentheses, and may be separated by commas. The right (closing) parentheses is optional.

The allowable options are:

Option	Meaning
B	Suppress initial terminal line-feed prior to output.
C	Refer to Modifier COL-HDR-SUPP
D	Refer to Modifier DET-SUPP
H	Refer to Modifier HDR-SUPP
I	Refer to Modifier ID-SUPP
N	Refer to Modifier NOPAGE
P	Refer to Modifier LPTR

3.15 Connective Synonyms:

A set of common synonyms are provided for the various connectives in the language as follows:

Synonym	Standard Connective
&	AND
EVERY	EACH
IF	WITH
WITHOUT	WITH NO
AFTER	GT
BEFORE	LT
NOT	NO
SUPP	HDR-SUPP
HEADER	HEADING
CAPTION	GRAND-TOTAL

3.16 Throw-away Connectives:

A set of common words are included that are not used by the language but may be included by the user to make the sentence more readable, which are as follows:

Throw-aways

!
A
AN
ARE
DATA
FILE
FOR
IN
ITEMS
OF
OR
THE

4.Ø Verbs

4.1 LIST verb:

The SMA/Retrieval language verb, LIST, is used to generate a formatted output of selected items and attributes of a specified file.

The LIST verb has the following form:

```
     LIST     {file.modifiers} file.name
              {item.list / item.id.selection.criteria}
              {USING {DICT} file.name}
              {data.definition.item.selection.criteria}
              {output.criteria{print.limiters}}
              {modifiers}{(option,...)}
```

See the appropriate sections for further details on each part of the sentence.

In the absence of an active item list, the output sequence of the LIST verb will be unordered. With an active item list, either explicit or implicit, the output sequence will be in the order given by the list.

4.2 SORT Verb

The SMA/Retrieval language verb, SORT, is used to generate a sorted and formatted output of selected items and attributes of a specified file.

The SORT verb has the following form:

```
     SORT     {file.modifiers} file.name
              {item.list / item.id.selection.criteria}
              {USING {DICT} file.name}
              {data.definition.item.selection.criteria}
              {sort.criteria}
              {output.criteria{print.limiters}}
              {modifiers}{(option,...)}
```

See the appropriate sections for further details on each part of the sentence.

4.3 SELECT verb:

The SMA/Retrieval language verb, SELECT, creates a temporary implicit-list of the selected elements for later usage.

The SELECT verb has the following form:

```
     SELECT   {file.modifiers} file.name
              {item.list / item.id.selection.criteria}
              {USING {DICT} file.name}
              {data.definition.item.selection.criteria}
              {output.criteria}
```

If there are no output.criteria specified, the item-ids are stored in a temporary implicit-list for use by the next verb as an implied 'item.list'. If the next verb is a SAVE-LIST verb, then the temporary implicit-list is saved. (See the section on SAVE-LIST).

If output.criteria are specified, the value of the specified attribute(s) will be saved in the implicit-list. Each value of a multi-valued attribute is treated as if it were in a single-valued attribute. The item-ids will not be saved in the select-list when output.criteria are specified.

The output from the SELECT verb is a temporary implicit-list and a message specifying the number of elements selected.

4.4 SSELECT verb:

The SMA/Retrieval language verb, SSELECT, creates a sorted temporary implicit-list of the selected elements from a file, for later usage.

The SSELECT verb has the following form:

```
     SSELECT  {file.modifiers} file.name
              {item.list / item.id.selection.criteria}
              {USING {DICT} file.name}
              {data.definition.item.selection.criteria}
              {sort.criteria}
              {output.criteria}
```

The output of the SSELECT verb is the same as the SELECT verb, with the exception of the order.

4.5 COUNT Verb:

The SMA/Retrieval language verb, COUNT, will count the number of items in a file meeting the criteria as specified by the combination of item.list and/or selection.criteria.

The COUNT verb has the following form:

```
     COUNT    {file.modifiers} file.name
              {item.list / item.id.selection.criteria}
              {USING {DICT} file.name}
              {data.definition.item.selection.criteria}
              {(option,...)}
```

A message is displayed showing the number of those items meeting the specifications of the item.list and/or the selection.criteria if present. If neither are specified, then the number displayed is the number of items in the specified file.

4.6 SUM Verb:

The SMA/Retrieval language verb, SUM, will generate the sum of the data elements of the items specified by the selection.criteria.

The SUM verb has the following form:

```
     SUM      {file.modifiers} file.name
              {item.list / item.id.selection.criteria}
              {USING {DICT} file.name}
              {data.definition.item.selection.criteria}
              data.definition.item
              {(option,...)}
```

The output produced by the SUM verb includes the output title for the data.definition.item and the computed total.

4.7 STAT Verb:
The SMA/Retrieval language verb, STAT, generates a set of statistics for data elements in the items of a file.

The STAT verb has the following form:

```
     STAT      {file.modifiers} file.name
               {item.list / item.id.selection.criteria}
               {USING {DICT} file.name}
               {data.definition.item.selection.criteria}
               data.definition.item
               {(option,...)}
```

The output of the STAT verb includes the output title for the data.definition.item, the generated sum of all data elements, the average of the data elements (total divided by count), and the number of items used in the generation of the statistics.

4.8 Reformat Verbs:
The SMA/Retrieval Language verbs, REFORMAT and SREFORMAT, are equivalent to the LIST and SORT verbs, except that the output is directed to another file or to tape, instead of a terminal or printer.

The REFORMAT and SREFORMAT verbs have the following forms:

```
     REFORMAT     {file.modifiers} file.name
                  {item.list / item.id.selection.criteria}
                  {USING {DICT} file.name}
                  {data.definition.item.selection.criteria}
                  {output.criteria{print.limiters}}
                  {modifiers}{(option,...)}

                    .... and ....

     SREFORMAT    {file.modifiers} file.name
                  {item.list / item.id.selection.criteria}
                  {USING {DICT} file.name}
                  {data.definition.item.selection.criteria}
                  {sort.criteria}
                  {output.criteria{print.limiters}}
                  {modifiers}{(option,...)}
```

After the REFORMAT or SREFORMAT sentence is entered, the system will prompt for:

FILE NAME:

The name of the file where the output is to be stored, or the word 'TAPE' if the output is to go to tape, must be entered.

4.8.1 Reformatting To Another File:
When reformatting into another file, the first value specified in the output.criteria is used as the item-id for the item, and the remaining values in the output.criteria are attributes in the item. Each item selected becomes an item in the new file.

4.8.2 Reformatting To The Same File:
When reformatting to the same file, the first value specified in the output.critera is used as the item-id for the item, and the remaining values in the output.criteria are attributes in the item. Each item selected becomes an additional item in the file. On a REFORMAT of a file onto itself, an implicit or explicit item list must be defined; otherwise, an infinite loop in which items are added to the file may occur.

4.8.3 Reformatting To Tape:
When reformatting a file to tape, the values specified in the output.criteria are concatenated together to form one tape record for each item that is selected. The record output is either truncated or padded at the end with nulls (hex 'ØØ's) to obtain a record the same length as specified by the last T-ATT verb.

A tape label which contains the file name, tape record length (in hex), the time and date, is written on the tape first, unless the HDR-SUPP or COL-HDR-SUPP modifiers or the options H or C are specified. Two End-Of-File marks (EOF's) terminate the file on tape. The item-id's will be displayed as the items are dumped unless the ID-SUPP modifier or the I option is specified.

4.8.4 Reformatting with an Exploding Sort Connective:
If an Exploding Sort Connective is used for the first data.definition.item specified in the output.criteria, then an item is created in the new file for each exploded entry. The remaining output.criteria attributes are repeated for each exploded entry.

4.9 Label Verbs:
The SMA/Retrieval Language verbs, LIST-LABEL and SORT-LABEL, are used for printing mailing labels or other special purpose listings.

Functionally, the LIST-LABEL and SORT-LABEL verbs are almost identical to the LIST and SORT verbs, except that the data associated with each item is grouped into a block by the LIST-LABEL and SORT-LABEL verbs, and several blocks can be placed across each page of a listing.

The LIST-LABEL and SORT-LABEL verbs have the following forms:

```
LIST-LABEL  {file.modifiers} file.name
            {item.list / item.id.selection.criteria}
            {USING {DICT} file.name}
            {data.definition.item.selection.criteria}
            {output.criteria{print.limiters}}
            {modifiers}{(option,...)}

              .... and ....

SORT-LABEL  {file.modifiers} file.name
            {item.list / item.id.selection.criteria}
            {USING {DICT} file.name}
            {data.definition.item.selection.criteria}
            {sort.criteria}
            {output.criteria{print.limiters}}
            {modifiers}{(option,...)}
```

After the sentence is entered, an additional set of parameters is prompted for with a question mark (?) until a null line of data ([CR]) is entered. The first additional parameter must be entered and has the following format:

```
?count,row,skip,indent,size,space{,C}
```

These parameters determine the arrangement of attribute values into blocks and are entered in the following format:

Parameters	Meaning
count	The number of items (labels) across each page.
rows	The number of lines printed for each label (height of each label, in ...rows)
skip	The number of lines to skip between labels (vertical spacing between labels, in rows)

4.9 Label Verbs (continued):

Parameters	Meaning
indent	The number of spaces to indent the the data from the left margin.
size	The maximum width permitted for the data associated with each attribute name (width of each label, in columns)
space	The number of horizontal spaces between labels in columns
C	Optional; if present, specifies that null attributes are not to be printed. If the C is not specified, null values will be printed as all blanks.

The values must conform to the following range:

```
(count * size) + ((count-1) * space) + indent <= (page width)
```

where 'page width' is the number defined for the output device (terminal or spooler).

Otherwise the system will respond with an error message indicating that an invalid numeric parameter was entered.

The normal non-columnar list heading (page number, time and date) will print on the top of each page unless suppressed by the COL-HDR-SUPP modifier or (C) option. If headings are suppressed, pagination and all top-of-forms are suppressed, then a continuous forms structure without page breaks is produced.

A set of row header data lines will be requested, immediately following the first parameter request, if the 'indent' parameter is non-zero. The parameter 'rows' specifies how many row headers will be requested because one row header is printed for each row of the label. The row headers will be printed in the 'indent' area of the left-hand margin. Null headers may be specified by entering null lines ([CR]) to the header data requests.

4.1Ø Item Listing Verbs:

The SMA/Retrieval language verbs, LIST-ITEM and SORT-ITEM, copy a complete item to the terminal or spooler.

The LIST-ITEM and SORT-ITEM verbs have the following forms:

```
     LIST-ITEM    {file.modifiers} file.name
                  {item.list / item.id.selection.criteria}
                  {USING {DICT} file.name}
                  {data.definition.item.selection.criteria}
                  {modifiers}{(option,...)}

                    .... and ....

     SORT-ITEM    {file.modifiers} file.name
                  {item.list / item.id.selection.criteria}
                  {USING {DICT} file.name}
                  {data.definition.item.selection.criteria}
                  {sort.criteria}
                  {modifiers}{(option,...)}
```

The data from each attribute in the item, with a three digit number in the left margin, is copied to the terminal or spooler.

The permissible LIST-ITEM and SORT-ITEM verb options are:

Option	Meaning
F	Causes a Form-Feed for each item. Starts a new page for each item.
N	Inhibits the pause at the end of each page when the output is to the terminal (NOPAGE).
P	Sends the output to the spooler (LPTR).
S	Suppresses the three digit line number in the left margin.

The equivalent modifiers for the options are shown in parentheses above.

4.11 FILE-TEST Verb:

The SMA/Retrieval language verb, FILE-TEST, provides a means of generating statistics for a specified file, or testing a new configuration for a specified file.

The FILE-TEST verb has the following form:

```
     FILE-TEST    {file.modifiers} file.name
                  {item.list / item.id.selection.criteria}
                  {USING {DICT} file.name}
                  {data.definition.item.selection.criteria}
                  {modifiers}{(option,...)}
```

where 'file.name' is the name of the file for which statistics are to be generated.

After the FILE-TEST sentence has been entered, the system may prompt for one or more parameters which may be used to define an alternate configuration for the file. If a null return is supplied to the prompt, then the current file configuration parameters will be used.

The statistics that are generated for the selected items will include; the count of items, the total number of bytes in all the items, and the average number of bytes per item. In addition there may be statistics relevant to the structure of the specified file. This structural information may be presented in different ways, as in the case of a hashing structure with the histogram, the average number of items per group and standard deviation, and the average number of bytes per group.

If no 'item.list' or 'select.criteria' are specified, then all the items in the file will be used for generating the statistics.

The option (S) causes the detail information to be suppressed, and only the statistical data to be printed.

4.12 CHECK-SUM Verb:

The CHECK-SUM verb generates a checksum for file items.

The CHECK-SUM verb has the following form:

```
     CHECK-SUM   {file.modifiers} file.name
                 {item.list / item.id.selection.criteria}
                 {USING {DICT} file.name}
                 {data.definition.item.selection.criteria}
                 {data.definition.item}
                 {(option,...)}
```

The checksum is calculated as an arithmetic total, disregarding overflow, of all the bytes in the selected data elements. If a data.definition.item is specified, then only the values of the specified data elements will be used in the calculation.

The result is presented in a message indicating the number of items checked and the checksum calculated.

4.13 Tape Verbs And The Tape Modifier:

The SMA/Retrieval language verbs, T-DUMP and T-LOAD, provide the facility to write the items of a specified file to tape, or to load items into a specified file from a previously generated T-DUMP tape. The TAPE modifier may be used with other SMA/Retrieval language verbs to access data from a T-DUMP tape.

4.13.1 Tape Dumping Verbs:

The T-DUMP verb will dump the specified items from a specified file to tape. The S-DUMP verb will also dump specified items from a file to tape, except that the items will be sorted before they are dumped. The tape drive must be first attached to the users account via the T-ATT command. The T-ATT command is also used to set the record length to be used by the T-DUMP or S-DUMP verb.

The T-DUMP and S-DUMP verbs has the following form:

```
T-DUMP      {file.modifiers} file.name
            {item.list / item.id.selection.criteria}
            {USING {DICT} file.name}
            {data.definition.item.selection.criteria}
            {HEADING "text"}
            {modifiers}{(option,...)}

and

S-DUMP      {file.modifiers} file.name
            {item.list / item.id.selection.criteria}
            {USING {DICT} file.name}
            {data.definition.item.selection.criteria}
            {HEADING "text"}
            {sort.criteria}
            {modifiers}{(option,...)}
```

T-DUMP and S-DUMP cause a standard tape label to be written on the tape. If the optional HEADING modifier and 'text' are specified, the 'text' is added to the label. See SMA:2Ø1 the SMA/Data Interchange Standard.

If the optional file.modifier 'DICT' is specified, the dictionary section of the specified file will be dumped with the exception of File Definition Items. A File Mark indicator is written on the tape at the end of the dump.

The HDR-SUPP modifier or (H) option may be used to suppress the writing of the tape label.

The ID-SUPP modifier or (I) option may be used to suppress the listing of item-ids that are dumped.

A message is displayed when the dump is finished indicating the number of specified items dumped to tape.

4.13.2 T-LOAD Verb:

The T-LOAD verb loads the specified items into a specified file. The tape must be attached before the T-LOAD verb is used.

The T-LOAD verb has the following form:

```
     T-LOAD       {file.modifiers} file.name
                  {item.list / item.id.selection.criteria}
                  {USING {DICT} file.name}
                  {data.definition.item.selection.criteria}
                  {modifiers}{(option,...)}
```

The T-LOAD verb will read the SMA standard tape label if present and setup the tape record length from the label. If the tape is unlabeled or has a non-standard label, the record length must be set using the T-ATT verb.

The items, as restricted by the item.list or the selection.criteria, are loaded into the specified file if they do not exist in the file. The (O) option will permit the overwriting of items that exist in the file. If the item exists in the file and the (O) option was not specified, a message is displayed indicating the item-id of the item that exists on the file.

Any items existing in the file and not existing on the tape will be maintained.

Specifying the ID-SUPP modifier or the (I) option will suppress the listing of the item-ids during the loading of the specified file.

A message which indicates the number of items loaded will be displayed when the specified file has been loaded.

4.13.3 TAPE Modifier:

The TAPE modifier can be used to read data from a T-DUMP tape rather than the data portion of the specified file. The TAPE modifier can only be used with the verbs: LIST, LIST-LABEL, LIST-ITEM, SUM, STAT, FILE-TEST, or COUNT. The dictionary of the specified file will be used for the specified dictionary.definition.items.

4.14 List File Handling Verbs:

The SMA/Retrieval language provides a set of verbs for the saving, editing, copying, retrieving, forming, and deleting selected item.lists.

4.14.1 SAVE-LIST Verb:

The verb, SAVE-LIST, makes a stored item.list from a temporary implicit-list produced by the SELECT, SSELECT, and FORM-LIST verbs.

The SAVE-LIST verb has the following form:

```
SAVE-LIST {{DICT} file.name} list.name
```

The SAVE-LIST verb saves the temporary implicit-list and adds or updates the pointer to the list in the file POINTER-FILE, if file.name is not specified, with an item-id of list.name.

If file.name is specified, the list's pointer will be added or updated in the specified file. The file definition item of the specified file must declare that the file can hold indirect data pointers.

An existing stored list with the same name will be overlaid by the newly stored list.

The SAVE-LIST verb displays a message showing the list.name and the number of frames used to store the list.

The SAVE-LIST verb must be issued immediately after the creation of a temporary implicit-list in order to create a stored list.

4.14.2 GET-LIST Verb:

The verb, GET-LIST, retrieves a previously stored list and forms a temporary implicit-list.

The GET-LIST verb has the following form:

```
GET-LIST {{DICT} file.name} list.name
```

The list.name specifies which stored list is to be retrieved. If file.name is specified, the list is retrieved from the specified file. The POINTER-FILE is used if file.name is not specified.

If the specified item 'list.name' does not exist, a message indicating such is produced. If the item is found, a message is displayed showing the number of entries in the list and the temporary implicit-list is available for use.

4.14.3 DELETE-LIST Verb:
The verb, DELETE-LIST, deletes a stored item.list.

The DELETE-LIST verb has the following form:

```
DELETE-LIST {{DICT} file.name} list.name
```

The list.name specifies the stored list to be deleted. If the item does not exist, a message indicating such is produced.

If the item is found, a message is displayed stating that the list was deleted.

The POINTER-FILE is used if file.name is not specified.

4.14.4 FORM-LIST Verb:
The verb, FORM-LIST, will generate a temporary implicit-list from attribute(s) within an item or items in a file.

The FORM-LIST verb has the following form:

```
FORM-LIST {DICT} file.name {item.list} {(n)}
```

where the data is taken from the item(s) in the specified file. The item.list can be implicit, explicit, or an asterisk (*) specifying all items. All data from the items are stored in a temporary implicit-list unless the optional (n) specification is used; in which case, only data from the n-th attribute of each item is used. Multi-values or sub-values are stored as separate elements in the implicit-list.

A message indicating the number of list entries formed is displayed at the conclusion of the process.

THIS IS THE LAST PAGE