

"Made available under NASA sponsorship
in the interest of early and wide dis-
semination of Earth Resources Survey
Program information and without liability
for any use made thereof."

FUNCTIONAL DESIGN SPECIFICATION
FOR
ENHANCEMENT OF THE AUTOMATIC STATUS
AND TRACKING SYSTEM SOFTWARE

Job Order 71-695

(TIRF 77-0035)

(E80-10190) FUNCTIONAL DESIGN SPECIFICATION
FOR ENHANCEMENT OF THE AUTOMATIC STATUS AND
TRACKING SYSTEM SOFTWARE (Lockheed
Electronics Co.) 60 p HC A04/MF A01

N80-28788

CSSL 05B G3/43

Unclas
00190

Prepared By

Lockheed Electronics Company, Inc.
System and Services Division
Houston, Texas
Contract NAS 9-15200

For

EARTH OBSERVATIONS DIVISION
SCIENCE AND APPLICATIONS DIRECTORATE



National Aeronautics and Space Administration
LYNDON B. JOHNSON SPACE CENTER

Houston, Texas

September 1977

LEC- 11199

T77-14242#
JSC- 13110
8.0-10190
NASA CR-
160624

JSC-13110

FUNCTIONAL DESIGN SPECIFICATION
FOR
ENHANCEMENT OF THE AUTOMATIC STATUS
AND TRACKING SYSTEM SOFTWARE

Job Order 71-695

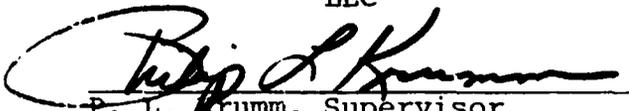
(TIRF 77-0035)

Prepared By

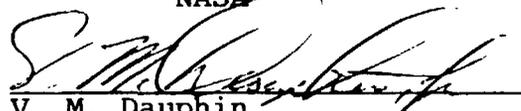
D. K. McCarley
J. M. Everette
K. P. Eckel

APPROVED BY

LEC


P. L. Krumm, Supervisor
Applications Software Section

NASA


V. M. Dauphin
Systems & Facilities Branch

Prepared By

Lockheed Electronics Company, Inc.

For

Earth Observations Division

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

September 1977

LEC-11199

CONTENTS

Section	Page
1. INTRODUCTION.	1-1
2. REFERENCES	2-1
3. FUNCTIONAL MODIFICATIONS.	3-1
3.1 <u>ADDITIONAL DATA BASE PROTECTION</u>	3-1
3.1.1 MODIFICATIONS TO DELETE SET, DELETE RECORD, DELETE KEY NAME, NO KEY	3-1
3.2 <u>NULL SET DETECTION AND CONTROL TRANSFER.</u>	3-2
3.2.1 TEST AND JUMP COMMAND	3-2
3.2.2 LABEL	3-2
3.3 <u>ARITHMETIC OPERATORS AND INTERDATA BASE COMPARISON</u>	3-3
3.3.1 SELECT NON-KEY MODIFICATION	3-4
3.3.2 CHANGE FIELD MODIFICATION.	3-4
3.3.3 DISPLAY FORMATTED AND JOINT DISPLAY FORMATTED MODIFICATIONS.	3-4
3.3.4 JOINT SELECT-NON KEY	3-5
3.3.5 JOINT SORT.	3-6
3.4 <u>SUBGROUPING BY FIELD WITH MAXIMUM, MINIMUM AND COUNT FUNCTIONS.</u>	3-6
3.4.1 REPORT COMMAND	3-6
3.4.2 JOINT REPORT COMMAND	3-8
4. PROGRAM DESIGN	4-1
4.1 <u>GENERAL</u>	4-1
4.2 <u>GENERAL FLOW FOR COMMANDS USING ARITHMETIC EXPRESSIONS</u>	4-3
4.3 <u>NEW SUBROUTINE DESCRIPTIONS.</u>	4-7

Section	Page
4.4 <u>DESCRIPTION OF SUBROUTINE MODIFICATIONS.</u> . . .	4-28
4.5 <u>DESCRIPTION OF NEW SYSTEM TABLES</u>	4-32

Appendix	Page
A NEW AND MODIFIED RIMS COMMANDS SYNTAX.	A-1

FIGURES

Figure		Page
4.1-1	Command Subroutine Cross Reference	4-2
4.2-1	General Flow for CF, JN, SN, JF, DF, JP, and RP Commands	4-4

1. INTRODUCTION

This document provides a functional design for enhanced data management capabilities of the LACIE Automated Status and Tracking System (ASATS) as requested in Reference 2. ASATS was implemented on the PDP 11/45 using the Regional Information Management System (RIMS), a generalized data base management system. The requested enhancements will be made to RIMS. They include:

- a. Additional Data Base Protection - In order to prevent inadvertent destruction of the data base, additional user interaction to verify the user's desire to execute the command is requested for certain data base update commands.
- b. Null Set Detection and Control - In order to prevent production of headers for reports containing no data, a test and jump command is requested.
- c. Arithmetic Operators - The ability to allow arithmetic operations on fields of data for certain operations is requested.
- d. Interdata Base Comparisons and Arithmetic - RIMS currently has the ability to generate sets by specifying arithmetic relationships between fields and literal values. The ability to specify arithmetic relationships between fields is requested. These relationships should allow the use of arithmetic operators and allow the comparison of fields between a FLOCON record and its parent DAPTS record.
- e. Subgrouping by Field with Maximum, Minimum and Count Functions - The ability to specify fields for which records are to be grouped by value and print field values, maximum or minimum field values, or count of records for the resulting groups is requested. The capability should include multi-level groupings.

Reference 9 identifies new RIMS commands to be added and existing RIMS commands which will be modified to provide the required

1.1
/

enhancements. Section 3 of this document describes the new commands and changes to existing commands.

Section 4 of this document identifies new subroutines and modifications to existing subroutines for satisfying the commands described in Section 3.

2. REFERENCES

The following documents provide a baseline for overall development and implementation of the enhancements to the Automatic Status and Tracking System:

- Reference 1 - Implementation Specification for LACIE ASATS, JSC-11401, Revision A.
- Reference 2 - TIRF 77-0035.
- Reference 3 - Operators Guide for LACIE ASATS, JSC-12729.
- Reference 4 - Users Guide for ASATS, JSC-12535, Revision A.
- Reference 5 - Users Guide for RIMS, LEC-9301, Revision A.
- Reference 6 - "As-Built" Design Specification for ASATS, JSC-12743, Revision A.
- Reference 7 - Functional Design for ASATS, JSC-11835.
- Reference 8 - RIMS Maintenance Document, LEC-9566.
- Reference 9 - Project Development Plan for the Enhancement of the Software of the LACIE Automatic Status and Tracking System.

3. FUNCTIONAL MODIFICATIONS

This section describes the new RIMS commands and modifications to existing RIMS commands. The description of the new report commands in section 3.4 assumes an understanding of arithmetic operators described in section 3.3. A detailed description of the syntax for the new and modified commands is given in Appendix A.

3.1 ADDITIONAL DATA BASE PROTECTION

Certain update commands are occasionally executed accidentally. In such cases, the data base has to be restored. In order to save the lost time, an additional data base protection feature requiring input verification for certain commands will be added.

3.1.1 MODIFICATIONS TO DELETE SET, DELETE RECORD, DELETE KEY NAME, NO KEY

Purpose of Modification: To require verification for execution of commands.

Input Modifications: Syntax of these commands is unchanged. But before execution, each command will require verification in one of the following forms:

- A "Y" or "N" in or after column 60 of the command line.
- A YES or NO on the card image following the command in the command file.

Output Modifications: If "Y" or "N" does not appear after column 59 of the command line, the command is echoed to the report file with "YES or NO" appended.

Processing Modifications: If "YES" does not appear in one of the two input forms, the command will not be executed.

3.2 NULL SET DETECTION AND CONTROL TRANSFER

In order to provide a method of eliminating headers and other data not desired when a null set is encountered, two commands are being implemented. These two commands are Test and Jump, and Label; together they will provide the null set detection and transfer capability.

3.2.1 TEST AND JUMP COMMAND

Purpose of Modification: To transfer past a group of RIMS commands if the specified set is null.

Input: JTSN,LA

where: JT is the command mnemonic
SN is the set number
LA is a two character Label

Output: None

Processing: If the specified set contains entries, no action is taken otherwise, commands are read and ignored until the specified label is found on a label command (LA). If the set number is zero, the command will be an unconditional jump.

3.2.2 LABEL

Purpose of Modification: To provide a location in the command file to which a jump instruction may transfer.

Input: LALB

where: LA is the command mnemonic
LB is the specified two character label

Output: None

Processing: Acts as a NOP.

3.3 ARITHMETIC OPERATORS AND INTERDATA BASE COMPARISON

In order to support arithmetic operators and interdata base comparison, four commands will be modified and two new commands will be implemented.

Arithmetic expressions containing arithmetic operators will be allowed. Arithmetic operators supported will include: addition (+), subtraction (-), multiplication (*), and division (/). Arithmetic operators will be separated by field names or literal values; e.g., A+B*2 where A and B are field names and 2 is a literal value. Literal types supported include: (1) dates and (2) integers. Arithmetic expression will be allowed in replacement clauses and relational clauses. Alphanumeric strings may be used with replacement and relational clauses, but not with arithmetic operators.

Replacement and relational clauses are currently allowed for certain RIMS commands. But their use is restricted to replacing a field value with an alphanumeric literal or comparing a field value with an alphanumeric value. The inclusion of arithmetic expressions, as previously defined, with relational clauses provides for interdata base comparison.

Interdata base comparison is defined in reference 7 as the comparison of fields within a record or the comparison of fields between two hierarchically related records. Field names in arithmetic expressions may be related to any hierarchically related record for joint type operations, operations which operate on multiple levels (DAPTS and FLOCON for ASATS).

3.3.1 SELECT NON-KEY MODIFICATION

Purpose of Modification: To provide for generation of sets based upon arithmetic relationship of fields.

Input Modifications: New type relational clauses rather than old type.

Output Modifications: None.

Processing Modifications: Computations associated with arithmetic operators of relational clauses will be performed.

3.3.2 CHANGE FIELD MODIFICATION

Purpose of Modification: To provide for arithmetic operators in replacement clauses and allow use of relational clauses.

Input Modifications: ● New form of replacement clause instead of old form
● New form of relational clause

Output Modifications: None.

Processing Modifications: Only those records satisfying all relational clauses will be updated. Computations will be performed for arithmetic operators on both relational and replacement clauses.

3.3.3 DISPLAY FORMATTED AND JOINT DISPLAY FORMATTED MODIFICATIONS

Purpose of Modification: To provide for arithmetic operations on output and conditional output.

Input Modifications: ● Relational clauses
● Replacement clauses

Output Modifications: None.

Processing Modifications: Only records which satisfy all relational clauses will be output. Output records will include computational results for replacement fields of replacement clause. Note: Replacement fields do not have to be data base fields; but they have to be specified in the output format.

3.3.4 JOINT SELECT-NON KEY

Purpose: To create sets from hierarchically related records based on the contents of the data, interdata base comparisons, and/or arithmetic relationships.

Input: JNSN, RC, RC₂, ... RC_N,
where: JN is the command mnemonic
SN is the set number
RC is a relational clause (new type)

Output: Set of records satisfying all relational clauses.

Processing: The following operations will be performed for each record in the input set.

- Get record ID from input set
- Retrieve record from data base
- Save needed data
- Retrieve parent record
- Save needed data
- Perform arithmetic operations for relational clauses
- Evaluate relationships
- Place input record ID for accepted records in resulting set

3.3.5 JOINT SORT

Purpose: To produce an ordered set of records based upon the contents of specified fields in the records of the input set and their hierarchically related records. This command will allow the user to order a set of FLOCON records based upon the contents of fields in either or both the FLOCON records and their associated DAPTS record.

Input: JSSN, F1, F2 ..., FN

where: JS is the command mnemonic

SN is the input set number

F1, F2, ... FN are field names in any level of the data base hierarchy.

Output: An ordered set. Same set number as the input set.

Processing: Each record in the input set and its parent record will be retrieved. A sort file will be constructed using field values from the specified fields and the lower level record I.D. After the sort file has been built using all input records of the set, it will be ordered. Record I.D.s will then be retrieved from the "sort file" and placed in the input set.

3.4 SUBGROUPING BY FIELD WITH MAXIMUM, MINIMUM AND COUNT FUNCTIONS

3.4.1 REPORT COMMAND

Purpose: To generate reports based upon report specifications and the data base contents.

Input:

1. Command mnemonic (RP)

2. Set number used to generate report
3. Grouping Fields, fields for which associated parametric data is printed on change of value
4. Fields to be printed
5. Expression results to be printed
6. Text values to be printed
7. Formatting and type conversion for date and integer calculations
8. Functions to be computed; summation, maximum value, minimum value, and count of occurrences for specified fields.

Output: Report results on report file.

Processing: Report data will be produced on a change of value for a grouping field. It may include any combination of the following:

- a. Value of specified fields
- b. Arithmetic expression results
- c. Literal values (specified constant)
- d. Results of a function

Maximum value, minimum value, count, and summation functions will be included for specified fields.

Field values and expression values will be derived and printed from the record where the associated field change for the grouping field occurs. All records within the group are used in deriving function results. Hence, function results are printed when the subsequent field value change for the grouping field occurs. Literals may be included at either place. If no lower level group exists, the derivation of all data for a group appears to occur simultaneously.

When a grouping field value change occurs, all lower level grouping fields will be treated as if their grouping field value changed. Up to five levels of grouping fields will be supported.

Formatting of data on the report is accomplished automatically. It is a function of the specified field grouping and the specified data list for each group. Data for each grouping field is output in the order specified from left to right. Two spaces will be inserted between each output field.

The relative positions of data for each group will be determined by the order that the grouping fields are specified. They are placed from left to right with two spaces in between.

3.4.2 JOINT REPORT COMMAND

Purpose: To generate a report for a given set of FLOCON records using data from both the FLOCON records and their associated DAPTS records according to a report specification. The report specification indicates grouping criteria, computations to be performed, and information to be printed.

Input: (Same as Report except for command mnemonic (JP).)

Output: (Same as report)

Processing: The processing is identical to the report command except that when a record from the input set is retrieved, the parent (DAPTS) record must also be retrieved and merged with the child record to form a combined record.

4. PROGRAM DESIGN

4.1 GENERAL

Subroutines which comprise the RIMS system can generally be categorized as (1) system control routines, (2) individual command control routines, (3) common routines and (4) elementary routines. The system control routine is JLASYS. Usually, command control routines exist for each RIMS command. Common routines are used for those functions common to several commands. Elementary routines perform those functions common to all of RIMS such as string manipulation and data base I/O.

The new data base protection feature will be incorporated by defining two routines which are referenced from the system control routine. The command will be verified before executing the appropriate command control routine.

The Jump Test and Label commands will be implemented by two command control routines referenced from the system control routine.

The Joint Sort command will be implemented by modifying the current sort command control routine.

All other new and modified commands must be able to handle arithmetic operators. Several new subroutines common to these commands have been defined for interpreting command lines, building internal tables and formats, and performing arithmetic operations. While these commands have many common functions, a separate control routine will exist for each command and its associated joint command. Section 4.2 describes the general approach for implementing commands which require arithmetic operations. Figure 4.1-1 is a command/subroutine cross reference table illustrating which subroutines are called by each of these commands having arithmetic operators.

FIG. 4. 2- BOX NO.	COMMAND			
	CF	JN, SN	JF, DF	JP, RP
1	AEINIT AEPR APSINT* DTEINT FTCMP FTFMT RLCLPR SETINI* SQZE XXINI*	AEINIT AEPR DTEINT FTCMP FTFMT PRNTID RLCLPR SETINI* SETOUT* SQZE XXINI*	AEINIT AEPR BLDTBF DTEINT FTCMP FTFMT LODFMT* PRNTID RLCLPR SETINI* SQZE XXINI*	AEINIT AEPR CIRP DTEINT FTCMP FTFMT PRNTID SETINI* SQZE XXINI*
2	GETREC* XXINI*	GETREC* XXINI*	GETREC* XXINI*	GETREC* XXINI*
3	TFORMW	PRNTID TFORMW	PRNTID TFORMW	PRNTID TFORMW
4	DTEINT EXCMD	DTEINT EXCMD	DTEINT EXCMD	
5	APSTUP* DTEINT EXCMD		DTEINT EXCMD	DTEINT EXCMD
6	A. STUP* DTEINT TFORMZ		DTEINT TFORMZ	DTEINT TFORMZ
7	REPR*	XXOUT*	DISFMT*	
8	XXINI*	XXINI*	XXINI*	XXINI*
9	APSCNT* AUPOST*	ENDSET*		

* EXISTING
SUBROUTINE

Figure 4.1-1.- Command Subroutine Cross Reference.

4.2 GENERAL FLOW FOR COMMANDS USING ARITHMETIC EXPRESSIONS

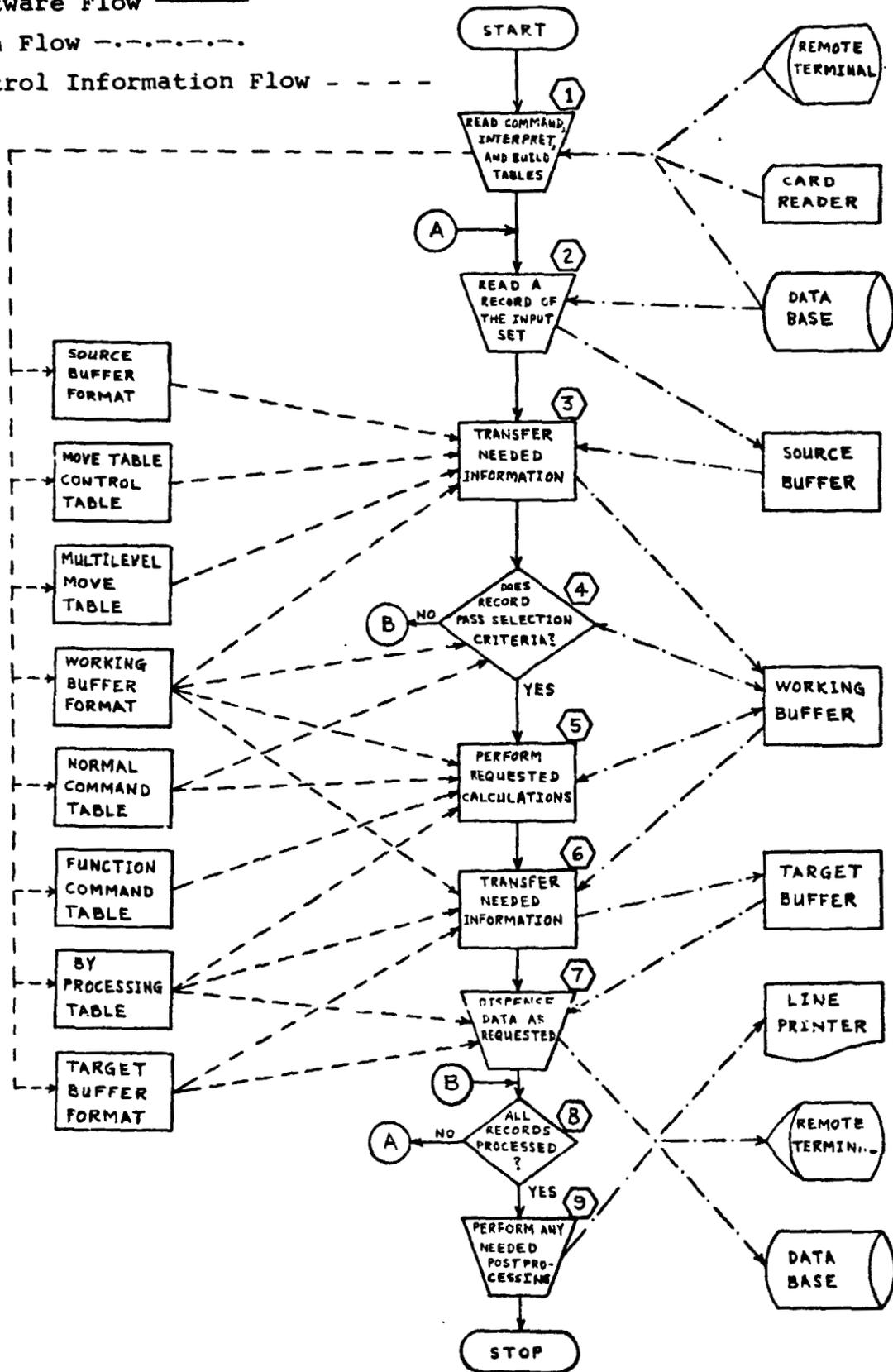
Figure 4.2-1 diagrams the flow of software, data, and control information in the RIMS commands that are related through the provision for arithmetic expressions in their syntax. In box 1, the original command is read from the terminal or cards, and all blanks except those between quote marks are deleted. The input set number and output format number (if any) are extracted for later use. Then the clauses of the command (portion between adjacent commas) are interpreted to build tables and formats used later in the processing. Relational clauses, which specify selection criteria, are processed to build field names and literal references into the working buffer format and to build a sequence of commands in the normal command table which, when executed, will tell whether a particular record passes the selection criteria or not. Replacement clauses, which cause the results of a calculation to be displayed or stored in an existing field of a record, are processed to build the field names and literal references into the working buffer format and to build a sequence of commands in the normal command table which, when executed, will provide the results to be displayed or stored. BY clauses from the report command are processed to build field names and literal references into the working buffer format and to build information concerning group and subgroup boundaries and related processing into the BY processing table. Function references in report command clauses are processed to build the appropriate commands into the function command table.

After processing the input command, the formats of the input records and their ancestors (if needed) are retrieved from the data base and used to build the source buffer format. At this point, information such as field type and length is available and is used to complete portions of the working and target buffer formats. As the format at each level of a data base is processed,

Software Flow ———

Data Flow - - - - -

Control Information Flow - - - - -



GENERAL FLOW FOR CF, JN, SN, JF, DF, JP, AND RP COMMANDS

FIGURE 4.2-1

entries are made into the multilevel move table to indicate fields to be transferred from the source to the working buffer, and an entry is made into the move table control table to specify which portion of the multilevel move table is to be used at that particular data base level. Then either the record format itself, the specified output format, or the automatically generated format is used to build the target buffer format. The working and target buffer formats are finally completed by the generation of field locations within their respective buffers.

Each iteration through box 2 causes the next record of the input set to be read from the data base and placed in the source buffer. Box 3 uses the portion of the multilevel move table specified by the move table control table to transfer the proper fields from the source to the working buffer. If a "joint" command is being processed and the input record has a new parent record, then successive iterations are made between boxes 2 and 3 to get all the required fields from a data base hierarchy. Boxes 4 and 5 are essentially the process of executing the commands previously built into the normal command table. The failure of a relational clause will be trapped in box 4 and will cause the input record to be ignored. For the report command, box 5 will process the function command table (maintaining intermediate results in the working buffer) and, as directed by the BY processing table, may process only portions of the normal command table to produce results needed at that moment.

Box 6 transfers data from the working buffer to the target buffer, directed by the target buffer format (and the BY processing table for the report command), and converting data from binary form to character strings where needed. Box 7 sends the data in the target buffer to the appropriate place depending on the command being executed. The change field

command stores revised records in the data base, whereas the display formatted and report commands send data to the terminal or printer. After box 8 determines that all records of the input set have been processed, box 9 handles such final processing as revising the file of key values for the change field command and building and displaying a status table entry for the select non-key command.

4.3 NEW SUBROUTINE DESCRIPTIONS

This section contains all new subroutines. They are listed in alphabetical order by subroutine name. SORTS, the control routine for SORT, and all command control routines for existing commands now allowing arithmetic operators required major redesign; therefore, they are listed as new subroutines.

SUBROUTINE NAME: Arithmetic Expression Command Initialization,
AEINIT

PURPOSE: To initialize standard areas of core for commands
which allow arithmetic expressions in their syntax.

INPUT:

- Calling arguments
- Command type indicator
- Common block SYSCØM
- input command string

OUTPUT:

- Common block SY3CØM
- initialized counters and arrays
- packed command string
- Calling arguments
- set number to be used for input
- format number to be used for output

PROCESSING: Zeroes are stored in such data locations as last
used row number for format arrays and field lengths in buffer
formats. The command line is compacted into its permanent array,
and if the command type indicator says this is a JP or RP
command, more input lines are read, if needed, and transferred.
The set number is converted to an integer from the command
line, as is the format number for JF or DF commands.

SUBROUTINE NAME: Arithmetic Expression Processor, AEPR

PURPOSE: To parse an arithmetic expression made up of arithmetic operators and operands (field names, dates, or integer constants), entering operands (or pointers to them) into the working buffer format and building a sequence of internal commands to evaluate the expression and store the value into a specified result variable.

INPUT:

- Common block SY3CØM
- packed input command string
- Calling arguments
- location and length of expression in command line
- start point in command table
- start point in working buffer format
- results location in working buffer format

OUTPUT:

- Common block SY3CØM
- normal command table
- working buffer format
- Calling argument
- syntax error indication

PROCESSING: The expression is scanned, and as elements are recognized, they are listed in the working buffer format to enable later reference to them. As subexpressions are recognized, commands to evaluate them are stored in the normal command table using a compilation algorithm which will retain the meaning of the original expression. The last command of the sequence generated will cause the value of the expression to be stored in the specified results area in the working buffer. Any illegal syntax will stop further action by this subroutine and cause the output syntax error indicator to be set.

SUBROUTINE NAME: Build Target Buffer Format, BLDTBF

PURPOSE: To convert a data base format into a form suitable for standardized processing.

INPUT:

- Common block SY2CØM
- Format array
- Common block SY3CØM
- Working buffer format
- Calling argument
- page number of format array in SY2CØM

OUTPUT:

- Common block SY3CØM
- Target buffer format

PROCESSING: Information about each field listed on the input page number of the format array in SY2CØM is extracted and stored in the target buffer format. Instead of field name being stored in the target buffer format, a pointer back to the corresponding field in the working buffer format is stored, thereby constructing a built-in move table.

SUBROUTINE NAME: Change Field Control Routine, CFCR

PURPOSE: To direct the overall processing sequence for the Change Field (CF) command.

INPUT:

- Common block SYSCØM
- input command line
- status table

OUTPUT:

- Revised records in the data base
- Revised key field values in the data base

PROCESSING: An interpretation phase parses the command line and accesses the record format to construct internal command tables and buffer formats used in processing the input set of records. Then a processing phase reads each record of the set, transfers needed fields to the working buffer, and executes the internal commands to determine if the record passes the selection criteria. If it does not pass, the record is ignored. If it does pass, the appropriate fields are modified as specified in the command, and the record is stored back into the data base.

A final phase deletes all the old key field values and adds the new key field values to the data base key value files. The key field values were saved on scratch files as each record was processed.

SUBROUTINE NAME: Command Interpreter for RP and JP commands,
CIRP

PURPOSE: To direct the activities of parsing the command line,
building tables, and building buffer formats.

INPUT:

- Common block SY3CØM
- command string
- Calling argument to specify which command (JP or RP) is being processed

OUTPUT:

- Common block SY3CØM
- BY processing table
- normal command table
- function command table
- working buffer
- working buffer format
- target buffer format
- source buffer format

PROCESSING: Each character string between pairs of commas in the input command is processed sequentially. If it is a BY clause, appropriate entries are made in the BY processing table. If it is a report expression, appropriate entries are made in the command tables and buffer formats. After the entire command has been processed, buffer formats are completed with field start, length, and type information.

SUBROUTINE NAME: Data Base Protection, DBPRØ

PURPOSE: To verify if command is to be executed.

INPUT: ● Common blocks SYSCOM and SY2COM
● A command line containing "YES" or "NO"

OUTPUT: ● An indicator signifying whether or not command was verified (1 for yes, 2 for no)

PROCESSING: Upon entry to the routine, the command line is checked for a "Y" or "N". If present, the indicator is set accordingly and the subroutine returns control to the calling routine. If "Y" or "N" are not present, the command line is echoed with a tag of "YES" or "NO". The user must then respond with a "YES" or "NO". If a "Yes" is entered, the indicator is set to "1", if a "No" or "CR" is entered, the indicator is set to "0".

A record is then read from the command file. If a "Y" is in the line verification indicator is set indicating acceptance, otherwise, it is set to indicate non-acceptance.

SUBROUTINE NAME: Date/Integer Conversion, DTEINT

PURPOSE: To convert date format to/from binary integer format.

INPUT:

- Calling arguments
- Function indicator
- Binary integer, if function indicator ≠0
- Character string, if function indicator =0

OUTPUT:

- Calling arguments
- Binary integer, if function indicator =0
- Character string, if function indicator ≠0

PROCESSING: Based on the value of the function indicator, a conversion is made between an alphanumeric character string in date format (YDDD) and an integer in binary format (suitable for use in computations). The year specified in the date format is related to a base year. An internal table will specify the base year and the number of days per year for the base year and the nine-year period following.

SUBROUTINE NAME: Execute Command, EXCMD

PURPOSE: To perform the operations specified in one row of a command table.

INPUT:

- Common block SY3CØM
- working buffer
- working buffer format
- intermediate storage registers
- Calling arguments
- command table name
- command table row number

OUTPUT:

- Common block SY3CØM
- working buffer
- intermediate storage registers
- Calling arguments
- error indicator
- comparison flag

PROCESSING: The operands are taken from the working buffer or intermediate storage registers and the operation is performed as specified at the input row number of the input command table. The error indicator is set non-zero if the calculation cannot be performed due to the absence of data in an operand field. The comparison flag is set to a logical value of .FALSE. if the operator is a relational operator and the comparison fails. Dates and numeric character integers are converted to and from binary integers based on the type specified in the working buffer format.

SUBROUTINE NAME: Format Completion, FTCMP

PURPOSE: To calculate starting character positions for fields in generated formats.

INPUT:

- Calling arguments
- Format array name
- number of spaces desired between fields

OUTPUT:

- Calling argument
- Format array name

PROCESSING: The assumption is made that the first field of the format starts in character 1 of its associated buffer. The length of the field and the input number of spaces between fields is added to the start character to give the start character for the next field. This process is continued for all fields (except \$T fields, where the text is maintained in the command line) until a length of zero is encountered as a signal to stop processing. In addition to the above, any \$R field encountered is initialized with the contents of word one of that row of the format.

SUBROUTINE NAME: Family Tree Format, FTFMT

PURPOSE: To retrieve information from formats associated with records in the same family tree.

INPUT:

- Calling argument
- record ID
- number of data base levels
- Common block SY3CØM
- working buffer format

OUTPUT:

- Common block SY3CØM
- working buffer format
- source buffer format
- multilevel move table
- move table control table

PROCESSING: The input record ID is used to retrieve the format number and then the format associated with the records on that data base level. The field names already in the working buffer format are compared with the names in the record format. For each match, (1) the field length and type are entered into the working buffer format, (2) an entry is created in the source buffer format, and (3) an entry is created in the multilevel move table. After all matches have been processed an entry is created in the move table control table. Then the parent of the input record is accessed for the next higher level of the data base, and the process is repeated to the number of levels that was input.

SUBROUTINE NAME: Joint Display Formatted and Display Formatted
Control Routine, JDFDCR

PURPOSE: To direct the overall processing sequence for the
Joint Display Formatted (JF) and the Display Formatted (DF)
commands.

INPUT:

- Common block SYSCOM
- input command line
- status table
- Calling argument to specify which command (JF or DF) is being processed.

OUTPUT: Reformatted records on the report file.

PROCESSING: An interpretation phase parses the command line and accesses the record format, its parent record format, and the output format to construct internal command tables and buffer formats used in processing the input set of records. Then a processing phase reads each record of the set (and its parent when needed), transfers needed fields to the working buffer, and executes the internal commands to determine if the record passes the selection criteria. If it does not pass, the record is ignored. If it does pass, the remainder of the internal commands are executed to perform the calculations specified in the command line and the record is written to the report file in the specified output format.

SUBROUTINE NAME: Joint Select Non-key and Select Non-key
Control Routine, JNSNCR

PURPOSE: To direct the overall processing sequence for the Joint
Select Non-key (JN) and Select Non-key (SN) commands.

INPUT:

- Common block SYSCØM
- input command line
- status table
- Calling argument to specify which command
(JN or SN) is being processed.

OUTPUT:

- Common block SYSCØM
- new status table entry
- Display of new status table entry
- New set of record pointers on a data base scratch
file.

PROCESSING: An interpretation phase parses the command line and
accesses the record format and its parent record format to
construct internal command tables and buffer formats used in
processing the input set of records. Then a processing phase
reads each record of the set (and its parent when needed),
transfers needed fields to the working buffer, and executes
the internal commands to determine if the record passes the
selection criteria. If it does not pass, the record is ignored.
If it does pass, then the record pointer is added to a collection
on a scratch file.

After all records have been processed, a final phase creates a
status table entry for the new collection of record pointers
and displays the entry.

SUBROUTINE NAME: Joint Report and Report Control Routine, JPRPCR

PURPOSE: To direct the overall processing sequence for the Joint Report (JP) and Report (RP) commands.

INPUT:

- Common block SYSCØM
- input command line
- status table
- Calling argument to specify which command (JP or RP) is being processed.

OUTPUT: Information written to the report file based on groups and subgroups of records in the input set.

PROCESSING: An interpretation phase parses the command line and accesses the record format and its parent record format to construct internal command tables, to build buffer formats, and to build a table defining the grouping and subgrouping characteristics of the input set. Then a processing phase reads each record of the set (and its parent when needed) and transfers needed fields into the working buffer. A check is then made to see if the processing is at a group or subgroup boundary. If it is, then results for the previous group are printed and reinitialized, and requested fields for the new group are printed. Then processing continues as if this were not a boundary, with the computation of requested functions such as MAX, MIN, and CØUNT. All the records of the input set are thus processed.

SUBROUTINE NAME: Parent ID, PRNTID

PURPOSE: To return the record ID of the next higher level record in the same family tree (inverted tree logical structure of the data base).

INPUT:

- Calling argument
- Record ID of child

OUTPUT:

- Calling arguments
- Record ID of parent
- Top data base level indicator

PROCESSING: Using an algorithm that is application dependent, the parent record ID is calculated from the child record ID. If the parent record is located at the top level of the data base, the output indicator is set to a non-zero value.

For ASATS, the child record (FLOCON) ID is the segment number concatenated with the acquisition date. The parent record (DAPTS) ID is generated by setting the acquisition date portion to zero. The DAPTS records are at the top level of the data base.

SUBROUTINE NAME: Relational Clause Processor, RLCLPR

PURPOSE: To parse a selection clause of the form AE.ØP.AE (where AE is an arithmetic expression, and ØP is a comparison operator) and build a table of commands to evaluate the clause.

INPUT:

- Common block SY3CØM
- packed input command string
- Calling arguments
- location and length of clause in command line
- start point in command table
- start point in working buffer format

OUTPUT:

- Common block SY3CØM
- normal command table
- working buffer format
- Calling argument
- syntax error indication

PROCESSING: The arithmetic expression portions of the clause are passed to an arithmetic expression processing subroutine which builds commands needed to evaluate the expression and return a result. Pointers to the results from each arithmetic expression are built into a command with the appropriate comparison operator, and the command is added to the normal command table. Elements needed for calculations will have been entered into the working buffer format by the arithmetic expression processor.

SUBROUTINE NAME: Sort Routine, SORTS

PURPOSE: To order a set according to a specified input list.

INPUT:

- Set number for temporary set to be reordered.
- A variable containing the number of entries in the input list.
- A list containing the sort parameters in descending order.
- A flag to indicate what type of sort to do (Joint or Regular).
- Common SYSCOM and SY2COM

OUTPUT: The reordered temporary set.

PROCESSING: For all sort types, the child records of the set are retrieved. In addition for JOINT sorts, the parent records are also retrieved. The entries in appropriate fields are compared with the sort list. Matching field values and their associated child record ID are placed on a sort data file. When all records in the set have been processed, the SSORT subroutine is called to sort the sort data file. Finally, the record ID's of each entry in the sort data file is returned to the original set in the new order.

SUBROUTINE NAME: Squeeze, SQZE

PURPOSE: To delete extraneous blanks from a character string.

INPUT: An array containing a string of characters.

OUTPUT: Another array containing the input characters but without extraneous blanks, and a comma location array.

PROCESSING. Each character in succession is examined in the input array. If it is not a blank, it is transferred to the next available character position in the output array. If it is a blank, it is bypassed. This process continues for all the characters in the input string or to the end of the output string, whichever comes first, except when a single or double quote character is encountered. Then all characters are transferred until the single or double quote character, respectively, is encountered again, at which time normal blank deletion resumes. The location of each comma encountered (and a possible final exclamation point) is stored in a comma location array for use in the interpretation phase of a command.

SUBROUTINE NAME: Transform to Working Buffer, TFØRMW

PURPOSE: To transfer data from a source buffer to the working buffer for later manipulation by other routines.

INPUT:

- Common block SY2CØM
 - the source buffer
- Common block SY3CØM
 - the working buffer
 - the working buffer format
 - the source buffer format
 - the multilevel move table
 - the move table control table
- Calling arguments
 - the source buffer row number
 - the move table control table row number

OUTPUT:

- Common block SY3CØM
 - new data in the working buffer from selected fields of a data base record in the source buffer

PROCESSING: The row of the move table control table whose number is input is accessed to get the starting row number and number of rows to be processed in the multilevel move table. These rows are then accessed sequentially to get the source buffer format row number and the working buffer format row number for each data transfer. Each buffer format tells where the data is located in its buffer (by starting character number and number of characters). The data is then transferred.

SUBROUTINE NAME: Transform from Working Buffer, TFØRMZ

PURPOSE: To transfer data from the working buffer to the target buffer.

INPUT:

- Common block SY3CØM
- working buffer
- working buffer format
- target buffer format
- Calling arguments
- target buffer row number
- print flag

OUTPUT:

- Common block SY2CØM
- target buffer
- Common block SY3CØM
- working buffer

PROCESSING: Data is transferred from the working buffer to the row of the target buffer whose number is input. The fields transferred are those in the target buffer format whose print flag value matches the input print flag value. If the field is a results field in the working buffer, then after the transfer, it is reinitialized with a value from the working buffer format. Result fields are converted to date or numeric character integers as specified by their field types in the target buffer format. As special processing for the ASATS JF and DF commands, if the target field type is a 4, 5, or 9, a conversion from internal tables is done, based on the field contents and the contents of the source record fields named UNLØAD and LSD.

SUBROUTINE NAME: Test and Jump, TJUMP

PURPOSE: To bypass a group of command lines for specified null sets.

INPUT:

- Input command string.
- Common blocks SYSCOM and SY2COM.

OUTPUT: None.

PROCESSING: If the input set contains entries, no action is taken. If the set is empty, the command file is read until a label card containing the specified label is found.

4.4 DESCRIPTION OF SUBROUTINE MODIFICATIONS

This section contains all subroutines requiring minor modifications.
Those requiring major redesign are in 4.3.

SUBROUTINE NAME: Integer to Character Conversion, CHAR

PURPOSE OF MODIFICATION: To allow handling of negative integers.

INPUT MODIFICATIONS: None.

OUTPUT MODIFICATIONS: Leftmost character of output string will be a minus sign if the input integer is negative.

PROCESSING MODIFICATIONS: After converting the absolute value of the input integer to a character string, the leftmost character is replaced by a minus sign if the input integer is negative.

SUBROUTINE NAME: System Control Routine, JLASYS

PURPOSE OF MODIFICATION: To execute new system capabilities.

INPUT MODIFICATIONS: None.

OUTPUT MODIFICATIONS: None.

PROCESSING MODIFICATIONS: Before executing the DS, DR, DK, and NK commands, subroutine DBPRO will be executed. If an indicator is not set to indicate the command was verified, the corresponding command control routines will not be executed. The LA command will not cause execution of a command control routine. The following commands will cause execution of the listed subroutines replacing the existing routine:

<u>Command</u>	<u>Subroutine</u>
CF	CFCR
SN	JNSNCR
JF	JFDFCR
DI	JFDFCR

The following new commands will cause execution of the listed subroutines:

<u>Command</u>	<u>Subroutine</u>
JN	JNSNCR
JS	SORTS
JP	JPRPCR
RP	JPRPCR
JT	TJUMP

SUBROUTINE NAME: Specify Set, SPCSET.

PURPOSE OF MODIFICATION: To provide for reading an end-of-file mark.

INPUT MODIFICATIONS: None.

OUTPUT MODIFICATIONS: None.

PROCESSING MODIFICATIONS: Change the statement that reads the record IDs to include an end-of-file branch. When an end-of-file mark is read, the program branches to the same place as when a record ID of zero is read.

4.5 DESCRIPTION OF NEW SYSTEM TABLES

This section describes the common block and tables used in conjunction with implementing arithmetic operators.

NEW COMMON BLOCK, SY3CØM

Contents:

- Array for packed input command
- Array for normal command table
- Array for function command table
- Array for BY processing table
- Array for multilevel move table
- Array for move table control table
- Array for working buffer
- Array for working buffer format
- Array for source buffer format
- Array for target buffer format
- Variables giving last used row in above arrays where applicable
- Array for intermediate storage registers

BUFFER FORMATS LAYOUT

General layout for Source Buffer Format (SBF), Working Buffer Format (WBF), and Target Buffer Format (TBF).

Column 1 - 1 word - SBF: unused

WBF: the value to be used for initialization after printing of a results field

TBF: print flag to associate printing of this field with a change of a BY field (BY processing table row number N)

Print flag = $2*N-1$ means print this field at top of BY number N

Print flag = $2*N$ means print this field at bottom of BY number N

Column 2 - 3 words - SBF: unused

WBF: four types of data: (1) alphanumeric characters representing field names, (2) \$Lbb in first word for integer literal in command line, (3) \$Tbb in first word for alphanumeric literal in command line, (4) \$Rbb in first word for calculation results

TBF: First word is row number of WBF of desired output field. Second word is key field indicator for CF command. Third word is unused.

Column 3 - 1 word - SBF, WBF, TBF: starting character for actual value in buffer being used. (In WBF, the data is actually in the command line instead of the working buffer if column 2 = \$Tbb)

Column 4 - 1 word - SBF, WBF, TBF: length of field (in characters)

Column 5 - 1 word - SBF, WBF, TBF: type of data in the field:

-1 means a binary integer contained in 4 characters

0 means an alphanumeric character string

1 means an integer in a numeric character string

2 means a date in YDDD numeric character string format

BY PROCESSING TABLE LAYOUT

Each successive row of this table defines a successively lower level subgroup of the input data and the processing associated with a change at that subgroup level.

Column 1 - 1 word - binary integer; index to Working Buffer Format (i.e., row number) pointing to the Grouping Field Name (GFN). If 0, it means the GFN was E&E. If <0, then a calculation must be performed before a test for the BY change can be made.

Column 2 - 1 word - binary integer; starting row number of normal command table when column 1 is <0.

Column 3 - 1 word - binary integer; number of rows of normal command table to be processed when column 1 is <0.

Column 4 - 1 word - binary integer; starting row number of normal command table for use when the value of this BY field or calculation changes.

Column 5 - 1 word - binary integer; number of rows of normal command table to process when the value of this BY field or calculation changes.

Column 6 - 5 words - current value of the GFN for this subgroup level. An integer or calculation result is stored in the first word, whereas a text field may be all 20 characters.

MULTILEVEL MOVE TABLE LAYOUT

Each row of this two-column table represents a move of data to be made by TFØRMW subroutine.

Column 1 - 1 word - binary integer whose value is the row number of the Source Buffer Format array where information about the field in the Source Buffer is located.

Column 2 - 1 word - binary integer whose value is the row number of the Working Buffer Format array where information about the field in the Working Buffer is located.

MOVE TABLE CONTROL TABLE LAYOUT

Each row of this two-column table defines which moves in the Multilevel Move Table are to be performed for the record in the Source Buffer from a particular data base level.

Column 1 - 1 word - binary integer whose value is the starting row number in the Multilevel Move Table.

Column 2 - 1 word - binary integer whose value is the number of rows in the Multilevel Move Table to be processed via TFØRMW subroutine to get all the needed data transferred from the Source Buffer to the Working Buffer at a particular data base level.

Column 3 - 1 word - binary integer whose value is the format number for records at this data base level.

COMMAND TABLE LAYOUT

General layout for the normal command table and the function command table. Each row of this five-column table represents an operation to be performed by the execute command subroutine, EXCMD.

Column 1 - 1 word - binary integer pointing to the first operand. A positive number is the row number of the Working Buffer Format. A negative number means an intermediate storage register, and its absolute value tells which register.

Column 2 - 1 word - positive binary integer representing the operation to be performed.

Column 3 - 1 word - binary integer pointing to the second operand for binary operations. Same type pointer as column 1.

Column 4 - 1 word - binary integer pointing to the location where the result of the operation is to be stored. Same type pointer as column 1.

Column 5 - 1 word - binary integer whose value is the row number of this command table to which a jump is made when the current operation cannot be performed due to absence of data in an operand.

APPENDIX A
NEW AND MODIFIED
RIMS COMMAND SYNTAX

NEW AND MODIFIED RIMS COMMAND SYNTAX

Listed below, in alphabetical order by command mnemonic, are the new and modified RIMS commands with the syntax to satisfy TIRF 77-0035. The first two characters of each command are the command mnemonic, and the other elements of the commands are described after all the command listings.

1. CF, Change Field

CFSN, [AE.ØP.AE,] [AE.ØP.AE,]... [AE.ØP.AE,] FN=AE [, FN=AE]
[, FN=AE]... [, FN=AE]

2. DF, Display Formatted

DFSN, FMT [, AE.ØP.AE] [, AE.ØP.AE]... [, AE.ØP.AE]
[, FN=AE] [, FN=AE]... [, FN=AE]

3. JF, Joint Display Formatted

JFSN, FMT [, AE.ØP.AE] [, AE.ØP.AE]... [, AE.ØP.AE]
[, FN=AE] [, FN=AE]... [, FN=AE]

4. JN, Joint Select Non-Key

JNSN, AE ØP.AE, [AE.ØP.AE,]... [AE.ØP.AE,]

5. JP, Joint Report

JPSN [, BY:GFN [, RE] [, RE]... [, RE]] [, BY:GFN [, RE] [, RE]... [, RE]]
[, BY:GFN [, RE] [, RE]... [, RE]] [, BY:GFN [, RE] [, RE]... [, RE]]
, BY:GFX, RE [, RE]... [, RE]!

6. JS, Joint Sort

JSSN, FN [, FN]... [, FN]

7. JT, Jump Test

JTSN, LB

8. LA, Label

LALB

9. RP, Report

RPSN[,BY:GFN[,RE][,RE]...[,RE]][,BY:GFN[,RE][,RE]...[,RE]]
[,BY:GFN[,RE][,RE]...[,RE]] [,BY:GFN[,RE][,RE]...[,RE]]
,BY:GFX,RE[,RE]...[,RE]!

10. SN, Select Non-Key

SNSN,AE.ØP.AE,[AE.ØP.AE,]...[AE.ØP.AE,]

Definition of syntactical units:

[] - items included in brackets are optional.

SN - set number (in the Status Table) upon which the operation will be performed.

FMT - format number for controlling the output format on the DF and JF commands.

FN - field name of a field that occurs in the records (or their parents) being processed.

LB - a two-character alphanumeric label.

AE - either a character string delimited by single quote mark characters or an arithmetic expression that can be computed to yield a single value. An arithmetic expression consists of alternating operands and operators, starting and ending with operands.

An operand is either a field name or a constant.

A constant is one of the following:

(1) a positive decimal integer less than 2147483648 (commas are not allowed).

(2) a date in the format #YDDD where # is a required date constant identifier, Y is the desired year

A-252

modulo 1970, and DDD is the three-digit decimal day within the year. For use in calculations, dates are converted to the number of days past 12/31/69.

An operator is either a + (addition) - (subtraction), * (multiplication), or / (integer division), with multiplication and division being computed before addition and subtraction, and then computations performed left to right as written. Parentheses may be used to affect the order of computation.

- = - an operator that means replacement of the current value of the field on the left of the = with the value of the arithmetic expression or character string on the right.
- .OP. - relational operator from the following set:
 - .EQ. equal to
 - .NE. not equal to
 - .GE. greater than or equal to
 - .LE. less than or equal to
 - .GT. greater than
 - .LT. less than
- BY: - grouping delimiter for the RP and JP commands.
- ! - end-of-command delimiter for the RP and JP commands.
- GFN - grouping field name which specifies on which fields the input set has been sorted for the RP and JP commands.
- GFX - GFN as defined above or the three special characters E&E which are a special grouping field name to specify processing on each and every record of the set.
- RE - report expression, for use in the RP and JP commands in any of the following forms:
 - (1) 'text' - a character string delimited by the single quote mark characters. This string will be displayed

at the beginning of a new value of the associated GFN (the preceding GFN in the command line) or a preceding GFN.

- (2) "text" - a character string delimited by the double quote mark characters. This string will be displayed at the end of a current value of the associated GFN (just before it or a preceding GFN changes to a new value).
- (3) FN - field name of a field that exists in the records in set SN (or in their parent records in the JP command). The contents of this field will be displayed at the beginning of a new value of the associated or a preceding GFN.
- (4) SUM(FN) - the sum of the values of field FN within the associated GFN. If field FN of any record has no value, that record is ignored in the sum. The sum will be displayed at the end of a current value of the associated or a preceding GFN.
- (5) CØUNT(FN) - the count of records within the associated GFN whose field FN contains data. The count will be displayed at the end of a current value of the associated or a preceding GFN.
- (6) MAX(FN) - the maximum value of a field FN within the associated GFN. The maximum value will be displayed at the end of a current value of the associated or a preceding GFN.
- (7) MIN(FN) - the minimum value of field FN within the associated GFN. The minimum value will be displayed at the end of a current value of the associated or a preceding GFN.

- (8) $IW=AE$ - the value of the arithmetic expression AE will be computed and stored right-justified in a field that is W characters wide ($1 \leq W \leq 99$). This value will be displayed at the beginning of a new value of the associated or a preceding GFN, and it will be computed using values from fields of the record(s) containing the new value of the associated or a preceding GFN.
- (9) $DW=AE$ - the value of the arithmetic expression AE will be computed, converted to date format, and stored left-justified in a field that is W characters wide ($4 \leq W \leq 99$). This value will be displayed at the beginning of a new value of the associated or a preceding GFN, and it will be computed using values from fields of the record(s) containing the new value of the associated or a preceding GFN.