

ESD RECORD COPY

ESD-TR-68-452

RETURN TO
SCIENTIFIC & TECHNICAL INFORMATION DIVISION
(ESTI) BUILDING 1211

ESLFE

JOVIAL EVALUATION PROJECT

William M. O'Brien

ESD ACCESSION LIST
63875

ESTI Call No. _____
Copy No. 1 of 2 cys.

15 October 1968

COMMAND SYSTEMS DIVISION
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

This document has been approved for public release and sale; its distribution is unlimited.

(Prepared under Contract No. AF 19628-68-C-0110 by Data Dynamics, Inc., 9800 S. Sepulveda Boulevard, Los Angeles, California 90045.)



ADU 6811 38

ESD TR-68-452
File Copy

LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

JOVIAL EVALUATION PROJECT

William M. O'Brien

15 October 1968

COMMAND SYSTEMS DIVISION
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

This document has been
approved for public release and
sale; its distribution is
unlimited.

(Prepared under Contract No. AF 19628-68-C-0110 by Data Dynamics, Inc.,
9800 S. Sepulveda Boulevard, Los Angeles, California 90045.)



FOREWORD

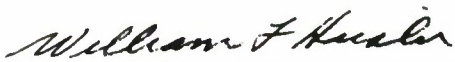
This report is the result of an extensive evaluation of Air Force Manual 100-24, Standard Computer Programming Language for Air Force Command and Control Systems. The evaluation centered on user experience with operational JOVIAL systems. It was accomplished by the use of both interview and questionnaire techniques.

The work has been performed as part of Project 6917 Task 04 under contract number F19628-68-C-0110 for Electronic Systems Division U.S. Air Force Systems Command. The project monitor was Capt. Martin J. Richter, Hq ESD (ESLFE).

The report contains detailed recommended changes to AFM 100-24 in Section 3.

This report has been reviewed and is approved.


MARTIN J. RICHTER, Capt., USAF
Project Monitor


WILLIAM F. HEISLER, Col., USAF
Chief, Command Systems Division
Directorate of Planning & Technology

ABSTRACT

The results of the evaluation of the JOVIAL Language as specified in Air Force Manual (AFM) 100-24 are contained in this report. This evaluation was based primarily on experience of users of JOVIAL Language dialects. The goal of this evaluation was to recommend deletions, retentions, modifications, and extensions to the JOVIAL language as specified in AFM 100-24 based on the users experience. The methodology of the evaluation consisted of collecting user experience data by means of a "JOVIAL Application Questionnaire" and interviews; and evaluating this data based on criteria established and documented in the "Approach for Change". This report contains a list of JOVIAL features recommended for deletion and retention and detailed specifications of recommended modifications and extensions to the JOVIAL language. In addition, the report contains the detailed interview notes and questionnaire responses which were the basic data used to arrive at the recommendations.

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ii
ABSTRACT	iii
SECTION I	INTRODUCTION
	1.1 Project Overview
	1.2 List of Features and Concepts Considered
	1.3 Method of Analysis
	1.3.1 Philosophy of Language and Language Standardization
	1.3.2 Method of Analysis of JAQ Responses
	1.3.3 Method of Analysis of Inter Responses
SECTION II	CONCLUSION
	2.1 JOVIAL Language Extention
	2.1.1 List of Extentions
	2.1.2 Remarks on Extentions
	2.2 Deletions
	2.3 List of Nucleus Features
	2.4 List of Optional Features
	2.5 Observations
SECTION III	EXTENSION SPECIFICATIONS
	3.1 General
	3.2 Recommendations
	3.2.1 Deletions
	3.2.2 Additions and Modifications
	3.2.3 Nucleus Features
	3.2.4 Optional Features
SECTION IV	ADDITIONAL RECOMMENDATIONS
	4.1 Specific Study Topics
	4.2 General Concepts
	4.3 Specific Concepts
	4.4 Language/Compiler Design Goals
APPENDIX I	Interviewee Directory and Notes
APPENDIX II	JAQ Feature Resolution and Analysis
APPENDIX III	Interview Questions
APPENDIX IV	Interview Response Analysis

SECTION I

INTRODUCTION

1.1 Project Overview

The JOVIAL Evaluation Project consisted of a study of the Standard Computer Programming Language for Air Force Command and Control System (J3 dialect), as specified in AFM 100-24. The goal of this study is to recommend deletions, retentions, modifications, and propose extensions to the features of the Standard JOVIAL language dialect as specified in AFM 100-24 on the basis of command and control programming requirements.

The study has been carried out by soliciting information from a community of users of JOVIAL. Members of this community were selected to obtain as broad a cross-section of those JOVIAL command and control programming groups as possible. There are thirteen members in the community canvassed - - they are identified in Appendix 1. It is important to note that several of the military members are non-Air Force and that the list also includes two industrial groups using JOVIAL for non-military applications. It is felt that the information obtained from such a diverse community has produced a blueprint for a "globally" useful command and control programming language.

Information was solicited from the members of the community by two means:

- (1) a JOVIAL Application Questionnaire (JAQ), ESD-TR-
and
- (2) an in-person interview conducted by DDI.

The JAQ contained comprehensive descriptions of the features defined by AFM 100-24 with questions designed to determine

- (1) deviations of the JOVIAL dialects currently in use from the standard, and
- (2) the degree of usage of those features which did not deviate significantly from their standard counterparts. (Respondents could either estimate the number of usages of the feature in their application or give an exact number.)

Information pertaining to hardware configuration, operating procedures, programmer experience levels, etc., were also solicited in the JAQ. One or more copies of the JAQ were sent to each member of the community. In most cases, each of many sub-groups of a given user group responded to a JAQ. Such responses were made with respect to the particular program subsystem for which the sub-group was responsible. In some instances, different sub-groups used different compilers (dialects);

in other instances, the same compiler (dialect) was used by many or all of the sub-groups. (Note that "members of the community" or groups and sub-groups are alternately referred to in this document as "applications".)

Interviews were conducted by DDI personnel with each of the members of the community to determine the need for additional features to the language and the need for modifications and extensions to certain existing features. Some questions at the interviews were asked in order to gain insight into broad command and control programming requirement areas. The typical interview lasted for approximately three hours. A list of the questions asked at each interview may be found in Appendix 3. Respondents to the JAQ and the interview questions were generally enthusiastic and diligent in providing the great amounts of information requested of them. We wish to acknowledge the cooperative efforts of all of those who participated in this study.

1.2 List of Features and Concepts Considered

This paragraph contains a list of those features and concepts which were considered in the study and about which information was solicited. The word "concept" as used here refers to functions performed by command and control programmers rather than explicit language facilities or features. Concepts were explored during the interviews along with certain additional features. An example of a concept is "Dynamic Storage Allocation". It was the point of the interviews to identify and develop features - - explicit language facilities - - from those concepts which the members of the community thought relevant to the solution of their problems.

Selection of features and concepts is relatively subjective. No claim is made that those chosen for investigation are exhaustive. In the interest of economy, certain Standard JOVIAL features were deliberately left out of the list because it was felt that little utility could be gained from their study.

The features and concepts have been organized into "function modules" and sub modules in order to clearly point up functional relationships between them. This organization serves to impose a perspective on the large number of features and concepts being evaluated.

In the following list of features and concepts the notation "(Q)" denotes a feature about which information was solicited in the JAQ and "(I)" denotes a feature or concept about which information was solicited at the interviews.

- (1) Function Module: Data
 - a) Item Data (Non-Structured Data)
 - Literal Items
 - Standardized Expanded Character Set (I)
(Expanded JOVIAL sign set)

- Item Type Hollerith (Q)
 - Item Type Standard Transmission Code (Q)
 - Character Data Size Attribute (Q)
 - User Definable Character Set/Encoding Scheme (I)
 - Status Items (Q)
 - Status Formulas (Q)
 - Explicit Status Size Attribute (Q)
 - Item Type Bit String (I)
- b) Data Structures (Aggregates of Items)
- Structure Array (Q)
 - Structure Table (Q)
 - Table Variability Attribute - Rigid (Q)
 - Table Variability Attribute - Variable (Q)
 - NENT (Q)
 - NWDSN (Q)
 - Structure String
 - Structure GROUP (I)
 - Structure SET (I)
- c) Dynamic Allocation of Data (I)
- (2) Function Module: Referencing Data
- BIT (Q)
 - BYTE (Q)
 - CHAR/MANT (Q)
 - Subscripting
 - Subscripting Expressed as Complex Numeric Formula (Q)
 - Nested Subscripting (Q)
 - Table Entry Referencing (ENTRY/ENT) (Q)
 - Table Entry Referencing Notation - ENT (Q)
- (3) Function Module: Representing Data in the Standard JOVIAL J3 Memory
- Computer Representation of Data (Q)
 - Importance of Storage Allocation Facilities (I)
 - Importance of Machine Independence (I)
 - Memory "Model" (Q)
 - Signed Magnitude Representation of Numbers (I)
 - Octal Constant (Q)
 - Hexadecimal Constants (I)
 - Basic Table Structure Attribute - Parallel (Q)
 - Basic Table Structure Attribute - Serial (Q)
 - Generalized Packing (I)
 - Ordinary (Table) Packing - Medium and/or Dense (Q)
 - Ordinary Packing - Medium (Q)
 - Ordinary Packing - Dense (Q)
 - "No Packing" Notation - "N" (Q)

- Defined (Table) Packing (Q)
- Independent Overlay Declaration (Q)
- Subordinate Overlay Declaration (Q)
- 'LOC (Q, I)
- Absolute addresses (OVERLAY, PROGRAM) (I)

(4) Function Module: Numeric Items and Arithmetic Operations

- Item Type Integer (Q)
- Item Type Fixed Point (Q)
- Item Type Fixed Point
Scale in Declaration Blank or Zero (Q)
- Range Attribute (Q)
- Item Type Floating Point (Q)
- Sign Attribute (Signed and Unsigned) (Q)
- Double Precision
(also called Extended Precision) (I)
- Parenthesized Numeric Formulas (Q)
- Mixed Item Types in Numeric Formulas (Q)
- Prefix + and - (Q)
- Precedence of Prefix + and - (I)
- Exponentiation Operator (Q)
- Exponentiation Notation - (**) (Q)
- Absolute Value Operator (Q)
- Absolute Value Notation - (/ /) (Q)
- REM (Q)
- REMQUO (Q)

(5) Function Module: Dual Items and Dual Arithmetic Operations

- Item Type Dual (Q)
- Dual Arithmetic Operations (Q)
- Structure Operators - Arrays, Tables (I)
- Structure Operator - Matrix (I)
- Structure Operator - Complex (I)

(6) Function Module: Testing Operations

- Hollerith Comparisons (I)
- Relational Formulas (excluding chained) (Q)
- "Chained" Relational Formulas (Q)
- ODD (I)

(7) Function Module: Boolean Items and Logical Operators AND, OR, and NOT

- Boolean Formulas-with AND, OR, NOT (Q)
- Boolean Items (Q)

(8) Function Module: Assignment

- Numeric Item Type Conversion in Assignment Statements (Q)
- Rounded Numeric Assignment (I)

- Round Attribute (Q)
- Boolean Assignment Statement (Q)
- Item Preset (Q)
- Array Preset (Q)
- String Preset (Q)
- Exchange Statement (Q)

(9) Function Module: Program Structure and Execution of Program Structure

- a) Parallel Processing (I)
- b) Compound Statement (Q)
- c) GOTO Statement (Q)
- d) Stopping and Pausing
 - STOP (Q)
 - STOP (Modification) (I)
 - STOP Statement'label (Pausing) (Q)
 - Pausing (I)
- e) Conditional Transfer of Control
 - IF Statement (Q)
 - IFEITH/ORIF (Q)
 - Index Switch (Q)
 - Item Switch (Q)
 - Label Items (I)
 - Switch Names within Switch Declarations (Q)
 - Close Names within Switch Declarations (Q)
 - Parallel "Monitoring" (I)
- f) FOR Statements (Iteration)
 - FOR Loop - One Factor (Q)
 - FOR Loop - Two Factor (Q)
 - FOR Loop - Three Factor (Q)
 - FOR Loop - Decrementing (Q)
 - Test Statement (Q)
- g) Closed Subprograms and their Execution
 - Program (Q)
 - Procedure (Excluding Function) (Q)
 - Alternate Procedure Entrances (Q)
 - Close in Parameter List (Q)
 - Statement Label in Parameter List (Q)
 - Function (Q)
 - Close (Q)
 - Closed Subprogram (Delete Close in Favor of Proc?) (I)
 - Return Statement (Q)
 - 'PROGRAM Declaration (Q)
 - Recursive Subprograms (I)
- h) Direct Code (Q)

- (10) Function Module: Input/Output
- Structure File
 - Structure File - - Hollerith or Binary (Q)
 - Structure File - - Record Length Variability (Q)
 - Input/Output - POS (Q)
 - Input/Output - OPEN Statement with Operand (Q)
 - Input/Output - OPEN Statement without Operand (Q)
 - Input/Output - SHUT Statement with Operand (Q)
 - Input/Output - SHUT Statement without Operand (Q)
 - Input/Output - Input Statement (Q)
 - Input/Output - Output Statement (Q)
 - Stream Oriented I/O (I)
 - Functional Files (I)
 - Device Oriented I/O (I)
- (11) Function Module: Naming Data and Program Structures
- Proc-like Name Scopes (I)
 - Multiple Statement Labels (Q)
- (12) Definition Facilities
- Define Directive (Q)
 - Extended Define Directive (I)
 - Mode Directive (Q)
 - Extended Mode Directive (I)
 - Like Table Declaration (Q)
 - Macro Facilities (I)

1.3 Method of Analysis

This subsection describes the philosophy of language and language standardization which is central to the methods of this analysis. In addition, the method of analysis of the JOVIAL Application Questionnaire and Interview Responses is described.

1.3.1 Philosophy of Languages and Language Standardization

Very briefly, a language is a notational system with a community of users. The structure of the language "sentences" and the meanings associated with them reflect both

- a set of problems shared by members of the community, and
- a set of tools which make the problems tractable.

To standardize a language means to determine the problems faced by all, or a substantial subset, of the members of the community and to obtain a consensus as to the tools to be employed in solving the common problems. Clearly, a language standard should not support tools designed for coping with problems peculiar to a small subcommunity nor should the standard support a variety of different sets of

tools designed to handle the same set of common problems. The importance of standardizing a language lies in the fact that more efficient communication of problems and problem solution will, in general, ensue. Standardization creates a common frame of reference for the members of the community of users.

The language which is the object of this study is, of course, the JOVIAL J3 dialect of JOVIAL specified in AFM 100-24. The community of users consists of all those applications personnel who are employing JOVIAL in computerized command and control systems. The problem solving tools are the JOVIAL language features themselves and, equally important, sets of features which in this document are referred to as function modules. It is felt that evaluation of individual features in an isolated fashion is insufficient because many related features, or tools, are generally brought to bear in concert to solve command and control problems. Thus, the significance of fixed point items, say, can be meaningfully understood only in the context of the other numeric items (integer and floating point), since the numeric items are functionally connected.

Similarly, the significance of numeric items can be better understood when examined in the context of arithmetic operations. "Numeric Items and Arithmetic Operations" constitute a subsystem of individual JOVIAL features which address the general problem of representing numeric entities and doing arithmetic with them. This study then relies heavily on the concept of problem solving subsystems, or function modules, to clarify command and control programming processes and to aid in the evaluation of individual features.

The analysis techniques employed in the evaluation of the JOVIAL language resulted in the partitioning of the JOVIAL features into three subsets:

- A Nucleus subset which includes features used to deal with the problems faced by a substantial majority of the members of the community,
- An Optional subset which includes the features used to deal with the problems faced by a significant number of members of the community, and a
- Deleted subset consisting of those features which are used infrequently or not at all by all but a very few members of the community.

It is our intention that all implementations of Standard JOVIAL J3 contain the features of the Nucleus subset; that implementors select all or none of the features in the Optional subset -- depending on their requirements; that no implementation of Standard JOVIAL J3 contain any of the Deleted subset features.

The technique used to determine if a feature is to be in the Nucleus, Optional, or Deleted subsets is called the "resolution". Concepts may be thought of as placed in Nucleus or Optional concept subsets. Features -- explicit language facilities -- are then placed accordingly in the Nucleus or Optional feature subset. The word "resolution" will be used to describe the process of disposition of concepts as well as features. With respect to both JAQ and interview response analysis, retention as nucleus will be recommended if two thirds or more of the (effective) respondents show a high degree of appreciation for the feature/concept; retention as optional will be recommended if fewer than two thirds but not fewer than one third show a high degree of appreciation. "High degree of appreciation" is defined separately for JAQ and Interview Analysis.

1.3.2 Method of Analysis of JAQ Responses

The Standard JOVIAL J3 features about which information was collected in the JAQ's has been divided into two classes:

- Class 1 the features in this class have been "Accepted as nucleus" features a priori; they are deemed so important that retention without question is recommended and they are automatically placed in the Nucleus feature subset.
- Class 2 the features in this class are all those features not in Class 1. Their placement in the Nucleus, Optional, or Deleted subsets must be determined on the basis of the JAQ responses.

In almost every case, the function performed by a Class 2 feature can be performed by one or more Class 1 features. The resolution of each Class 2 feature is performed using a cost-effectiveness technique based upon a "usage rate", or resolution equation, and comparison of the usage rate with a "usage rate threshold". It should be noted that a distinct usage rate and usage rate threshold have been defined for all of the Class 2 features considered in the study. The usage rate is either the number of times the feature was reported used in the JAQ or it is a function of this number. In most cases the function is a simple ratio with the number of uses appearing as the numerator. Ratios of this kind serve to measure the fraction of times a feature was used, where it could have been used to satisfy a certain application requirement. For example, in those cases where some "recurrent" data structure is required to fulfill an application requirement, the utility of using tables, given that both tables and arrays will satisfy this requirement, may be measured by the usage rate,

$$U = \frac{N(\text{TABLES})}{N(\text{TABLES}) + N(\text{ARRAYS})} = \frac{Q78}{Q78 + Q74}$$

where

$N(\text{TABLES})$ represents the number of occurrences of a TABLE declaration in an application as contained in JAQ question Q78, and $N(\text{ARRAYS})$ represents the number of occurrences of an ARRAY declaration in an application as contained in JAQ question Q74.

If the usage rate for tables, when evaluated with JAQ response data for a specific application, is greater than one half, say, we infer that the application programmers find their "recurrent" data structure problems more readily tractable in terms of tables than arrays in more than one half of the instances in which a recurrent data structure is required. Two facts regarding usage rate evaluation must be noted.

- (1) In cases where both the numerator and denominator are 0, the usage is taken as 0 by convention
- (2) Because the JAQ permits usage responses which are estimates rather than exact counts, such ratios as the one in the example above may yield values greater than one on evaluation. In such cases the evaluated usage rate may be thought of as a "preference indicator".

The usage rate threshold is a number against which usage rates, evaluated for a given application, are compared. With respect to a given feature, the usage rate threshold is constant for each application. If the usage rate for a given application is greater than or equal to the threshold specified for the feature, we infer that the application programmers obtain a reasonable degree of utility (i.e., a high degree of appreciation) from the feature; otherwise, we infer that the utility gained does not offset the "cost" of the feature. It must be noted that it is impossible to arrive at precisely accurate thresholds since no accurate means exist to measure costs and utilities. Indeed, DDI has specified thresholds subjectively, based upon our own experience with command and control applications and knowledge of the compilation process. Where we have specified a high threshold, we feel that the cost outweighs the utility; a low threshold indicates that the utility is worth the cost. Usage rate thresholds, then, have been arrived at subjectively by weighing the cost of each Class 2 feature against its effectiveness or utility. Examples of characteristics which drive up the cost of a feature are:

- (1) The feature is redundant -- its function can be performed by other (Class 1) features. The redundancy requires a more complex compiler and more learning (with possibly more errors made) by the programmer.
- (2) The feature is context sensitive, introducing certain levels of complexity into the compilation process (e.g., status constant).

A Class 2 feature is, in general, effective in that it provides a programmer convenience. Thus the exchange statement

$$AA = BB \quad \$$$

is more convenient to the programmer than

$$\begin{aligned} TEMP &= BB \quad \$ \\ BB &= AA \quad \$ \\ AA &= TEMP \quad \$ \end{aligned}$$

though an expense is incurred in the implementation of the exchange feature.

Because of the subjectivity involved in specifying thresholds, we have deliberately leaned toward low values for thresholds. The low values of thresholds reflects an underlying philosophy maintained throughout this study, namely that the error incurred by accidentally deleting or making optional a useful feature was far more serious than the error of retaining a non-useful features.

The usage rates and thresholds for each feature are given with the response summaries for the feature in Appendix 2. There, usage rate threshold is denoted by T1.

1.3.3 Method of Analysis for Interview Responses

A concept or proposed new feature was retained as nucleus if more than two thirds of the members of the community responding responded positively. The concept or proposed new feature was retained as optional if there were from one third to two thirds positive responses and it was deleted if less than one third responded positively. The Interview Response Analysis may be found in Appendix 4.

SECTION II
CONCLUSION

This section contains recommendations, based on both the JAQ and Interview Response Analysis, for alteration of the Standard JOVIAL (J3) language dialect. The recommendations include a list of JOVIAL language extensions along with a list of deletions of features currently specified in AFM 100-24. Additionally, all current and extended features are classified as either Nucleus or Optional. Finally, this section contains a discussion, in general terms, of our observations on command and control programming language requirements.

2.1 JOVIAL Language Extensions

2.1.1 List of Extensions

DDI recommends that the following extensions, described fully in the indicated subsections of Section 3, be incorporated into the Standard JOVIAL J3 language as Nucleus or Optional features:

- (1) Hexadecimal Constants (3.2.2.1)
(Optional - - User may implement hexadecimal, or octal, or both, but either hexadecimal or octal must be implemented.)
- (2) Simplified Hollerith Constant Form (3.2.2.2)
(Optional - - Removes the requirement to count the number of hollerith characters.)
- (3) User Definable Character Encoding (3.2.2.3)
('CHARCODE Directive)
(Optional - - Provides facility for defining a character set and encoding scheme for each character in it.)
- (4) Operations on Literal Data (3.2.2.4)
(Nucleus - - Establishes facility for Hollerith and other Literal Comparisons as well as conversion between Literal item types on assignment.)
- (5) Computer Representation of Numeric Constants and Variables (3.2.2.5)
(Nucleus - - loosens the machine dependent restrictions on representation of numeric data.)
- (6) Precedence of Unary Operators (3.2.2.6)
(Nucleus - - Changes the precedence of prefix + and - to next highest after exponentiation.)

- (7) Extended Define Directive (3.2.2.7)
(Optional - - Allows "arguments" to be included with a DEFINE identifier.)
- (8) Extended Mode Directive (3.2.2.8)
(Optional - - Allows mode directive to select item description on basis of first letter of name.)
- (9) Alternate Entrances to Procedures and Functions (3.2.2.9)
(Nucleus)
- (10) Extended Precision Numeric Items and Constants (3.2.2.10)
(Optional, with hardware - - Permits declaration of double precision floating point numbers.)
- (11) Device Oriented Input/Output Module (3.2.2.11.1)
(Optional - - Provides framework for commanding and responding to I/O devices.)
- (12) Functional File Input/Output Module (3.2.2.11.2)
(Optional - - Provides framework for functional ("logical") file processing.)
- (13) Data Editing and Conversion (Formatting) (3.2.2.11.3)
(Nucleus - - provides FORTRAN-like formatting capability.)
- (14) Stop Statement (3.2.2.12)
(Nucleus - - Alters the meaning of STOP to suspend processing wherever executed.)
- (15) Pause Statement (3.2.2.13)
(Optional - - Provides pausing facility.)
- (16) Rounded Assignment Statement (3.2.2.14)
(Optional - - Provides the facility for declaring rounding in an assignment statement, not in the item declaration.)
- (17) Extension to Program (3.2.2.15)
(Optional - - Allows the ability to declare a procedure as a "stand-alone" subroutine.)

2.1.2 Remarks on Extensions

2.1.2.1 Input/Output

The proposed (optional) Device Oriented I/O module and Functional File module are intended to supercede the File Structure and Operators currently specified in AFM 100-24. The two modules proposed, force a separation of hardware considerations from function or logical file elements with the intent of providing much greater flexibility and power than that offered by the current file system of the standard. It must be pointed out, however, that the Functional File Module is very loose and therefore highly "participational" in nature. The applications programmer must write his own set of support programs to do all the real file manipulation work; however,

it is anticipated that user installations fix on a standard set of support programs which would be used by all applications programmers. The Functional File Module is loose for the reason that we are unable at this time to fully identify functional file properties. This was due to the fact that the Input/Output information obtained in the study was sparse, and the latter fact may be true because I/O at this level tends to be complex. We recommend that this area be studied further, in the light of command and control applications.

The formatting module has been made independent of the other I/O modules, since it is nucleus, while the others are optional.

2.1.2.2 Overriden Extensions

Certain features and concepts were well received at interview and should, by all rights be proposed as either nucleus or optional extensions now. These features and concepts are:

- Item Type Bit String
- (Data) Structure GROUP
- Structure SET
- Dynamic Storage Allocation Facilities
- Generalized Packing
- Label Item
- Recursive Subprograms

Our action -- to make no recommendation at this time -- is based on arguments discussed under observations (sub-section 2.5, below) and also on the fact that the addition of these new structures would result in some complex ramifications which are unresolved at this time involving currently available structures.

2.2 Deletions

The criteria utilized to determine which JOVIAL features should be deleted is given in Section 1 (1.3). Based on these criteria and the data summarized in the JOVIAL Feature Resolution Analysis contained in Appendix 2 as qualified by the Interview Response Analysis contained in Appendix 4, Data Dynamics recommends that the following features be deleted from the JOVIAL Specifications AFM 100-24.

The number in parenthesis in the list below is the Approach for Change (AFC) reference number which references the Resolution Analysis of each specific feature.

- | | <u>AFC#</u> |
|---|-------------|
| 1) Item Type Standard Transmission Code | (A1.1.1.2) |
| 2) Standard Transmission Code Formulas | (A1.1.1.2) |
| 3) Structure String | (A1.2.5) |
| 4) CHAR/MANT | (A2.3) |
| 5) Table Entry Referencing Notation - ENT
(ENTRY only may be used) | (A2.6) |

	<u>AFC#</u>
6) Ordinary Packing - Medium (Only Dense Packing is allowed)	(A3.4)
7) Ordinary Packing - No packing specified	(A3.4)
8) Subordinate OVERLAY Declaration	(A3.6)
9) Round Numeric Assignment Range Attribute	(A4.3)
10) Exponentiation Notation - (* *) (Only ** may be used)	(A4.6)
11) Absolute Value Notation (/ /) (Only ABS may be used)	(A4.8)
12) REM	(A4.9)
13) REMQUO	(A4.9)
14) Item Type Dual	(A5)
15) Dual Formulas	(A5)
16) String Preset	(A8.1)
17) Round Attribute	(A8.2)
18) Stop Statement Label	(A9.3)
19) Switch Names within Switch Declarations	(A9.4.3)
20) Close Names within Switch declarations	(A9.4.3)
21) Structure File	(A10)
22) Structure File - Hollerith or Binary	(A10)
23) Structure File - Record Length Variability	(A10)
24) Open Statement with Operand	(A10)
25) Open Statement without Operand	(A10)
26) Shut Statement without Operand	(A10)
27) Shut Statement with Operand	(A10)
28) Input Statement	(A10)
29) Output Statement	(A10)

2.3 List of Nucleus Features

Based on the criteria and data as indicated in sub-Section 2.2, the features listed below are recommended for inclusion in the Nucleus. The numbers in parenthesis reference the AFC number in the JOVIAL feature resolution analysis in Appendix 2 and references to sub-Sections of Section 3.

- 1) Item Type Hollerith (A1.1.1.1)
- 2) Hollerith Formula (A1.1.1.1)
- 3) Character Data Size Attribute (A1.1.1.3)
- 4) Operations on Literal Data (3.2.2.4)
- 5) Explicit Size Attribute (A1.1.3)
- 6) Structure Table (A1.2.2)
- 7) Table Variability Attribute - Rigid (A1.2.3)
- 8) Table Variability Attribute - Variable (A1.2.3)

- 9) NENT (A1.2.3)
- 10) BYTE (A2.2)
- 11) Subscripting - Expressed as Complex Numeric Formula (A2.4)
- 12) Subscripting - Nested (A2.4)
- 13) Table Entry Referencing (A2.5)
(ENTRY only may be used)
- 14) Computer Representation of Data (A3)
- 15) Computer Representation of Numeric Items and Constants (3.2.2.5)
- 16) Octal Constants (A3.1, 3.2.2.1)
Hexadecimal Constant (Must implement one or the other,
may implement both)
- 17) Basic Table Structure Attribute - Parallel (A3.3)
- 18) Basic Table Structure Attribute - Serial (A3.3)
- 19) Defined (Table) Packing (A3.5)
- 20) Independant Overlay Declaration (A3.6)
- 21) 'LOC (A3.8)
- 22) Parenthesized Numeric Formulas (A4)
- 23) Mixed Item Types in Numeric Formulas (A4)
- 24) Item Type Integer (A4)
- 25) Item Type Fixed Point (A4)
- 26) Item Type Fixed Point - Scale in Declaration Blank or Zero (A4)
- 27) Item Type Floating Point (A4)
- 28) Sign Attribute (A4.4)
- 29) Prefix + and - (A4.5)
- 30) Precedence of Unary Operators (3.2.2.6)
(Alteration of precedence of +, -)
- 31) Exponentiation Operator (A4.5)
(Notation ** only)
- 32) Relational Formulas (excluding chained) (A6.1)
- 33) Boolean Formulas - with AND, OR, NOT (A7.1)
- 34) Item Type Boolean (A7.2)
- 35) Numeric Item Type Conversion in the Assignment Statement (A8)
- 36) Boolean Assignment Statement (A8)
- 37) Item Preset (A8.1)
- 38) Table Preset (A8.1)
- 39) Compc und Statement
- 40) GOTO Statement (A9.2)
- 41) STOP Statement (A9.3, 3.2.2.12)
- 42) IF Statement (A9.4.1)
- 43) FOR-loop-One Factor (A9.5)
- 44) FOR-loop-Two Factor (A9.5)
- 45) FOR-loop-Three Factor (A9.5)
- 46) FOR-loop-Decrementing (A9.5)
- 47) Test Statement (A9.5.4)

- 48) Program (A9.6, A9.7)
- 49) Procedure (Excluding Function) (A9.6.1)
- 50) Function (A9.6.1)
- 51) Alternate Entrances to Procedures and Functions (3.2.2.9)
- 52) CLOSE (A9.6.2)
- 53) Return Statement (A9.6.3)
- 54) Direct Code (A9.8)
- 55) Data Editing and Conversion (3.2.2.11.3)
- 56) Multiple Statement Labels (A11.2)
- 57) ALL (A12.5)
- 58) COMPOOL (A12.6)

2.4 List of Optional Features

Based on the criteria and data as indicated in sub-section 2.2, the following features are recommended to be included as optional features:

- 1) Hexidecimal Constants (3.2.2.1)
- 2) Simplified Hollerith Constant Form (3.2.2.2)
- 3) User Definable Character Encoding Scheme ('CHARCODE) (3.2.2.3)
- 4) Item Type Status (A1.1.2)
- 5) Status Formulas (A1.1.2)
- 6) Explicit Status Size Attribute (A1.1.3)
- 7) Structure Array (A1.2.1)
- 8) NWDSN (A1.2.4)
- 9) BIT (A2.1)
- 10) Ordinary Packing (Dense only is allowed) (A3.4)
- 11) Absolute Value Operator (ABS only is allowed) (A4.7)
- 12) Extended Precision Numeric Items and Constants (3.2.2.10)
- 13) Chained Relational Formulas (A6.2)
- 14) Array Preset (A8.1)
- 15) Exchange Statement (A8.3)
- 16) IFEITH/ORIF Statements (A9.4.2)
- 17) Index Switch (A9.4.3)
- 18) Item Switch (A9.4.4)
- 19) CLOSE in Parameter List (of Procedure) (A9.6.1)
- 20) Statement Label in Parameter List (of Procedure) (A9.6.1)
- 21) 'PROGRAM Declaration (A9.7)
- 22) Item Declaration (A11.1)
- 23) Define Directive (A12.1)
- 24) Extended Define Directive (3.2.2.7)
- 25) MODE Directive (A12.2)
- 26) Extended MODE Directive (A12.2, 3.2.2.8)
- 27) Like Table Declaration (A12.4)

} (Must be implemented together)

- 28) Device oriented Input/Output Module (3.2.2.11.1)
- 29) Functional File Input/Output Module (3.2.2.11.2)
- 30) Pause Statement (3.2.2.13)
- 31) Extension to Program ("Stand-alone" procedure) (3.2.2.15)

2.5 Observations

2.5.1 Array

The only feature accepted as nucleus originally in the Approach for Change which did not find reasonable user support was the Array. We have therefore made Array an optional feature. Users found Table far more useful.

2.5.2 Data Structures

Interview responses regarding proposed new data structures (GROUP, SET, etc.) together with the heavy use of Tables inclined us toward the belief that command and control programmers require the ability to manipulate highly complex data structures. We feel that it is unwise to simply pile on new structures at this point. Rather, we believe, the command and control language should contain structure building primitive operators which would allow for the definition and creation, either at compile time or execution time, of a wide variety of complex structures, these structures taking a form specified by the applications programmer. We are not currently able to identify these primitives but feel that the task of such identification should be pursued.

2.5.3 Resource Allocation

Because of the enthusiasm of interviewees for packing capabilities, and the proposed generalized packing, it has become clear that the command and control programmer requires the ability to manage the memory space resources available to him. We note the remark of ADPAC, requesting a special "fast-executing" subset of JOVIAL for real time application, and conclude that management of execution time resources is probably important as well. We feel that a command and control language should probably have even more resource management facilities than JOVIAL and that these should be connected with ease to the data structure building primitives described in sub-section 2.5.2.

SECTION III

EXTENSION SPECIFICATIONS

3.1 General

This section contains a complete description of the deletions, modifications, and additions to the JOVIAL J3 language as specified in AFM 100-24 resulting from the analysis phase of this project. In addition, the section contains a complete list of all recommended nucleus and optional features along with a cross reference of each feature to the appropriate section in the "JOVIAL Application Questionnaire" and the "Approach For Change" documents.

During this project many JOVIAL language requirements evolved which DDI was unable within the scope of the project to investigate in detail. An example of such a requirement includes language features which would allow constructing new types of data structures and allocating these structures and computer programs in terms of a specified optimization of time and space. Other examples as included in Section 4 below, such as including real time concepts and list processing capabilities, were also not within the scope of the project. However, considering these limitations, DDI is confident that JOVIAL Extension Specifications, described in this section, is a significant improvement over the current JOVIAL J3 (AFM 100-24) Specifications.

3.2 Recommendations

The following are DDI's recommendations for modifications to JOVIAL Specifications as contained in AFM 100-24.

3.2.1 Deletions

The following features are recommended for deletion from the JOVIAL Language. The criteria used for deletion of these features is described in Section 1 and the specific resolution analysis of each feature is contained in Appendix 2. The references for each feature is to the sub-section relating to the feature in the JOVIAL Application Questionnaire (JAQ) and to the Approach For Change (AFC) documents. As the JOVIAL Feature Resolution Analysis in Appendix 2 is organized by AFC numbers, the referenced AFC can be utilized to find the specific resolution or justification for deletion of each of the following features:

	<u>Features</u>	<u>JAQ</u>	<u>AFC</u>
1)	Item Type Standard Transmission Code	J3.5.1.11.2	A1.1.1.2
2)	Standard Transmission Code Formulas	J3.5.2.5	A1.1.1.2

	<u>Features</u>	<u>JAC#</u>	<u>AFC#</u>
3)	Structure String	J3.5.1.4	A1.2.5
4)	CHAR/MANT	J3.5.2.2.2.3-4	A2.3
5)	Table Entry Referencing Notation-ENT (ENTRY only may be used)	J3.5.2.2.1	A2.6
6)	Ordinary Packing - Medium (Only Dense Packing is allowed)	J3.5.1.3.6.1	A3.4
7)	Ordinary Packing - No packing specified	J3.5.1.3.6.1	A3.4
8)	Subordinate OVERLAY Declaration	J3.5.1.7.2	A3.6
9)	Range Attribute	J3.5.1.1.8	A4.3
10)	Exponentiation Notation - (* *) (Only * * may be used)	J3.5.2.3.3	A4.6
11)	Absolute Value Notation (/ /) (Only ABS may be used)	J3.5.2.3.4	A4.8
12)	REM	J3.5.4.9	A4.9
13)	REMQUIO	J3.5.4.7	A4.9
14)	Item Type Dual	J3.5.1.1.7	A5
15)	Dual Formulas	J3.5.2.4	A5
16)	String Preset	J3.5.1.6.4	A8.1
17)	Round Numeric Assignment-Round Attribute	J3.5.1.1.10	A8.2
18)	Stop Statement Label	J3.5.5.7	A9.3
19)	Switch Names within Switch Declarations	J3.5.5.2	A9.4.3
20)	Close Names within Switch Declarations	J3.5.5.2	A9.4.3
21)	Structure File	J3.5.1.5	A10
22)	Structure File - Hollerith or Binary	J3.5.1.5	A10
23)	Structure File - Record Length Variability	J3.5.1.5	A10
24)	Open Statement with Operand	J3.5.3.3	A10
25)	Open Statement without Operand	J3.5.3.3	A10
26)	Shut Statement without Operand	J3.5.3.3	A10
27)	Shut Statement with Operand	J3.5.3.3.	A10
28)	Input Statement	J3.5.3.3	A10
29)	Output Statement	J3.5.3.3	A10

3.2.2 Additions and Modifications

3.2.2.1 Hexadecimal Constants

For some computers, the specification of address or "bit pattern" information is more conveniently expressed by use of the hexadecimal (base 16) number system than by the octal system. Depending on their needs, implementors may incorporate hexadecimal constants, octal constants, or both.

A hexadecimal:constant is composed of hexadecimal:numerals. A hexadecimal:numeral is one of the following sixteen JOVIAL numerals or letters:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

The correspondence between hexadecimal numerals and bit patterns is:

<u>Hexadecimal Numeral</u>	<u>Bit Pattern</u>
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

A hexadecimal:constant is formed by the letter X, followed by a left parentheses, followed by a string of hexadecimal:numerals, followed by a right parentheses:

X(string:of:hexadecimal:numerals)

Examples of hexadecimal:constants are:

X(777)
X(F)
X(01A9)
X(FAD)

A hexadecimal:constant is represented as a string of bits, equal in length to four times the number of hexadecimal:numerals making up the string:of:hexadecimal:numerals in the constant form; four bit groups making up the bit string correspond to and have the same ordering as the hexadecimal:numerals making up the hexadecimal:constant as shown in the table above.

Hexadecimal:constants may be used in the same places as octal:constants and have corresponding meanings. More specifically: Hexadecimal:constants may be used in machine address specifications (as in OVERLAY), preset specifications in numeric: or literal:item, array:, or table:item:declarations, and in literal:assignment:statements.

3.2.2.2 Simplified:Hollerith:Constant Form

A hollerith:constant is either a regular:hollerith:constant or a simplified:hollerith:constant. A regular:hollerith:constant has the form number H(string:of:hollerith:characters) (see CED 2418, AFM 100-24 under hollerith:constant). A simplified:hollerith:constant has the form

H(string:of:hollerith:characters)

with the restrictions on the string:of:hollerith:characters that, in scanning from left to right through string, the number of right:parentheses encountered at any given position must never exceed the number of left:parentheses encountered, and the total number of left-parentheses included in the string must equal the total number of right:parentheses.

Examples of simplified:hollerith:constants are

H(NO MORE COUNTING WITH MOST HOLLERITH CONSTANTS)

and

H(NO MORE COUNTING WITH (MOST) HOLLERITH CONSTANTS)

3.2.2.3 User Definable Character Encoding ('CHARCODE)

This facility allows the programmer to define a variety of character sets and encoding schemes, and is a compiler directive in the sense of the JOVIAL MODE and DEFINE features. A 'CHARCODE directive shows a correspondence between each member of a user selected set of characters and a user determined "bit pattern" memory representation for that character.

The set of all characters from which the programmer may choose in building a character set/encoding scheme contains at least the JOVIAL signs, and possibly some implementation dependent characters as well; additionally this "set of all characters" has an implementation dependent "bit pattern" memory representation or encoding scheme. In JOVIAL terminology, this character set/encoding scheme is referred to by the adjective hollerith.

It is anticipated that the two major uses for 'CHARCODE will be

- to provide a means for interfacing with many kinds of I/O devices (each of which may have different bit pattern representations for a character set), and
- to permit the specification of user defined collating sequences.

A charcode:directive is:

```
'CHARCODE θ* charcode:letter θ fill:character θ  
BEGIN θ char:code:correspondence:list θ END θ $
```

Charcode:letter means any of the letters excluding I, A, F, H, B, S, O, X, or V;
char:code:correspondence:list means either: Char:code:correspondence or char:code:correspondence:list θ; char:code:correspondence means:

Hollerith:character θ = θ hex:or:octal:constant

hex:or:octal:constant means an octal:constant, if the implementation supports octal:constants, hexadecimal:constant if the implementation supports hexa:decimal:constants, or either if the implementation supports both octal and hexadecimal. Fill:character and hollerith:character are single hollerith characters expressed as hollerith:constants one character in length. The meaning of fill:character will be explained in connection with literal:assignment:statements and literal comparisons in sub-section 3.2.2.4. It is required that fill:character appear in the char:code:correspondence:list.

The charcode:directive serves to define a new item type with a name "abbreviation" specified by charcode:letter, in the same sense that "H" is the (built-in) literal:item type hollerith. The adjective literal will henceforth be applied to items, constants, arrays, etc., for which either the adjective hollerith applies, or where the data involved are defined with respect to a charcode:directive.

While a charcode:directive is active, it is possible both to declare items and to express constants for the literal:item type defined by that charcode:directive. Item:declarations and constants for literal:item types defined by charcode:directives are formed precisely as hollerith:item:declarations and constants, except that "H" in such declarations and constants is replaced by charcode:letter; also the string of characters appearing in such constants must be made up only of those characters which appear in the directive.

*θ means that a space is required or permitted.

Regarding the computer memory representation of charcode defined character sets, these character set are represented in byte strings in the same manner as Hollerith characters, each byte being ω bits in length (see AFM 100-24, Figure 24-1 and CED 2419.6). Further, the encoding of characters within bytes is, of course, as specified in the charcode:directive.

Regarding the char:code:correspondence:list:

- A given hollerith character may appear once and only once in any charcode:directive.
- A given hexadecimal or octal encoding may appear once and only once in any charcode:directive.
- If a hex:or:octal:constant has a size (number of bits in length) greater than the implementation specific ω bits per byte, only the rightmost bits of the byte are used to define the encoding; if the size is less than ω , as many bits with value 0 are added to the left to make a bit pattern ω in length.

Regarding the scope of a charcode:directive: The effect begins at that point in the source code where the directive appears and terminates at the TERM statement or at the next charcode:directive with the same charcode:letter, whichever comes first.

An example of a charcode:directive will be given in which Standard Transmissan Code is defined in terms of hollerith:

```
'CHARCODE      T      1H(      )
  BEGIN        1H( ) = O(0), 1H(A) = O(6),
                1H(B) = O(7), 1H(C) = O(10),
                1H(D) = O(11), 1H(E) = O(12),
                1H(F) = O(13), 1H(G) = O(14),
                1H(H) = O(15), 1H(I) = O(16),
                1H(J) = O(17), 1H(K) = O(20),
                1H(L) = O(21), 1H(M) = O(22),
                1H(N) = O(23), 1H(O) = O(24),
                1H(P) = O(25), 1H(Q) = O(26),
                1H(R) = O(27), 1H(S) = O(30),
                1H(T) = O(31), 1H(U) = O(32),
                1H(V) = O(33), 1H(W) = O(34),
                1H(X) = O(35), 1H(Y) = O(36),
                1H(Z) = O(37), 1H() = O(40),
                1H(-) = O(41),
                1H(+) = O(42), 1H(=) = O(44),
                1H($)= O(47), 1H(*) = O(50),
                1H(!) = O(51), 1H(,) = O(56),
```

```

1H(0) = O(60), 1H(1) = O(61),
1H(2) = O(62), 1H(3) = O(63),
1H(4) = O(64), 1H(5) = O(65),
1H(6) = O(66), 1H(7) = O(67),
1H(8) = O(70), 1H(9) = O(71),
1H(') = O(72), 1H(/) = O(74),
1H(.) = O(75) END $

```

The above example shows how a charcode:directive may be used to define a "standardized" character set and encoding scheme for interchange between both human and mechanical processors -- another example might have shown a definition for the ASCII code. The following brief example shows a charcode:directive for defining an "inverted" collating sequence for the digits 0 through 9:

```

'CHARCODE      Q  1H(0)
  BEGIN        1H(0) = O(11), 1H(1) = O(10),
                1H(2) = O(7), 1H(3) = O(6),
                1H(4) = O(5), 1H(5) = O(4),
                1H(6) = O(3), 1H(7) = O(2),
                1H(8) = O(1), 1H(9) = O(0) END $

```

3.2.2.4 Operations on Literal Data

3.2.2.4.1 Literal:Assignment:Statements

Literal:assignment:statements have the form:

```

literal:variable   θ = θ literal:formula   θ $

```

where literal:formula is either a literal:variable, a literal:constant, a literal:function, an octal:constant or a hexadecimal:constant. Note that a literal:variable or a literal:constant means either a hollerith:variable or hollerith:constant or a variable or constant defined with respect to a charcode:directive.

If, in a literal:assignment:statement, both literal:variable and the evaluated literal:formula have the same length, the following facts apply:

- No conversion of one literal:item type to another takes place if literal:variable and literal:formula are either both hollerith or both defined with respect to the same charcode:directive, or if literal:formula is a hexadecimal:constant or an octal:constant.
- Otherwise, a conversion takes place, byte by byte, according to the one or more charcode:directives which may be involved.

A literal:formula byte is converted to a literal:variable byte in the following way: First, it is determined if the bit string content of the literal:formula byte is a member of the bit string set of the encoding scheme associated with the literal:formula. If not, the corresponding literal:variable byte is filled with the bit string representation of the fill:character associated with the literal:variable. If the bit string content of the literal:formula byte is a member of the encoding scheme associated with the literal:formula, it is determined if the character from the character set associated with the literal:formula which corresponds to this bit string is also a member of the character set associated with the literal:variable. If so, the literal:variable byte is filled with that bit string from the encoding scheme associated with the literal:variable which corresponds to this character. If not, the literal:variable byte is filled with the fill:character bit string associated with the literal:variable. Note that, by definition, the fill:character associated with the hollerith character set/encoding scheme is a space.

If, in literal:assignment:statements, a literal:variable is shorter than the evaluated literal:formula, excess bytes are truncated starting at the left for the purposes of assignment. (Conversions are made as required.) If, in literal:assignment:statements, literal:variable is longer than the evaluated literal:formula, excess bytes of literal:variable are filled with the fill:character associated with the literal:variable. (Conversions are then made, as required.)

Note that in procedure:call:statements, assignments of literal:formulas used as actual:input:parameters to formal:input:parameters takes place according to the above rules, as do assignments of literal:formulas to actual:output:parameters.

3.2.2.4.2 Literal Comparisons

Comparisons of literal data are effected by means of literal:relational:formulas of the form:

literal:formula θ relational:operator θ literal:formula

(Note that "chained" relational:formulas may also be used -- the description above is used for simplicity and applies to the chained case by extension.)

When a literal:relational:formula is evaluated, first the literal:formulas are evaluated. Then, if one of the literal:formulas is of a literal type different from the other, both are converted to hollerith character strings, and the literal:relational:formula is evaluated by comparing the hollerith strings. Otherwise, no conversion is made. (No conversion takes place if either literal:formula is an octal: or a hexadecimal:constant.) The literal:formulas so evaluated are compared, in effect, byte by byte from left to right. This is not meant to imply that implementations are required to perform actual byte by byte comparisons, but a given implementation's means of comparison must yield the same result.

One byte is $\left\{ \begin{array}{l} \text{equal to} \\ \text{less than} \\ \text{greater than} \end{array} \right\}$ another byte according as the
bit pattern of the first byte is $\left\{ \begin{array}{l} \text{equal to} \\ \text{less than} \\ \text{greater than} \end{array} \right\}$ the bit pattern of

the second byte, where both bit patterns are treated as unsigned: integers, ω bits in length.

Two literal: formulas of the same length are compared from left to right, a byte at a time. The two formulas are equal if all bytes are equal. Otherwise, the literal: formula on the left of the relational: operator is $\left\{ \begin{array}{l} \text{less than} \\ \text{greater than} \end{array} \right\}$ the literal: formula on the right of the relational: operator, if the byte by byte comparison finds at some byte position, that a byte in the left literal: formula is $\left\{ \begin{array}{l} \text{less than} \\ \text{greater than} \end{array} \right\}$ the corresponding byte in the right literal: formula.

If the two literal: formulas are of differing lengths, the following rules apply:

- If the shorter literal: formula is an octal: constant or hexadecimal: constant, it is made to be the same size as the larger by adding the appropriate number of zero bits at the left of the shorter literal: formula.
- If the shorter literal: formula is not an octal: constant or a hexadecimal: constant it is made to be the same size as the larger by adding one or more fill: characters of the associated character set/encoding scheme at the left of the shorter literal: formula.

Once the two literal: formulas are made the same size, they are compared according to the scheme described above for literal: formulas of the same length. When evaluated, a literal: relational: formula yields the value "true" if and only if the relational: operator is

EQ		equal to
NQ		less than, greater than
LS	and the comparison yields	less than
LQ		equal to, less than
GR		greater than
GQ		equal to, greater than

Otherwise, the literal: relational: formula yields the value "false".

3.2.2.4.3 Byte

When BYTE operates on a named: literal: variable a literal: variable is produced with the same type as the named: literal: variable operand. If the named: literal: variable is of type

Hollerith, so is the BYTE literal:variable. If the named:literal:variable is of a type declared in a charcode:directive, the BYTE literal:variable is of the same type.

3.2.2.5 Computer Representation of Numeric Constants and Variables

The functional characteristics of all manipulations involving numeric constants and variables are described in terms of a signed magnitude representation. The implementor's representation need not be signed magnitude, but the implementor's representation should, in essence, cause the program and the computer to impinge upon the environment in the same way as would occur if the actual representation were signed magnitude. However, it is recognized that it is infeasible in practice to ensure a signed magnitude equivalent effect in all areas. For example, an implementor using a one's complement fixed point representation may decide that the cost of guaranteeing that

BIT(\$1,5\$) (ALPHA)

will produce the magnitude of the integer ALPHA as declared by

ITEM ALPHA I 6 S \$

is not warranted by the compatibility which would be gained. Implementors, then, may depart from a total signed magnitude simulation, but in doing so must publish warnings where these departures are concerned.

3.2.2.6 Precedence of the Unary Operators

The precedences of the arithmetic:operators in the evaluation of a numeric:formula, from highest to lowest will be changed as follows:

<u>Current</u>	<u>New</u>
Unary-	**
**	Unary -
*/	*/
+ -	+ -

Note that this means that $-x ** 2 = -4$ when $x = 2$ not $+4$. The precedences of the arithmetic:operators complies with the conventions of algebra.

3.2.2.7 Extended:define:directive

This facility extends the capability of the define:directive in that it allows the programmer to include a parameter list with the defined name.

3.2.2.7.1 Form of the Extended:define:directive

An extended:define:directive means either:

```
DEFINE  $\theta$  name  $\theta$  "symbols"  $\theta$  $
```

or

```
DEFINE  $\theta$  name  $\theta$  ( $\theta$  formal:define:parameter:list  $\theta$ )  $\theta$  "symbol"  $\theta$  $
```

A formal:define:parameter:list is either a formal:define:parameter or a formal:define:parameter:list θ , θ formal:define:parameter. A formal:define:parameter is a name.

Symbols denotes a string of one or more of the JOVIAL symbols and space or spaces. Note that two consecutive primes are not allowed in the symbols string; also the last symbol before the second set of primes must not be a prime.

Note that no comment may appear in the extended:define:directive between the right parenthesis and the first " . The names making up the formal:define:parameter:list must be unique within the formal:define:parameter:list but may appear anywhere else in the program in any context whatever.

A name appearing in a formal:define:parameter:list serves no other purpose than to define an argument for the extended:define:directive. Formal:define:parameter:list names may each appear any number of times (or not at all) inside the quoted string of symbols appearing with the formal:define:parameter:list in the extended:define:directive.

3.2.2.7.2 Scope of the Definition

The scope of the definition of an extended:define:directive begins at that point in the source text where it occurs and ends at the end of the program (TERM) or at some subsequent point where another extended:define:directive with the same defined:name occurs, or where a define:directive with the same defined:name occurs. Note that the scope of a define:directive terminates at a subsequent point in the program where an extended:define:directive for the same name occurs.

3.2.2.7.3 Extended:define:invocation

An extended:define:invocation has the form

```
defined:name  $\theta$  ( $\theta$  actual:defined:parameter:list  $\theta$ )
```

or

```
defined:name  $\theta$  ( $\theta$ )
```

and may appear anywhere within the scope of the extended:define:directive with which defined:name is associated except within another extended:define:directive. Note that the extended:define:invocation is not recognized as such when it appears within a status:constant, a comment, or literal:constant.

An actual:define:parameter:list is actual:define:parameter or, actual:define:parameter:list θ , θ actual:define:parameter.

An actual:define:parameter is "symbols".

Symbols denotes a string of one or more of the JOVIAL symbols and space or spaces. Note that two consecutive primes are not allowed in the symbols string; also the last sign before the second set of primes must not be a prime.

If the actual:formal:parameter:list contains a larger number of actual:define:parameters than there are formal:define:parameters in the formal:define:parameter:list, the right most excess actual:define:parameters are ignored. If the number of formal:define:parameters exceeds the number of actual:define:parameters, the right most missing actual:define:parameters are assumed to be "space". In either case a warning to the programmer will be generated.

The effect produced by an extended:define:invocation is as follows:

- 1) A copy of the quoted symbol string associated with the extended:define:directive is made,
- 2) Each formal:define:parameter appearing in the copy is replaced with the symbols of its correspondent actual:define:parameter, and
- 3) The extended:define:invocation is replaced by the expanded copy.

At this point the compiler will process the expanded copy at its leftmost space, spaces, or symbol in exactly the same way as if it were part of the source text being input from the source text input unit. A name which followed DEFINE in a (currently active) define:directive which now appears in the expanded copy is treated at the time it is encountered in exactly the same manner as it would be treated had it appeared in the source text input stream. The same is true for extended:define:invocations, define:directives, and extended:define:directives which may have appeared in the expanded copy.

Note that names appearing after DEFINE in a define:directive, extended:define:invocations, define:directives, and extended:define:directives may appear in the expanded copy produced by an extended:define:directive. As with the use of a define:directive, the use of an extended:define:directive must not result in circular definitions.

Note that the symbols making up an actual:define:parameter, as well as the symbols in the symbols string of an extended:define:directive, may contain names which appeared after DEFINE in a (currently active) define:directive, or define:directives, or extended:define:directives, or extended:define:invocations. These names, directives, or invocations are treated simply as symbols in this context and do not produce replacements or expanded copies until after the expanded copies replace the extended:define:invocations which produced them.

Example:

The extended:define:directive

```
DEFINE AA (XX, YY, ZZ) " BB ($ SIN(ZZ), YY,XXS) + XX " $
```

causes

```
CC = BETA + AA ("P1", "P2", "P3+P4") $
```

to be "read" by the compiler as

```
CC = BETA +
      BB ($ SIN (P3+P4), P2, P1 $) + P1 $
```

3.2.2.8 Extended Mode Directive

This facility extends the mode:directive of AFM 100-24 in that it allows the programmer to select an item:description for an undeclared name on the basis of the first letter of that name. The form of an extended:mode:directive is MODE θ extended:mode:list θ \$.

An extended:mode:list is

```
extended:mode:list:element θ,θ extended:mode:list:element
```

or

```
extended:mode:list θ , θ extended:mode:list:element
```

An extended:mode:list:element is either a letter:mode:specification or a simple:mode:specification; the latter must appear as an extended:mode:list:element in the extended:mode:list but it must appear only once.

A simple:mode:specification is either item:description or item:description θ P θ optionally:signed:constant.

A letter:mode:specification is letter $\theta = \theta$ simple:mode:specification.

If more than one letter:mode:specification appears as an extended:mode:list:element, the letters in these letter:mode:specifications must be distinct.

The scope of effect of an extended:mode:directive beings at the place in the source text where it appears and ends as the end of the program (TERM) or at the next mode:directive or extended:mode:directive. Any name appearing within the scope of the extended:mode:directive, which has not been formally declared elsewhere is acted on by the extended:mode:directive in the following ways: If the first letter of the name matches a letter appearing in one of the letter:mode:specifications, the name is automatically declared as an item with the associated simple:mode:specification, otherwise, "free standing" simple:mode:specification.

Note that an extended:mode:directive serves to terminate the scope of effect of any mode:directive which occurred previously in the source text.

Example:

The following extended:mode:directive will mode declare all undeclared names in its scope beginning with I, J, K, L, M, or N as integer:items and all other undeclared items in its scope as floating.

```
MODE      I = I 48 S, J = I 48 S, K = I 48 S
          L = I 48 S, M = I 48 S, N = I 48 S,
          F $
```

The following two examples are equivalent to the extended:mode:directive above:

```
MODE      F, I = I 48 S, J = I 48 S
          K = I 48 S, L = I 48 S,
          M = I 48 S, N = I 48 S $
```

```
MODE      I = I 48 S, F, J = I 48 S,
          K = I 48 S, L = I 48 S,
          M = I 48 S, N = I 48 S $
```

3.2.2.9 Alternate Entrances to Procedures and Functions

Frequently it is desirable to enter a procedure (or function) at some point other than the beginning. This facility provides such a capability.

3.2.2.9.1 Alternate:Procedure:Entrance:Declaration

An alternate:procedure:entrance:declaration has the form

```
'ENTRANCE  $\theta$  alternate:entrance:name  $\theta$  $
```

or

```
'ENTRANCE  $\theta$  alternate:entrance:name  $\theta$   
( $\theta$  optional:formal:input:parameter:list  $\theta$ )  $\theta$  $
```

or

```
'ENTRANCE  $\theta$  alternate:entrance:name  $\theta$   
( $\theta$  optional:formal:input:parameter:list  $\theta = \theta$  formal:output:parameter:  
list  $\theta$ )  $\theta$  $
```

An alternate:entrance:name is a name.

An alternate:procedure:entrance:declaration is meaningful only if it appears in a procedure:declaration or a function:declaration, either in the procedure:heading or procedure:body of those declarations. (The containing procedure:or function:declaration is called the master:declaration.)

As many alternate:procedure:entrance:declarations as desired may be made within a master:declaration.

For each name declared as an alternate entrance by an alternate:procedure:entrance:declaration, there must be a matching statement:name within the procedure: or function: body part of the master:declaration. The position of this statement:name in the body defines the point of entry into the body when the procedure or function is invoked by its alternate:entrance:name.

Any alternate:procedure:entrance:declaration must occur prior to its associated statement:label.

Note that there must not be a statement:name in the body of the master:declaration which is the same as the name defined as a procedure: or function:name of the master: declaration.

Formal:parameter:names associated with an alternate:procedure:entrance:declaration may appear as formal:parameter:names in the master:declaration or in any other alternate: procedure:entrance:declaration; formal:input:parameters in any one such declaration may be formal:output:parameters in any other such declaration, and vice-versa.

However, as with procedure:declarations, names appearing as formal:parameters in any alternate:procedure:entrance:declaration may be formal:input:parameters or formal:output:parameters, but not both.

All formal:parameter:names (other than statement or close:names) whether they appear in the master:declaration or in alternate:procedure:entrance:declarations, must appear in data:declarations prior to their first reference in the body of the master:declaration.

Note that, with respect to the main:program, an alternate:entrance:name is a global Category 3 name. That is, it may only be referenced in an alternate:procedure:call or in an alternate:function:call and is not regarded as a statement:name.

From within the body of the master:declaration, a statement:name which matches an alternate:entrance:name is treated as a local statement:name when it appears as a switch:designator or as the object of a GOTO.

(Note that alternate:procedure:entrance:declarations can be embedded in function:declarations as well as procedure:declarations.)

3.2.2.9.2 Alternate Procedure Calls

The form of an alternate:procedure:call is precisely like that of a procedure:call, and the same restrictions as to the matching of number, position, and type of formal:parameters with actual:parameters apply. Prior to entry, actual:input:parameters are evaluated as required and bound to formal:input:parameters. Then, control is passed to the associated statement:name within the body of master:declaration.

At this point, execution takes place in exactly the same manner as occurs when a procedure with no alternate:procedure:entrance:declarations embedded in its declaration is invoked. If the flow of execution causes reference to be made to a formal:parameter which is not a formal:parameter of the alternate:procedure:entrance:declaration associated with an alternate:procedure:call, then such a reference is undefined.

On normal exit (exit achieved by other than a direct or indirect transfer to a main:program:statement:name not an actual:parameter), formal:output:parameters associated with the entrance invoked are bound to their actual:output:parameters, and control is returned as it would be from a procedure or a function with no alternate:procedure:entrance:calls embedded in its declaration.

3.2.2.9.3 Alternate:Function:Entrance:Declaration

An alternate:function:entrance:declaration has the same form as an alternate:procedure:entrance:declaration except that no formal:output:parameter:list is allowed; also an item with the same name as the alternate:function:entrance:name must be declared prior to its first

reference. Otherwise, all the rules applying to alternate:procedure:entrance:declarations apply to alternate:function:entrance:declarations. (Note that alternate:function:entrance:declarations may be embedded in procedure:declarations as well as function:declarations.)

3.2.2.9.4 Alternate Function Calls

The form of an alternate:function:call has the same form as a function:call. All remarks regarding actions initiated by an alternate:function:call parallel the remarks made regarding alternate:procedure:calls except those involving the binding of output:parameters; exit in this case results in the "replacing" the alternate:function:call with the value output as a result of that call.

3.2.2.9.5 Example

```
PROC CFM (FEET = MILES) $ "CONVERT FEET TO MILES"
```

```
    ITEM FEET F $
    ITEM MILES F $
    'ENTRANCE CMF (MILES = FEET) $ "CONVERT MILES TO FEET"
```

```
BEGIN
```

```
    ITEM FEET 'PER' MILE F P 5280.0 $
    MILES = FEET/FEET 'PER' MILE $
    RETURN $
```

```
CMF.
```

```
    FEET = MILES * FEET 'PER' MILE $
```

```
END
```

3.2.2.10 Extended Precision

Many computers programmed with JOVIAL have "double" precision hardware capabilities. In Standard J3, no means exist for making use of such capabilities. The feature described in this paragraph permits specification of an extended precision floating:point:item. It is expected that the great majority of command and control extended precision computational requirements can be met by an extended precision floating:point facility and, therefore, no extended precision fixed:point or integer capabilities are proposed; further, it is assumed that the exrad of an extended precision floating:point:number is the same size as the exrad of an ordinary floating point number.

ρ is used to denote the implementation dependent number of bits in the significand of the extended precision floating:point:number. σ is used to denote $\rho - 1$. The number of

bits per exrad of both ordinary and extended precision floating numbers is represented by ν . The adjective extended:precision will be used to describe extended precision floating:point data and the adjective ordinary will be used to describe floating:point data which are not of extended precision.

Descriptions for all ordinary:floating attributes will be found in AFM 100-24 wherever the floating adjective is used in that document.

Neither adjective will be used if it is clear from the context which kind of floating:point is meant nor will adjectives be used if it is clear from the context that either or both kinds are meant.

3.2.2.10.1 Constant Form

The form of an extended:precision:floating:constant is ordinary:floating:constant D.

Examples are:

3.D
3.14159D
.14159D
3.14159E-10D
.14159E-10D
3.E-10D

3.2.2.10.2 Declaring Extended:precision:floating:items

Extended:precision:floating:items are declared in item:, array:, and table: declarations by means of the extended:precision:floating:item:description, the form which is either

F θ D
or
F θ D θ R

(Where R specifies rounding on assignment to the declared datum.)

3.2.2.10.3 Computer Representation of Extended:Floating Constants and Variables

Extended:precision:floating:constants and extended:floating:items are represented as two signed numbers. The significand is represented in p bits; the exrad in ν bits.

3.2.2.10.4 Numeric:formulos and Numeric:ossignment

A unary arithmetic:operotor (unary minus, absolute value) operating on an extended:precision:floating variable produces an extended:precision:floating result. A binary arithmatic:operator between two operonds, one of which is extended:precision:floating produces an intermediate result which is extended:precision:floating. Thus, an extended:precision:floating result is generated if one operand is ordinary:floating and the other extended:precision:floating.

All colculotions in extended:precision:floating form ore truncoted to σ significant bits unless the statement colls for rounded ossignment. Then, each colculation is carried out to ρ significant bits and rounded by odding the extra bit into bit ω , discarding the extra bit, and renormalizing if necessary. The result of the rounding must be to produce the same effect os ordinary:floating rounding requirements, i.e., rounding must either produce the some value os would be obtained by truncotion or else put the value further from zero.

Necessary conversions between extended:floating integer and fixed dato ore carried out outomaticolly when these dato appear os operands in o numeric:formulo or numeric:assignment:stotements.

All remarks applying to ordinary:floating dato in numeric:formulos and numeric:assignment:stotements not specifically contradicted by the remarks above apply to extended:precision:floating dato os well.

3.2.2.10.5 Relotional Operotions

When o fixed, integer, or ordinary:floating operand appears on either side of o relational:operotor and the other operand is extended:precision:floating, the fixed, integer or ordinary:floating operand is converted to extended:precision:floating for the purposes of the comparison.

3.2.2.10.6 Allocation of Extended:precision:floating:items

Simple:extended:precision:floating:items ore allocoted os many full δ bit consecutive words as ore required to contain the items $\rho + \sqrt{\quad}$ bits. Any remaining bits in the container ore set to zero.

Each element of an extended:precision:floating:orroy is allocated space in the same manner os for simple:extended:precision:floating:items. Extended:precision:floating:items which appear in o non-packed ordinary:table ore allocoted space in the same manner os simple:extended:precision:floating:items. Medium-packed and dense-packed ordinary:tables may cause extended:precision:floating:items to cross word boundaries. This is true for defined:entry:tables os well. Parallel structuring of tables may, of course, cause parts of extended:precision:floating:items to be placed in non-consecutive words.

3.2.2.11 JOVIAL Input/Output Module

The proposed standard JOVIAL Input/Output MODULE (JIO) consists of four submodules: A file manipulation submodule (JFILE), a submodule which permits I/O devices to be controlled by the programmer (JDEV), a formatting submodule (JFORMAT), and a service routine submodule (JSERVICE). All submodules except JSERVICE consist of formal language features. JSERVICE is composed of a set of routines prepared to satisfy the requirements of a specific application by the programmers of that application.

In the descriptions of the submodules given below, reference is made to processors called DPEP, PFILE, and PFORM. These processors serve only as an aid in the description of actions taken by the execution of certain JIO statements; descriptions of them are not intended to be taken as specifications for subroutines which must be included in any implementation of the standard.

3.2.2.11.1 JDEV

The features in JDEV consist of:

- device procedures
- device:procedure:execution:statements
- a device:procedure execution processor (DPEP)

Devices are external equipments or other processor/memory configurations and are labeled by unique non-negative numbers. Device procedures are actions or set of action which devices may be commanded to perform. "Rewinding" and "initiating a read operation" are examples of device procedures which a tape transport device may be commanded to perform, for example. The commanding process will be described subsequently as "assigning a device procedure to a device for execution".

Device procedures are implementation dependent. A given device procedure defined for a given installation may be assigned to more than one device for execution. "Rewind" and "initiate read" will typically be executable by all or most tape drive devices in a particular installation.

Device procedures may have both input and output parameters associated with them much in the same manner as the ordinary JOVIAL procedures defined by procedure: declarations.

Note that formal:input:parameters and formal:output:parameters associated with a device procedure are of the same type as formal:input:parameters and formal:output: parameters associated with ordinary procedures.

A device procedure is assigned to a device for execution by means of a device:procedure:execution:statement the form of which is

```
'EXECUTE device:procedure:call  
($ device:number $)  
optional:completion:call $
```

A device:procedure:call has exactly the same form as a procedure:call:statement except that the terminating "\$" of the procedure:call:statement does not appear; all rules relating formal:parameters to actual:parameters in a procedure:call:statement apply correspondingly to a device:procedure:call. A device:procedure:name is a global Category 1 name. (Category 1, in fact, contains only device:procedure:names.)

Device:number is a numeric:formula. A completion:call is either a close:name or procedure:call where procedure:call takes the form of a procedure:call:statement without the terminating "\$". All rules relating formal:parameters to actual:parameters in a procedure:call:statement apply correspondingly to procedure:calls.

DPEP is a mechanism which provides an interface between devices and the execution of device procedures by means of device:procedure:execution:statements. DPEP must be able to determine:

- The status of each device at any time.
- The set of device procedures which may be executed by a given device (This set is in general a subset of the set of all possible device procedures).
- The set of device procedures which may be executed on a device when any given device procedure is currently being executed by that device. A device procedure is "currently being executed" from that moment when step (d), below, is entered until the device signals completion of execution of the device procedure as indicated in step (g) below. (Note that it may be possible to execute a device status checking device procedure on a device, while the device is executing another device procedure, for example.) A device is said to be "available" to a device procedure if that device procedure is a member of such a set as described above; otherwise, the device is "unavailable" to the device procedure.

Associated with each device procedure is an "execution time" property. Some device procedures (such as device status checking) may have an execution time of zero -- they execute "instantaneously". Other device procedures have a non-zero execution time -- they execute "non-instantaneously". DPEP is designed in such a way as to allow the programmer to perform processing in parallel with the execution of non-instantaneous device procedures and to perform interruption processing when these device procedures have signaled completion. It is the implementor's decision which of the device procedures are defined to be instantaneous and which are to be non-instantaneous.

A device:procedure:execution:statement causes the following operations to be performed:

- 1) Any actual:input:parameters associated with the device:procedure:call will be evaluated in precisely the same manner as actual:input:parameters are evaluated in a procedure:call:statement.
- 2) The device:number:numeric:formula will be evaluated exactly as if the numeric:formula were being used as subscript.
- 3) The DPEP mechanism will perform the following operations in the order indicated:
 - (a) If the device:number does not correspond to any of the device:numbers allowed, DPEP ceases processing and returns control to the next statement following the device:procedure:execution:statement. (Attempt to execute the device procedure results in a no-operation)
 - (b) DPEP determines if the device procedure specified in the device:procedure:call may be executed by the device specified by device:number. (A device procedure for backspacing a tape record cannot be executed on a card reader, for example) If the device procedure cannot be executed on the device, DPEP returns control to the next statement following the device:procedure:execution:statement. (Attempt to execute the device:procedure results in a no-operation.)
 - (c) DPEP determines if the device is "available" to the specific device procedure. If not, DPEP returns control to the next statement following the device:procedure:execution:statement. (Attempt to execute results in a no-operation.)
 - (d) If the device procedure specified is non-instantaneous, DPEP takes all necessary steps to regain control from any program segment which may be operating when the device signals that it has terminated execution of the device procedure.
 - (e) DPEP assigns the device procedure to the device for execution. If the device procedure executes instantaneously on the device, control is returned to the statement following the device:procedure:execution:statement. Then, steps (g) - (i) are never entered. If the device procedure is non-instantaneous, control is passed to step (f).
 - (f) DPEP returns control to the statement following the device:procedure:execution:statement.
 - (g) On the signal (interruption) from the device that it has completed processing of the device procedure, execution of the program segment currently operating is suspended, DPEP ensures that any actual:output:parameters associated with the device procedure are evaluated, and control is passed to step (h) below.
 - (h) DPEP evaluates any actual:input:parameters associated with the completion:call, if a completion:call was supplied. (Evaluation takes place exactly as if the completion:call were an ordinary procedure:call.) If no completion:call was specified, DPEP resumes processing at step (i). Otherwise, the

close or procedure specified by the completion:call is invoked and DPEP awaits its termination. Upon termination of the completion:call execution, DPEP passes control to step (i).

- (i) DPEP returns control to that point immediately following the last execution which occurred prior to regaining control on the termination signal from the device. Note that:
- A completion:call associated with an instantaneous device procedure is never operated (see step "e").
 - If the completion:call procedure or close does not return control to DPEP at step (g), further DPEP processing is undefined.
 - If the completion:call specified invocation of a close which was declared within a FOR statement which does not contain the device: procedure:execution:statement, and if that close manipulates the loop: variable defined by the FOR statement, the completion:call invocation step (g) is undefined.
 - If the completion:call specifies invocation of a procedure, and if one or more of the actual:parameters contains a loop:variable then the device: procedure:execution:statement must appear within the scope of definition of the loop:variables.

Examples of device:procedure:execution:statements are:

```
'EXECUTE REWIND ($23$) $  
'EXECUTE REWIND ($1+7$) $
```

"THE ABOVE TWO STATEMENTS WILL REWIND THE TAPES ON DEVICES 23 AND 1+7."

```
'EXECUTE DISKREAD (DISKADDRESS, MEMORY 'BUF, BUF'LENGHT  
= DISKREAD 'STATUS)  
($ DISKA $) INTRUPT'L( ) $
```

"THE ABOVE STATEMENT CAUSES A READ FROM DISKADDRESS TO BE INITIATED. THE DISKREAD DEVICE PROCEDURE WILL FILL DISKREAD 'STATUS WITH A VALUE DESCRIBING THE STATUS OF THE READ ON COMPLETION. THEN THE PROC INTRUPT'L IS OPERATED."

3.2.2.11.2 JFILE

The file processing submodule, JFILE, consists of the following features:

- the file:declaration
- file manipulation statements
- a control mechanism, the File Processor (PFILE)
- two built-in interface procedures, TLB and TBL

- the functional:modifier POS
- the functional:modifier 'FILENUM.

These features described fully in 3.2.2.11.2.4 have been designed and organized in such a manner as to reflect a point of view founded on a specific definition of file, and on the notion that files so defined have both "functional" and "representational" characteristics which can, and should be separated. The definition of file is:

- a collection of distinguishable objects, or records, called a record set, and
- the processes by which records in the record set are stored into and retrieved from storage media.

A file characteristic is "functional" if it applies to a record set or storage/retrieval processes independently of particular storage media and transmission devices. A file characteristic is "representational" if it applies to particular properties of a record set in some specific, physical storage medium or if it describes storage/retrieval processes in terms of some specific, physical transmission device. JFILE is so structured as to separate the functional characteristics of a file as completely as possible from the representational characteristics, thus rendering functional processes and representation characteristics independent.

3.2.2.11.2.1 The Functional File

3.2.2.11.2.1.1 The Functional Record Set

Each record in the functional record set is thought of as a "logical" or functional entity independently of its relationship to any other information group; functional records are generally defined by the programmer on the basis of certain functional characteristics of his program. Each (functional) record in the record set has no particular structure and is composed of no particular type of information as such. The programmer gives structure and information type to a record by the manner in which his program deals with its "main memory" image.

A record set contains n functional records, where n is some finite positive integer or 0. If $n=0$, the record set is called "empty". (In practice, the programmer does not ordinarily know the value of n .) Associated with each record in a (non-empty) record set is a unique non-negative integer k which comes from the set k ($k = 0, \dots, n-1$) where n is the number of records in the record set. This set of numbers is called the functional "reference set".

3.2.2.11.2.1.2 The Functional Storage/Retrieval Processes

This paragraph describes the referencing process consisting of four basic storage/retrieval processes (Input, Output, Insert, Delete), and two processes (Open, Shut) which serve

to initialize and terminate other storage/retrieval processes. When "transmission" is used below it means only the movement of information and implies no alteration of that information. Thus, transmission of a record image from the main memory to an external storage medium, followed by transmission of the same record back to main memory, should produce an identical record image in main memory if the transmission in each case is performed under the same set of conditions. ("Main memory" refers to the storage medium which contains the invocation command for a storage/retrieval process.)

Note that no notions of transmission initiation, transmission occurring in parallel with computation, and interruption on completion of transmission are associated with the functional storage/retrieval and initialization/termination processes, though, indeed, any representation of these functional processes will have to deal with such notions. From the point of view of the programmer, the execution of any functional process involving a transmission causes the transmission to take place and does not terminate before the transmission is completed; also, when a functional storage/retrieval or initialization/termination process completes execution, program control is passed to the statement following the process invocation. Further, this should be true for any representation of the functional processes.

A record in a record set is "referenced" by specifying the appropriate integer k from the reference set. A record is referenced when it is desired to use it as an operand in a storage/retrieval process. If a record k is referenced with $k(0)$ or $k(n-1)$ the result of the reference is, in general, undefined. (With respect to the basic functional storage/retrieval processes described below, this will be true unless explicitly stated as otherwise.) The following paragraphs describe the functional processes:

- a) "Input record k " is a retrieval process which causes the record referenced by k in the record set to be transmitted to the main memory. "Input record k " is undefined if $n=0$, i.e., if the record set is empty.
- b) "Output as record k " is a storage process which causes information to be transmitted from the main memory to a storage medium. If the record set in the storage medium already contains a record k , it is replaced by the transmitted information; the transmitted information is then established as record k of the record set. If $k=n$, the transmitted information is appended to the record set and the number of records in the record set is increased to $n + 1$.
- c) "Insert as record k " is a storage process which causes information to be transmitted to the storage medium. If $k = n$, "Insert as record k " has the same meaning as "Output as record k ". Otherwise, the transmitted information is established as record k of the record set. All those records of the record set which were referenced by $j \geq k$ prior to the invocation of "Insert" are referenced by $j + 1$ subsequent to the invocation of "Insert". "Insert" causes the number of records in the record set to be increased from n to $n + 1$.

- d) "Delete record k" is a storage process which causes record k of the record set to be removed from the record set. Records of the record set which were referenced by $j > k$ prior to the invocation of "Delete" are referenced by $j-1$ after invocation of "Delete". If $n = 1$, "Delete" causes the record set to be made empty.
- e) "Open file" is an initialization process which causes the record set and the storage/retrieval processes of the file to be readied for processing. The storage/retrieval processes "Input", "Output", "Insert", and "Delete" may be invoked only after "Open file" has been invoked.
- f) "Shut file" is a termination process which causes the record set and the storage/retrieval processes to be brought to a state of inactivity. Once "Shut file" has been invoked, "Input", "Output", "Insert", or "Delete" may not be invoked until "Open file" is invoked once again.

3.2.2.11.2.2 The Representational File

3.2.2.11.2.2.1 The Representational Record Set

Each record in the representational record set is an information group the attributes of which reflect properties peculiar to a particular storage medium and transmission (e.g., parity mode). Such information group attributes may also reflect the programmer's desire to conserve space in the storage medium, as when he blocks many functional records together into one representational record.

Representational records will have a structure and information type peculiar to the associated storage medium and transmission device. A representational record set may contain greater or fewer records than the number of records in the functional record set being represented. A representational record set has an associated reference set that is peculiar to the particular storage medium and transmission device being used. Such a reference set may consist of disk or drum addresses, or it may contain "position" numbers which specify where the record is on a tape, in a card reader, or on a printer.

It is important to note that a given functional record set may have associated with it a wide variety of representational record sets if the functional record set is to be represented in a variety of different storage media and transmitted by a variety of different transmission devices.

3.2.2.11.2.2.2 The Representational Storage/Retrieval Processes

"Transmission", as used in this paragraph may involve some alteration of information, as for example, conversions between internal and external hollerith encoding techniques.

The actual transmission associated with an Input or Output typically requires initiation of an Input or Output Operation and processing of the interrupt signal returned by the transmission device when it has completed the transmission. Thus, representational storage/retrieval processes may allow other forms of processing to occur between transmission initiation and completion. Device manipulations of all kinds may be required to effect the transmission; these manipulations may indeed be commanded by device:procedure:execution:statements, (JDEV).

A representational record is referenced by specifying its associated reference set element. This element may be an explicit disk or drum address or it may appear implicitly in a command, as in "read the next tape block". The following describes the representational storage/retrieval processes.

- a) To "Input" or "Output" a representational record will involve commanding some transmission device to move the representational record between main memory and a storage medium. Device commanding requires, of course, the specification of a variety of device specific parameters. "Outputting" a record may involve "Inputting" other records of the record set, as when dealing with disk records which are "linked" by disk addresses embedded in each record as part of the data of that record. Representational "Inputting" and "Outputting" may also involve commanding devices to perform non-transmission processes, e.g., backspacing, rewinding, positioning read/write heads, etc.
- b) "Inserting" representational records may involve both "Inputting" and "Outputting" as may "Deleting" representational records.
- c) "Opening" a representational file may involve securing a device, allocating main memory buffer space, initiating the "Input" of a physical record, and the like. "Shuting" a representational file may involve releasing a device and main memory buffer space.

It is clear that different representational storage/retrieval processes will need to be applied against the same functional record set represented in different storage media and transmitted by different devices. Also, it is clear that certain functional processes will be feasible for one file representation and not for another so that a functional file may be limited in the number of representations that can be defined for it. For example, a file requiring "Output" to "rewrite" a record cannot usually be represented feasibly where magnetic tape is a storage medium.

3.2.2.11.2.3 Functional and Representational Specifications and Interfaces in JFILE

JFILE allows for the manipulation of files in both a functional and a representational manner. Explicit language features are provided to deal with files functionally (see 3.2.2.11.2.4, below). The functional record set and reference set as well as the functional storage/retrieval processes of paragraph 3.2.2.11.2.1 are embodied in these language features. The

representational particular file aspects are dealt with in terms of "Service Routines" which are tailored to the specific storage media and transmission devices of a particular application. Service Routines are either closes or procedures which are written by the programmers for a specific application.

A Service Routine is associated with each of the functional storage/retrieval processes which will be carried out during the processing of the file; the association of Service Routine with functional process is effected by means of the file:declaration.

Service Routines handle all representational aspects of the file. In general, embedded within their structure will be facilities for effecting the following:

- device assignment
- recording modes and densities
- record set labelling
- internal buffering
- initiation of device procedure executions
- interrupt processing
- external storage media allocation and management
- transmission error checking and recovery procedures
- sequential or random access

(Note that device manipulations might typically be handled by device:procedure:execution:statements)

The set of all Service Routines is referred to as the JSERVICE submodule of JIO. It is anticipated that a given application will fix on a set of (application) standard Service Routines, or use such Service Routines as are provided by an operating system. A Service Routine specified for a given functional storage/retrieval process should simulate and preserve all the functional characteristics of files as described in 3.2.2.11.2.1, though there is no way to guarantee that this be so. (For example, the Service Routine associated with "Input record k" should be so written as to make the correspondence between the functional reference k and any physical reference, and cause transmission of functional record k to occur as described in 3.2.2.11.2.1.2(a) regardless of any (representational) device manipulations or internal buffering techniques which may be required to do so).

The syntactic entity, service:call, is the mechanism provided for specifying the invocation of a Service Routine when performing a functional file operation. A service:call is a close:name or a procedure:call. A procedure:call is the same as a procedure:call:statement except that it does not contain a terminal "\$". When a service:call is said to be "operated" in paragraph 3.2.2.11.2.4, the effect obtained is exactly that obtained by either

GOTO close:name \$

or

procedure:call \$

Note that any actual:input:parameters of the procedure:call are evaluated in the usual way at the time the service:call is operated, and not before. If the operation of a service:call causes a close to be invoked which was declared in a FOR statement and which references the loop:variable of the FOR statement, the effect of the close invocation is undefined.

If the service:call is a procedure:call and if one or more of the actual:parameters contains a loop:variable, the file:declaration containing the service:call must also be within the scope(s) of definition of the loop:variable(s). Also any open:, input:, output:, insert:, delete:, or shut:statement which, when executed, will cause such a service:call to be operated must be within the scope(s) of definition of the loop:variable(s); otherwise the operation of the service:call is undefined.

3.2.2.11.2.4 The File:Declaration and File Statements

3.2.2.11.2.4.1 The File:Declaration

A file:declaration is

```
FILE θ file:name θ file:process:list θ $
```

File:name is a Category 3 name which serves to identify the file. A file:process:list is either

```
file:process
```

or

```
file:process:list θ file:process
```

A file:process is

```
MODE θ mode:integer θ
      open:process:clause θ
      optional:pos:process:clause θ
      optional:input:process:clause θ
      optional:output:process:clause θ
      optional:insert:process:clause θ
      optional:delete:process:clause θ
      shut:process:clause 0
```

Note that at least one of the optional clauses in a file:process must be included in file:process. Mode:integer is an unsigned:integer which serves to identify the file:process in which it is contained; if more than one file:process appears in the file:process:list, the mode:integer of any one must be different from the mode:integer of any other.

A file:process is, in effect, a specification for a particular "mode" of file initialization, storage/retrieval and termination processes which will be carried out together; the

file:process:list allows as many such "modes" as desired to be specified in the file:declaration.

The purpose of the clauses in file:process is to show a correspondence between the functional file operators OPEN, POS, INPUT, OUTPUT, 'INSERT, 'DELETE, and SHUT, and the representational service:calls by which they are realized as indicated below.

Open:process:clause	}	is	}	OPEN θ	service:call
pos:process:clause				POS θ	service:call
input:process:clause				INPUT θ	service:call
output:process:clause				OUTPUT θ	service:call
insert:process:clause				'INSERT θ	service:call
delete:process:clause				'DELETE θ	service:call
shut:process:clause		SHUT θ	service:call		

Service:calls are not operated at the time a file:declaration is made, but rather at the time a file statement is executed, as described below. Note that at execution time, PFILE, the file processing mechanism, has access to all information declared by a file:declaration.

3.2.2.11.2.4.2 File Statements

Statements performing file processing are:

open:statements
input:statements
output:statements
insert:statements
delete:statements
shut:statements

Additionally, file processing operations may take place whenever a value is assigned to the functional:modifier POS.

Open:, input:, output:, insert:, delete:, and shut:statements are the JOVIAL features provided for effecting the functional file processes "Open", "Input", "Output", "Insert", "Delete", and "Shut" discussed in 3.2.2.11.2.1.2. References to records in the functional record set are made by assigning values to functional:modifier POS (see 3.2.2.11.2.4.3). Thus, the invocation of the functional retrieval process "Input as record k" is carried out by first assigning k to POS and then executing an input:statement. Note that:

- file statements do not automatically change the value of POS when they are executed (though the associated Service Routines may be written in such a way to perform this action if desired)

- assignment of a value to POS does not automatically cause positioning of the storage medium with respect to the transmission device (though Service Routines may produce this result if desired).

A open:statement is

OPEN θ file:name θ MODE θ mode:formula θ \$

where mode:formula is a numeric:formula.

An input:statement is

INPUT θ file:name θ io:list θ \$

An output:statement is

OUTPUT θ file:name θ io:list θ \$

An insert:statement is

'INSERT θ file:name θ io:list θ \$

A delete:statement is

'DELETE θ file:name θ \$

A shut:statement is

SHUT θ file:name θ \$

An io:list is

io:operand

or

io:list θ , θ io:operand

An io:operand is either a named:variable or table:name or array:name.

Note that the phrase "currently opened" will be used subsequently to describe any file whose file:name appears in an open:statement which has been executed and where no execution of a shut:statement with the same file:name has yet taken place.

3.2.2.11.2.4.3 Functional: Modifier POS

POS is a functional modifier which designates an unsigned:integer:variable when applied to a file:name and can therefore be assigned values. POS may appear in two forms:

POS θ (θ file:name θ)

or

POS

When the second form of POS is used, a file:name is automatically supplied. That file:name is supplied which refers to a currently opened file and which appeared in the last open:statement, input:statement, output:statement, insert:statement, or delete:statement executed prior to the POS reference. POS is undefined if used when no file is currently opened or when a specific file:name (form 1) used refers to a file not currently opened. The value taken by POS is the reference number of a (logical) functional record in the functional record set associated with file:name. POS provides the mechanism for functional "Referencing" as described in 3.2.2.11.2.1.2. A further description of POS is given in 3.2.2.11.2.4.5.7 below.

3.2.2.11.2.4.4 Functional:Modifier 'FILENUM

'FILENUM is a functional:modifier which produces an unsigned:integer when applied to a file:name. 'FILENUM does not designate a variable and cannot be assigned values in any JOVIAL J3 program. 'FILENUM may appear in two forms:

'FILENUM θ (θ file:name θ)

or

'FILENUM

When the second form is used, a file:name is automatically supplied. That file:name is supplied which refers to a currently opened file and which appeared in the last open:statement, input:statement, output:statement, insert:statement, or delete:statement executed prior to the 'FILENUM reference. 'FILENUM is undefined if used when no file is currently opened or if a specific file:name (form 1) is used which refers to a file not currently opened.

Values taken by 'FILENUM always lie in the range 0. . .m, where $m + 1$ is the maximum number of files ever "currently opened" during the course of program execution. 'FILENUM is undefined if file:name refers to a file not currently opened.

The purpose of 'FILENUM is to provide an interface between file manipulation statements and the Service Routines by which they are realized. Thus a set of Service Routines might be so written as to reference file parameters maintained in a table, with the parameter sets for distinct files stored in distinct entries. Service Routines, when operated, would determine the appropriate set (entry) of file parameters to use by means of 'FILENUM, e.g., FILE 'PARAM (\$ 'FILENUM \$).

3.2.2.11.2.4.5 PFILE and File Statements

PFILE supervises the execution of all file statements.

3.2.2.11.2.4.5.1 Execution of the Open:Statement

The following functions are performed in the execution of the open:statement.

- (a) PFILE determines if the file called file:name has already been established as "currently opened". If so, execution of the open:statement is undefined.
- (b) PFILE creates a unique non-negative integer and assigns it to the 'FILENUM associated with this file. As previously stated, the value of 'FILENUM is in the range 0. . .m where m + 1 is the maximum number of files ever "currently opened" during the course of program execution.
- (c) The mode:formula of the open:statement is evaluated in the same manner as if it were a subscript.
- (d) PFILE determines if a file:process identified by the value yielded by evaluation of the mode:formula has been included in the file:declaration for the file called file:name. If no such file:process was included in the declaration, execution of the open:statement is undefined. Otherwise, PFILE establishes the file:process so identified as the "current file:process" for the file called file:name.
- (e) PFILE establishes the status of the file called file:name as "currently opened".
- (f) PFILE operates the service:call given in the open:process:clause of the current file:process.
- (g) When the service:call returns control to PFILE, PFILE returns control to the first statement following the open:statement.

3.2.2.11.2.4.5.2 Execution of the Input:Statement

The following functions are performed in the execution of the input:statement.

- (a) PFILE determines if the status of File called file:name is "currently opened". If not, execution of the input:statement is undefined.
- (b) PFILE then determines if an input:process:clause is included in the current file:process established for the file called file:name. If not, execution of the input:statement is undefined.
- (c) The io:list of the input:statement is established as the "current io:list" by PFILE. Subscripts of named:subscripted:variables appearing in the io:list are evaluated at this time, in the same manner that they would be evaluated had the named:subscripted:variables appeared in any other statement. Also, subscripts for named:subscripted:variables are evaluated in the same left to right order that the named:subscript:variables appear in the io:list. Note that these remarks about subscript evaluation apply whenever the phrase "established as the current io:list" appears in subsequent paragraphs.
- (d) PFILE operates the service:call associated with the input:process:list in the current:process:list.
- (e) When the service:call returns control to PFILE, PFILE returns control to the first statement following the input:statement.

3.2.2.11.2.4.5.3 Execution of the Output:Statement

The description of output:statement execution is identical to the description of input:statement execution (3.2.2.11.2.4.5.2) except that every occurrence of the word input should be replaced by the word output.

3.2.2.11.2.4.5.4 Execution of the Insert:Statement

The description of Insert:Statement execution is identical to the description of Input:Statement execution (3.2.2.11.2.4.5.2) except that every occurrence of the word input should be replaced by the word insert.

3.2.2.11.2.4.5.5 Execution of the Delete:Statement

The description of delete:statement execution is identical to the description of input:statement execution (3.2.2.11.2.4.5.2) except that subparagraph (c) does not appear (a delete:statement has no io:list) and every occurrence of the word input should be replaced by the word delete.

3.2.2.11.2.4.5.6 Execution of the Shut:Statement

The following functions are performed in the execution of the shut:statement.

- (a) PFILE determines if the file called file:name has been established as "currently opened". If not, execution of the shut:statement is undefined.
- (b) Then PFILE operates the service:call given in the shut:process:clause of the current file:process for the file file:name.
- (c) When the service:call returns control to PFILE, PFILE establishes the status of the file file:name as "not currently opened" and returns control to the first statement following the shut:statement.

3.2.2.11.2.4.5.7 Execution of Assignment to POS

The following functions are performed in the execution of the assignment to POS.

- (a) PFILE determines if file file:name has been established as "currently opened". If not, any reference to POS is undefined (see 3.2.2.11.2.4.3).
- (b) The value to be assigned to POS for file file:name is assigned.
- (c) If a pos:process:clause is present in the current file:process for file file:name, PFILE operates it. Otherwise PFILE returns control to the first program step following the assignment to POS.
- (d) When the service:call returns control to PFILE, PFILE returns control to the first program step following assignment to POS.

- (e) Note that the reason for allowing a service:call to be operated when a value is assigned to POS is to allow a "repositioning" of the transmission device over the representational record set well in advance of the next access to a record. A programmer requiring random access to a disk, say, might do a POS assignment (with service:call), then some computations, and finally execute an input:statement, realizing that the computations will be carried on while a disk read-head is "repositioning".

3.2.2.11.2.5 The Built-in Procedures TBL and TLB

TBL (Transmit Buffer to io:list) and TLB (Transmit io:list to Buffer) allow the applications programmer writing Service Routines to transmit between functional records (which have been brought into main memory from an external storage medium or which will be transmitted from main memory to a storage medium) and the internal data structures which are named in the io:list of input:, output:, and insert:statements.

The following functions are performed in the execution of TBL and TLB which operate under the control of PFILE.

- (a) TBL is invoked by

TBL (BUFFER, DISPLACEMENT, LENGTH) \$

TLB is invoked by

TLB (BUFFER, DISPLACEMENT, LENGTH) \$

BUFFER is table:name or array:name denoting the (main memory) buffer area which will receive the functional record from the io:list (TLB) or which contains the functional record to be transmitted to the io:list (TBL).

DISPLACEMENT is an integer:variable which is an increment from the first word address of BUFFER: TBL, and TLB add DISPLACEMENT to the first word address of BUFFER to produce the first word address of the functional record storage area.

LENGTH is an unsigned:integer:variable representing the length in words of the functional record storage area.

- (b) TBL causes the functional record characterized by the input:parameters to be transmitted to the internal data structures of the "current io:list". The "current io:list" is established by PFILE as described in 3.2.2.11.2.4.5. TBL treats the functional record as a bit string and the elements of the

"current io:list" as a set of bit strings, one for each named:variable, table, or array making up the "current io:list". A bit string corresponding to a named:variable consists of all bits required to represent the variable, including any filler bits. Bit strings corresponding to tables and arrays are composed of those (contiguous) blocks of words used to represent the tables and arrays. In the case of tables, the contiguous block contains the NENT word as the first word; even if the table is variable, the length of the block is based on the total number of words required to represent the table, regardless of the current value of the NENT. The transmission behaves as if the "current io:list" bit strings formed one contiguous bit string, concentrated together in the same order that the named:variables, tables, and arrays appear in the "current io:list". Each functional record bit, starting with the left most record bit, is moved one at a time to a corresponding bit in the (contiguous) "current io:list" bit string, starting with the left most current io:list bit. The move terminates when the functional record bit string is exhausted or when the "current io:list" bit string is exhausted, whichever happens first.

(The description of the transmission as a bit by bit move is given only to indicate the functional characteristics of the move. An implementation scheme which achieves the same functional end is permissible.)

- (c) TLB causes the "current io:list" structures to be moved to a functional record storage area whose first word address and size are characterized by the TLB input parameters. The effect of the transmission is identical to the TBL transmission except, of course, that the move is from the (contiguous) "current io:list" to the functional record storage area.
- (d) It should be noted that the io:list, however complex, is treated as one functional record.

3.2.2.11.3 JFORMAT

The JFORMAT submodule is composed of:

- the format:variable
- the encode:statement and the decode:statement
- the format processor, PFORM

The purpose of the JFORMAT features is to provide the facilities for converting data in hollerith character string representations to internal representations and vice-versa.

A decode:statement is used to convert data represented as hollerith character strings to their corresponding internal representation. An encode:statement is used to convert data from their internal representations to hollerith character string representations. The purpose of the format:variable is to supply the necessary conversion and editing specifications.

Encode: and decode:statements are thought of as executed under the supervision of the processor PFORM.

3.2.2.11.3.2.1 The Format:Variable

A format:variable is either a hollerith:constant or a named:hollerith:variable.

PFORM interprets format:variables to determine what conversions and editings are to take place. The hollerith characters making up the content of a format:variable need have no particular structure until PFORM is called upon to interpret the format:variable during the execution of an encode:statement or a decode:statement. Note that all or part of any format:variable may be modified at execution time by any of the character manipulation features of the language; this allows for varying conversion and editing specifications at execution time.

When a format:variable is being interpreted, the hollerith characters comprising it must take the form of a specification'list. (Note: Prime, and not colon, will be used in the syntactic specifications of the content of a format:variable to draw attention to the fact that such specifications apply at execution time and not at compile time. Prime plays the same role played by colon in the compile time metalanguage specifications; all other compile time metasyntactic symbols have the same meaning when used in execution time metasyntactic symbols.)

A specification'list is a

conversion'or'editing'specifier

or a

specification'list θ , θ conversion'or'editing'specifier.

A conversion'or'editing:specifier is a

conversion'specifier

or

editing'specifier or iterator θ (θ specification'list θ).

Iterator is a number.

A conversion'specifier is one of the forms

W θ I θ d
W θ A θ d
W θ F θ d
W θ D θ d
W θ S
W θ B

W θ O
W θ X
W θ H
W θ H θ L

A fixed:field:list is either

fixed:field:element

or

fixed:field:list θ , θ fixed:field:element.

A free:field:list is either

free:field:element

or

free:field:list θ , θ free:field:element

A fixed:field:element is a named:variable. A free:field:element is either a named:variable, table:name, or array:name. Note that an array:name may be used in fixed:field:list as long as it is appropriately subscripted and that a table:name may not appear at all in a fixed:field:list.

Conversions are termed "free field" if they are specified by an encode: or decode: statement with a free:field:conversion:clause; otherwise they are termed "fixed field".

3.2.2.11.3.3 Fixed Field Conversions

"Field" means a substring of the buffer associated with an encode: or decode:statement.

Buffer:name is either the name of a literal:item, a table or array. If buffer:name is the name of a literal:item, that item must be not less than ψ bytes in length and must begin in byte 0 of a word. Further, if the structure called buffer:name is greater than one word in length, it must occupy consecutive words. When the structure called buffer:name is used in an encode: or decode:statement, it is treated as a string of hollerith characters, no matter how it may have been declared. (The structure called buffer:name is referred to subsequently as the buffer.)

A conversion:clause is either a fixed:field:conversion:clause or a free:field:conversion:clause.

A fixed:field:conversion:clause is either

(θ format:variable θ) θ fixed:field:list

or

(θ format:variable θ , θ service:call θ) θ fixed:field:list

A free:field:conversion:clause is either

(θ) θ free:field:list

or

(θ , θ service:call θ) free:field:list

An editing'specifier is one of the forms

\overline{W}
 $\overline{-} \theta \overline{W}$
 $\overline{W} \theta \overline{C}$
hollerith:constant

\overline{W} denotes a number, the value of which must be non-zero; \overline{d} denotes a number. The meaning of conversion'specifiers and editing'specifiers, and the role played by iterator will be explained in 3.2.2.11.3.3, 3.2.2.11.3.4, and 3.2.2.11.3.5.

3.2.2.11.3.2 The Encode:Statement and Decode:Statement

An encode:statement is

'ENCODE θ buffer:name θ conversion:clause θ \$

A decode:statement is

'DECODE θ buffer:name θ conversion:clause θ \$

3.2.2.11.3.3.1 Description of Conversion'specifiers

The number denoted by \overline{W} always represents the field width in terms of number of character positions. The number denoted by \overline{d} represents an implied negative power of ten when used in conjunction with a decode:statement; when used in conjunction with an encode:statement, \overline{d} denotes the number of digits to the right of the decimal point. The letter following \overline{W} in a conversion'specifier indicates the "conversion type".

L used in $\overline{W} \theta \overline{H} \theta \overline{L}$, means "left-justified". If $\overline{W} \theta \overline{H}$ is used, "right-justified" is implied. "Left-justified" and "right-justified" are defined in 3.2.2.11.3.3.2 and 3.2.2.11.3.3.3.

3.2.2.11.3.3.2 Conversion with a Decode:Statement

The execution of a decode:statement causes a field of the buffer to be converted from a hollerith representation to an internal representation which is then assigned to an associated named:variable. The form of the internal representation is defined by the item:declaration given for that named:variable which is associated with the conversion'specifier during the execution of the decode:statement.

The conversion of a hollerith representation to an internal representation takes place according to the same rules and is subject to the same restrictions as would apply if the conversion were effected by an assignment:statement of the form:

named:variable $\theta = \theta$ constant θ \$

where constant is simply the contents of the field being converted. However, it must be added that if the contents of the field form an integer:constant and the conversion type is A, F, or D, then the converted value obtained above is multiplied by 10^{-d} before assignment to the named:variable. Note that, with the exception of fields defined by conversion'specifier W θ H, leading spaces and trailing spaces in the field are not regarded as part of the datum. The character configurations which the field may take for any given conversion type are shown in paragraphs (a) - (f). The conversion is undefined if the character configuration of the field is not one of those specified as allowable in (a) - (f).

(a) Conversion Type I, A, F, or D

If the conversion type is I, A, F, or D, the allowable character configurations of the field being converted are:

optionally:signed:integer:constant
optionally:signed:fixed:constant
optionally:signed:floating:constant
optionally:signed:extended:precision:floating:constant
spaces (all characters)
hexadecimal:constant
octal:constant

If the character configuration is spaces, the field is treated as if the single digit 0 appeared.

(b) Conversion Type S

The allowable character configurations are:

status:constant

name

When name is used, the field is treated as if its character configuration were $\sqrt{\text{name}}$.

(c) Conversion Type B

The allowable character configurations are:

0
1
T
F

T and F are treated as if they were 1 and 0, respectively.

(d) Conversion Type O

The allowable character configurations are:

octal:constant
octal:digits

When octal:digits are used, the field is treated as if its contents were O (octal digits).

(e) Conversion Type X

The allowable character configurations are:

hexadecimal:constant
hexadecimal:digits

If the latter is used, the field is treated as if X (hexadecimal:digits) appeared.

(f) Conversion Type H

The allowable character configuration is any string of hollerith characters. If the string of hollerith characters, once stripped of any leading and trailing spaces, forms a hollerith:constant, the contents of the field are treated as if they were that hollerith:constant. Otherwise,

- If the field is defined as right-justified (W θ H), any trailing spaces are recognized as part of the datum.
- If the field is defined as left-justified (W θ H θ L), trailing spaces are not recognized as part of the data.
- The contents of the field are treated as if they were number H (string:of:hollerith:characters) where the value of number is the number of characters in the string of hollerith characters.

3.2.2.11.3.3.3 Conversion with an encode:statement

The execution of an encode:statement causes named:variables to be converted from their internal representations to hollerith character string representations which are inserted into the buffer. The internal representations are defined by the item:declarations given for the named:variables. The hollerith character string representations are defined by the parameters

of the conversion'specifiers as associated with the named:variables during the execution of the encode:statement. The number W of the conversion'specifier indicates the width of the field into which the hollerith character string produced by the conversion will be placed. The number d indicates the number of digits to the right of the decimal point in the generated character string. The conversion type describes the format of the generated character string. Associated with a given conversion type are a set of allowable item types. A named:variable being converted with respect to a given conversion type must admit to one of the allowable item types; if not, the effect of the conversion is undefined. Allowable item types and hollerith character string representation formats are described in paragraphs (a) - (h) below.

(a) Conversion Type I

Allowable item types of named:variables which may be converted with conversion type I are:

integer
fixed
floating
extended:precision:floating

The format of the hollerith character string generated by means of this conversion is:

S I . . . I

where S specifies sign and I . . . I specifies integer digits. If the item converted is signed and non-negative, or unsigned, S is a space; otherwise S is "-".

(b) Conversion Type A

Allowable item types which may be converted are:

integer
fixed
floating
extended:precision:floating

The format of the hollerith character string is

S I . . . I . F . . . F

S and I . . . I are as described above; F . . . F are the fractional digits. The number of fractional digits is equal to d.

(c) Conversion Type F

Allowable item types are:

integer
fixed
floating
extended:precision:floating

The format is

$$S \cdot F \dots F E^{\pm} e \dots e$$

S and F...F are as described above. $E^{\pm} e \dots e$ represents the exponent. The number of digits making up $e \dots e$ is implementation dependent.

(d) Conversion Type D

Allowable item types are:

integer
fixed
floating
extended:precision:floating

The format is

$$S \cdot F \dots F E^{\pm} e \dots e D$$

S, F...F and $e \dots e$ are as described above. The terminal letter D denotes extended:precision:floating.

(e) Conversion Types O and X

Any of the item types that are allowed. The format produced by type O is $O \dots O$, where O is an octal:digit. The format produced by type X is $X \dots X$, where X is a hexadecimal:digit.

(f) Conversion Type S

The only allowable item type is status. The format is status:constant:signs where status:constant:signs are those signs which appeared in any of the constants V (status:constant:signs) when the status:item was declared.

(g) Conversion Type B

The only allowable item type is boolean. The format is "T" or "F" according as the value of the boolean item is "true" or "false".

(h) Conversion Type H

The only allowable item types are literal. The format is $C \dots C$ where C denotes a hollerith character; the number of characters making up $C \dots C$ equals the length of the literal item converted.

3.2.2.11.3.4 Description of Editing'specifiers

An editing'specifier does not control conversion, but rather serves to describe to the format processor, PFORM, certain actions to take when processing the buffer when it is used with either an encode:statement or a decode:statement.

(a) The form W means "skip forward in the buffer over the next W character positions",

- (b) The form $- \theta W$ means "skip backward in the buffer over the next W character positions. If W is large enough to cause repositioning to the left of byte 0 of the buffer, the meaning is "skip back to byte 0 of the buffer".

When editing'specifier, hollerith:constant is used with an encode:statement, the meaning is "insert the character string contents of the constant into the buffer". When used with a decode:statement, the meaning is "skip forward in the buffer a number of character positions equal to the length of the constant". When the editing'specifier $W \theta C$ is used with an encode:statement, the meaning is "insert W spaces into the buffer"; when used with a decode:statement, the meaning is "skip forward in the buffer W character positions".

3.2.2.11.3.5 Description of Iterator

When iterator is used in the conversion'or'editing'specifier form

iterator θ (θ specification'list θ)

it has the effect of replicating the contents of the parenthesized specification'list that number of times specified by the value of iterator. Thus $3\theta(5A2, 3, 41)$ will produce the same result when processed by PFORM as $5A2, 3, 41, 5A2, 3, 41, 5A2, 3, 41$.

3.2.2.11.3.6 Execution of Encode: and Decode:Statements

The execution of encode: and decode:statements with either fixed:field: or free:field: conversion:clauses takes place under the supervision of the format processor PFORM.

3.2.2.11.3.6.1 Fixed:field Execution

This paragraph describes the execution of encode: and decode:statements with fixed:field: conversion:clauses. The action of PFORM causes three "pointers" to be manipulated.

The buffer pointer points to a character position in the buffer. The format pointer points to a conversion'or'editing'specifier in the format:variable. The list pointer points to a fixed:field:list:element in the fixed:field:list. The buffer is said to be exhausted if the buffer points to the right of the rightmost buffer character position. The fixed:field:list is said to be exhausted if the list pointer points to the right of the rightmost fixed:field:list:element. The format:variable is said to be exhausted if the format pointer points to the right of the rightmost conversion'or'editing'specifier. Note that in the following description, it is assumed that the format:variable data has the form specified by specification'list; if not, the resultant action is undefined. The sequence of events, which take place when an encode: or decode:statement with a fixed:field:conversion:clause is executed, is as follows:

- (a) PFORM sets the buffer pointer to point to byte 0 of the buffer, the format pointer to point to the leftmost conversion'or'editing specifier in the format' variable, and the list pointer to point to the leftmost fixed:field:list: element of the fixed:field:list.
- (b) If the conversion'or'editing'specifier specified by the format pointer is an editing'specifier, the action called out by it is performed (resetting the buffer point by the value W) and control is passed to step (e). Otherwise, control is passed to step (c).
- (c) If the conversion'or'editing'specifier is an iterated sub-expression (form: iterator θ (θ specification'list θ)), it is, in effect, replaced by its expansion as described in 3.2.2.11.3.5, and control is passed to step (b). (No pointer is altered) Otherwise control is passed to step (d).
- (d) The conversion'or'editing'specifier is a conversion'specifier. Any subscript associated with the fixed:field:list:element specified by the list pointer is evaluated and the conversion specified by the conversion'specifier is performed using that fixed:field:list:element. This causes the buffer pointer to be modified by the amount W. Then, both the list and format pointers are advanced one position to the "right" and control is passed to step (e).
- (e) If the format:variable is exhausted, PFORM returns control to the statement following the encode: or decode:statement. The same action is taken if the fixed:field:list is exhausted. If the buffer, fixed:field:list, and format: variable are not exhausted, control is passed to step (b). If the fixed:field:list and the format:variable are not exhausted, control is passed to step (f).
- (f) If a service:call has been supplied in the fixed:field:conversion:clause, it is operated. On completion of the service:call execution, PFORM sets the buffer pointer to point to byte 0 of the buffer and passes control to step (b).

It should be noted that the primary purposes of the service:call are to allow the programmer the facility for refilling the buffer (decode:statement) as when converting card reader records, or to output the buffer, as to a line printer (encode:statement).

3.2.2.11.3.6.2 Free:field Execution

This paragraph describes the execution of encode: and decode:statements with free:field:conversion:clauses.

3.2.2.11.3.6.2.1 Free:field Decode:statement Execution

The actions taken by PFORM in the execution of the decode:statement in the order shown are:

- (a) A "virtual" fixed:field:list is constructed by PFORM from the free:field:list. This construction takes place by moving left to right through the free:field:list

copying the named:variables where they are found and replacing each table:name and array:name with a list of subscripted item:names. In the case of a table:name, the item list is composed of sublists the number of which is governed by the current value of the table NENT. Each sublist is composed of the table:item:names, subscripted by a constant, in the same order that they appeared between the BEGIN and END in the table:declaration.

The subscripts of the table:item:names in a given sublist are the same value; subscripts for the table:item:names in different item sublists increase from 0 to $NENT(\text{table:name}) - 1$ as the sublists appear from left to right. (Each sublist corresponds to a table entry) Note that all item:names between the BEGIN and END brackets of the table:declaration will appear in the sublist for each entry, regardless of any overlying which may have been specified.

In the case of arrays, a list of subscripted array:names replaces the array:name; that list contains the same number of subscripted array:names as there are elements in the array. Subscripts of the array:names vary in such a way that the named:subscripted:array elements appear in the same left to right order that would occur if the constants of an array:declaration for the particular array were replaced by their correspondent subscripted array:names.

- (b) A virtual format:variable is constructed. This format:variable contains only conversion'specifiers; the number of conversion'specifiers equal the number of named:variables in the virtual fixed:field:list constructed in (a). The conversion type is generated on the basis of the item:descriptions of the named:variables. The parameter d is established as zero; hollerith conversion'specifiers are established as right-justified.
- (c) Now, in effect, PFORM substitutes the virtual format:variable into the free:field:conversion:clause of the decode:statement and replaces the free:field:list with the virtual fixed:field:list in such a manner as to transform the free:field:conversion:clause into a fixed:field:conversion:clause. Then, PFORM executes the transformed decode:statement exactly as it executes any decode:statement with a fixed:field:conversion:clause, except that the field width W is determined dynamically as PFORM sweeps the buffer from left to right. Each string of consecutive non-blank characters encountered in the buffer is taken as a field of length (W) which is the number of (non-blank) characters in the string.

3.2.2.11.3.6.2.2 Free:field Encode:statement Execution

The actions taken by PFORM in the execution of the encode:statement in the order shown are:

- (a) A virtual fixed:field:list is constructed in exactly the same manner as described in 3.2.2.11.3.6.2.1(a).

(b) A virtual format:variable is constructed on the basis of the item:declarations of the named:variables in the virtual fixed:field:list. The format:variable contains a number of conversion 'specifiers equal to the number of named:variables in the virtual fixed:field:list. For a given conversion type, the field width W (and the parameter d) are fixed but implementation dependent. In addition to conversion 'specifiers, editing 'specifiers are inserted into the virtual format:variable.

These editing 'specifiers are selected in such a way as to guarantee that:

- for named:variables in the free:field:list, the decode:statement will produce the variable:name followed by an equal:sign followed by the value
- for table:names appearing in the free:field:list, the decode:statement will produce, each in a different buffer image,
 - the table:name
 - the names of each table:item, as if in a column header
 - the value of each item for a given entry with the entry number
- for array:names appearing in the free:field:list, the decode:statement will produce, each in a different buffer image,
 - the array:name, plane number, and column count for each column in the plane
 - each row in the plane
 (Planes are repeated as often as required.)

(c) PFORM now substitutes the virtual format:variable into the free:field:conversion:clause of the encode:statement and replaces the free:field:list with the virtual fixed:field:list in such a manner as to transform the free:field:conversion:clause into a fixed:field:conversion:clause. Then PFORM executes the encode:statement exactly as it executes any encode:statement with a fixed:field:conversion:clause.

3.2.2.11.4 Example

The example in this section illustrates the various facilities provided by JIO and a hypothetical set of standardized application Specific Service Routines.

These Service Routines consist of a family of IOCS (Input/Output Control System) procedures and a family of IOBS (Input/Output Buffering System) procedures. The IOCS routines appear as service:calls in the file:declarations. They cause information to be moved between internal buffers and the io:lists in file manipulation statements. The IOCS family calls upon the IOBS family to transmit information between the buffers and tapes or disks by use of tape drives and disk drives. The following facts serve to clarify the example:

- There are 15 transmission devices in the hypothetical system. These are devices 1 through 15. Device 0 is the central processor itself. Devices 1 - 7 are tape drives. Devices 8 - 15 are disk drives.
- There are two files which are available to the programmer. These are SYSINP, the system input file, and SYSOUT, the system output file. SYSINP and SYSOUT are, respectively, permanently assigned to tapes 1 and 2.
- IOCS and IOBS make reference to five (global) tables. FPT (File Parameter Table) contains file parameters. FPTW is a like table with the same structure as FPT. FPTW is a "working" file parameter table. DFT is a device/file table. There is one entry for each device. In a given entry, the item DFA is "true" if the device whose number is the ordinal of the entry is assigned to a currently opened file. If DFA is true, DF contains the value of the 'FILENUM which was generated by PFILE for that file. File buffers are always maintained in a special (global) table called SYSBUF'TABLE.
- A buffer for a file is allocated in SYSBUF'TABLE by means of a system function called RESERVE'SYSBUF; the example assumes that SYSBUF'TABLE never runs out of space. (Note that the full computer word of this system takes 32 bits -- each SYSBUF'TABLE entry is one 32 bit word in length.) The table BFT (Buffer/File Table) relates the index of the first SYSBUF word of a file buffer to the 'FILENUM of the file; the ordinal of a BFT entry is, in fact, the 'FILENUM value.

The IOCS and IOBS Service Routines shown adhere to the following conventions:

- Both functional and representational records are fixed length. A representational record may be a block of one or more functional records, but the block must always contain the same number of functional records.
- A buffer for a file consists of the number of words in a representational record plus one word.
- The first word of the buffer is a buffer control word; its value is always the device status as determined by the device procedure STAT'CHECK.

- No multiple buffering is permitted; only one representational record at a time may be maintained in a file buffer.
- One and only one device is ever associated with a file. Only 15 files may ever be currently opened simultaneously.
- Reading always begins at the first record in the functional record set and proceeds from record to record in the order of increasing reference number; no backspacing is allowed. POS is always assigned values from within the IOCS routines.

Note that, for the purposes of simplicity, many important details have not been included in the example. It is for simplicity's sake also that the conventions above are used. Thus, this example is purposefully simplified and is not to be construed as a full-scale model for a set of Service Routines and is included on the following pages.

EXAMPLE

```
START $
"FILE PARAMETER TABLE"
  TABLE FPT V 100 S $
    BEGIN
      ITEM FILE NAME H 24 $ "NAME OF FILE"
      BEGIN H(SYSINP) H(SYSOUT) END
      ITEM FUNC'REC'LENGTH I 15 U $
        "NUMBER WORDS PER FUNC RECORD"
      BEGIN 20 30 END
        "20 WORDS = 80 CARD COLUMNS,
        30 WORDS = 120 CHARACTER PRINT LINE"
      ITEM FUNCS'PER'REP I 15 U $
        "NUMBER OF FUNC RECORDS IN
        REPRESENTATIONAL RECORD"
      BEGIN 1 1 END
        "SYSINP, SYSOUT ARE UNBLOCKED"
      ITEM BASE/NUM I 32 U $
        "IF STORAGE IS DISK, BASE'NUM = DISK
        ADDRESS OF ZERO-TH REP RECORD.
        IF STORAGE IS TAPE, BASE'NUM = FILE NUM ON TAPE"
      BEGIN 0 0 END
        "SYSINP/OUT ARE TAPE, EACH FILE NUMBER 0 ON THE REEL"
      ITEM DEVICE I 15 U $
        "NUMBER OF DEVICE ATTACHED"
      BEGIN 1 2 END
    END "FPT"
      "NOTE THAT THE PRESETS IN FPT ARE FOR SYSINP AND SYSOUT"
  ITEM NENT'FPT I 32 S P 2 $
  OVERLAY NENT'FPT = FPT $
  "ESTABLISH NENT (FPT) AS 2 TO ACCOUNT FOR PREDEFINED SYSINP,
  SYSOUT ENTRIES"
  "NOW DEFINE A 'LIKE' WORKING FPT"
  TABLE FPTW R 16 S L $
"DEVICE/FILE TABLE = ONE ENTRY PER DEVICE"
  TABLE DFT R 16 S $
    BEGIN
      ITEM DF I 32 S $
        "DF ($ DEVICE NUMBER $) = FILENUM OF ASSOCIATED FILE"
      ITEM DFA B $ "DFA ($ DEVICE NUMBER $)
        = TRUE IF DEVICE IS ASSIGNED TO A FILE, OTHERWISE FALSE
      BEGIN "PRESET DFA TO FALSE"
        0 0 0 0 0 0 0
        0 0 0 0 0 0 0
```

```

        END "PRESET"
    END "DFT"
"BUFFER/FILE TABLE -- ONE ENTRY PER CURRENTLY OPENED FILE"
    TABLE BFT R | 6 S $
        BEGIN
            ITEM BFC I 32 U $
                "BFC ($ 'FILENUM $) = INDEX OF FIRST SYSBUF
                WORD WHERE BUFFERS AND CONTROL WORDS FOR FILE
                'FILENUM BEGIN"
        END
"SYSTEM BUFFER AREA"
    TABLE SYSBUF' TABLE V ~ S $
        BEGIN
            ITEM SYSBUF I 32 S $
        END
"SYSINP AND SYSOUT ARE ASSUMED OPENED"
FILE AIRFIELD
    MODE 0    OPEN IOCSOPEN'INP (H(AIRFIELD))
              INPUT IOCSINP ( = AIRFIELD'STAT)
              SHUT IOCSSHUT'INP ( ) $
ITEM AIRFIELD'STAT I 32 S $
"THE FOLLOWING THREE ITEMS AND THE AF'PARAMS TABLE CONSTITUTE THE
AIRFIELD FILE RECORD STRUCTURE"
    ITEM AF'NAME H 8 $           "2 WORDS"
    ITEM AF'LAT F $             "1 WORD"
    ITEM AF'LONG F $           "1 WORD"
    TABLE AF'PARAMS V I 2 P $ "25 WORDS WITH NENT"
        BEGIN
            ITEM AF/PARAM'1 I 32 S $
            ITEM AF'PARAM'2 F $
        END
"FUNCTIONAL RECORD LENGTH IS 29 WORDS"
"DECLARE SYSINP BUFFER"
    ITEM SYSINP'BUF H 8 0 $
"NOW, BRING IN THE REPRESENTATION PARAMETERS FOR AIRFIELD FILE FROM
SYSINP. THESE PARAMS ARE FREE FIELD. THE GROUP OF THEM CONSTITUTES
AN FPT ENTRY. THE PARAMS ARE
    AIRFIELD
    29
    10
    3
    9
AIRFIELD IS THE HOLLERITH FILE NAME
29 IS THE WORDS PER FUNC REC
10 INDICATES THAT THE REPRESENTATIONAL RECORD IS A BLOC OF 10 FUNC RECORDS
3 MEANS THE FILE IS THE FOURTH FILE ON A TAPE REEL (0 - ORIGIN INDEXING)
9 MEANS TAPE DRIVE 9 IS THE DEVICE

```

"NOTE THAT IT IS A SIGNIFICANT FEATURE OF THE JSERVICE ROUTINES OF THIS EXAMPLE THAT FILE PARAMETERS MAY BE 'BOUND' AT EXECUTION TIME"

"BRING IN FIRST SYSINP RECORD"

GOTO BUF'FILLER \$

'DECODE SYSINP'BUF (, BUF'FILLER)

FILE NAME (\$2\$), FUNC'REC'LENGTH (\$2\$), FUNCS'PER'REP (\$2\$),

BASE'NUM (\$2\$), DEVICE (\$2\$) \$

CLOSE BUF'FILLER \$

BEGIN

"THIS CLOSE KEEPS THE BUFFER FILLED (FROM SYSINP) FOR
THE 'DECODE STATEMENT ABOVE"

INPUT SYSINP "FILE RECORD TO" SYSINP'BUF \$

END

"NOTE THAT FPT ENTRY 2 NOW CONTAINS THE AIRFIELD FILE PARAMS. IOCS WILL TRANSFER THESE PARAMS TO AN ENTRY IN FPTW WHEN THE FILE IS OPENED. IOCS AND IOBS WORK WITH THE PARAMS AS THEY APPEAR IN FPTW"

"OPEN THE FILE"

OPEN AIRFIELD MODE 0 \$

"READ FROM THE FILE"

INPUT AIRFIELD

AF'NAME, AF'LAT, AF'LONG, AF'PARAMS \$

"CHECK FOR EOF"

IF AIRFIELD'STAT EQ 3 \$

BEGIN "EOF CODE" END

IF AIRFIELD'STAT EQ 2 \$

BEGIN "BAD TRANSMISSION CODE" END

.

.

.

"THE FOLLOWING EXAMPLE ILLUSTRATES ANOTHER USAGE TO WHICH THE 'EXECUTE STATEMENT MAY BE PUT. THIS USAGE MONITORS ARITHMETIC OVERFLOWS"

'EXECUTE ENABLE (OVERFLOW) (\$0\$) OVF'1.\$

.

.

.

'EXECUTE DISABLE (OVERFLOW) (\$0\$) \$

"THE FIRST 'EXECUTE INFORMS THE CPU (DEVICE 0) TO BEGIN MONITORING FOR OVERFLOWS. THE SECOND 'EXECUTE CAUSES OVERFLOW MONITORING TO BE SUSPENDED. THE MONITORING IS PERFORMED FOR ALL STATEMENTS BETWEEN THE TWO 'EXECUTES. OVF'1 NAMES A CLOSE WHICH WILL BE INVOKED WHEN AN OVERFLOW OCCURS"

.

.

.

SHUT AIRFIELD \$

```

PROC IOCSOPEN'INP (NAME'OF'FILE) $ "OPEN INPUT SERVICE"
'ENTRANCE IOCSOPEN'OUT (NAME'OF'FILE) $ "OPEN INPUT SERVICE"
    ITEM NAME'OF'FILE H 24 $ "HOLLERITH FILE NAME"
'ENTRANCE IOCSINP (=FILESTAT) $ "INPUT SERVICE"
'ENTRANCE IOCSOUT ( ) $ "OUTPUT SERVICE"
'ENTRANCE IOCSINS ( ) $ "INSERT SERVICE"
'ENTRANCE IOCSDEL ( ) $ "DELETE SERVICE"
'ENTRANCE IOCSSHUT'INP $ "SHUT SERVICE FOR INPUT"
ITEM FILESTAT I 32 S $ "FILE STATUS FLAG"
BEGIN "IOCS COMPLEX"
"WORKING PARAMS"
ITEM FPT'INDEX I 32 U $
ITEM SYSIND I 32 S $
IOCSOPEN'INP.
    "LOOK UP NAME'OF'FILE IN FPT TO FIND INDEX OF ENTRY WITH NAME'OF'FILE"
    FOR I = ALL (FILENAME) $
    BEGIN
        IF FILENAME ($ I$) EQ NAME'OF'FILE $
            BEGIN
                FPT'INDEX = I $
                GOTO IOCSOPEN' I $
            END
        END
    END
    "NAME'OF'FILE NOT IN FPT -- ERROR CODE HERE"
    IOCSOPEN'I.
    "NOW MOVE FPT ENTRY AT FPT'INDEX TO FPTW ENTRY AT 'FILENUM FOR LATER
    EASE OF ACCESS"
        ENTRY (FPTW ($ 'FILENUM $) = ENTRY (FPT ($ FPT 'INDEX $)) $
    "NOW GO ASSIGN (AND POSITION) DEVICE AND RESERVE BUFFER SPACE
        IOBSOPEN'INP ( ) $
        RETURN $
    "END OF IOCSOPEN'INP!"
    IOCSINP. "CODE FOR INPUT SERVICE"
        "GET THE NEXT LOGICAL RECORD"
            IOBSINP ( = SYSIND) $
        "WAS THERE A TRANSMISSION ERROR"
        IF SYSIND EQ -1 $
            FILESTAT = SYSIND $
        "WAS THERE AN END OF FILE"
        IF SYSIND EQ -2 $
            FILESTAT = SYSIND $
        "TRANSMISSION WAS OK -- NOW MOVE BUFFER TO IO
        OPERAND (S) SPECIFIED IN INPUT STATEMENT"
        TBL (SYSBUF, SYSIND, FUNC'REC'LENGTHW($'FILENUM $) ) $
        RETURN $
    "END OF IOCSINP"

```

```

IOCSSHUT'INP. "ENTRANCE TO SHUT FILE"
  IF DEVICE W ($ ' FILENUM $) LS 8 $
    BEGIN "REWIND TAPE"
      WAIT'SHUT.
        'EXECUTE STAT'CHECK ( = TEMPI)
          ($ DEVICEW ($ 'FILENUM $) $)$
        IF TEMPI EQ 0 $ GOTO WAIT'SHUT $
        "INITIATE REWIND"
        'EXECUTE REWIND ( )
          ($ DEVICEW ($ ' FILENUM $ ) $
        END
"NOW UNRESERVE DEVICE"
  DFA ($ DEVICE ($ 'FILENUM $) $) = 0 $
  RETURN $
"NOTE --
  DEVICE PROCEDURE
    STAT'CHECK ( = STAT)
  IS PERFORMED INSTANTANEOUSLY
  STAT = 1      IF DEVICE READY WITH NO ERRORS
    = 2      IF DEVICE READY BUT LAST TRANSMISSION OPERATION
              RESULTED IN ERROR
    = 3      IF DEVICE READY BUT LAST TRANSMISSION WAS A
              READ WHICH ENCOUNTERED AN END OF FILE MARK
    = 0      IF DEVICE NOT READY
  STAT TAKES ONLY 0 AND 1 AS VALUES IF THE LAST OPERATION ON THE
  DEVICE WAS NOT A TRANSMISSION (e.g., REWIND)
PROC IOBSOPEN'INP ( ) "ASSIGN DEVICE AND POSITION"
  'ENTRANCE IOBSINP ( = SYSBUF'INDEX) $
  ITEM SYSBUF'INDEX I 32 S $
  BEGIN "IOBS COMPLEX"
    ITEM TEMP1 I 32 U $
    ITEM TEMP2 I 32 U $
"CODE FOR IOBS'OPEN'INP FOLLOWS"
"CHECK TO SEE IF DEVICE HAS ALREADY BEEN ASSOCIATED WITH A FILE"
IF DFA ($ DEVICEW ($'FILENUM$) $ ) $
  BEGIN "IN USE"
    "ERROR CODE HERE"
  END
"NOW SET DFA TO SHOW DEVICE IS ASSIGNED TO FILE. ALSO SET DF TO VALUE
OF FILENUM OF ASSOCIATED FILE"
DFA ($ DEVICEW ($ 'FILENUM $) $) = 1 $
  DF ($ DEVICEW ($ 'FILENUM $) $) = 'FILENUM $
"NOW FIGURE OUT HOW MUCH BUFFER SPACE TO GET"

```

```

TEMP 1 =      FUNC'REC'LENGTH W ($ 'FILENUM $) *
              FUNCS'PER'REPW ($ 'FILENUM $) + 1 $
"GET INDEX IN SYSBUF OF FIRST WORD OF BUFFER COMPLEX FOR THIS FILE"
TEMPI = RESERVE'SYSBUF (TEMPI) $
"NOTE RESERVE'SYSBUF IS A SYSTEM FUNCTION DESIGNED TO ALLOCATE
SPACE, AS SPECIFIED BY AN INPUT PARAM, IN SYSBUF. THE VALUE OF THIS
FUNCTION AFTER EXECUTION IS THE SYSBUF FIRST WORD INDEX"
"SET THIS INDEX UP IN BFT"
      BFC($ 'FILENUM $) = TEMPI $
"NOW DO DEVICE POSITIONING. IF TAPE, REWIND, IF DISK, DO NOTHING.
FIRST, WAIT FOR UNIT TO COME READY"
WAIT'0.
'EXECUTE STAT'CHECK (= TEMPI)
      ($ DEVICEW ($ 'FILENUM $) $) $
      "WAIT LOOP"
      IF TEMPI EQ 0 $
          GOTO WAIT'0 $
IF DEVICEW ($ 'FILENUM $) LS 8 $ "TAPE OR DISK"
      BEGIN "TAPE, INITIATE REWIND"
          'EXECUTE REWIND ( ) ($ DEVICEW ($ 'FILENUM $) $) $
          "BEGIN WAIT LOOP"
          WAIT'1. "STAT'CHECK IS INSTANTANEOUS"
          'EXECUTE STAT'CHECK (= TEMPI)
              ($ DEVICEW ($ 'FILENUM $) $) $
          IF TEMPI EQ 0 $ GOTO WAIT'1 $
          "END REWIND WHEN TEMPI EQ 0"
          "NOW INITIATE TAPE POSITIONING"
          'EXECUTE
              SKIP'FILES'FORWARD (BASE'NUM ($ 'FILENUM $))
                  ($ DEVICEW ($ 'FILENUM $) $) $
          WAIT '2. "WAIT TILL DONE -- NOOP IF BASENUM = 0"
          'EXECUTE STAT'CHECK (= TEMPI)
              ($ DEVICEW ($ 'FILENUM $) $) $
          IF TEMPI EQ 0 $ GOTO WAIT'2 $
          RETURN $ "END OF SKIP'FILE'FORWARD"
      END
"NOW START INPUT INTO BUFFER"
IOBS READ ('FILENUM, BASE'NUMW ($ 'FILENUM $) ) $
"NOTE BASE'NUMW ABOVE WILL NOT BE USED BY IOBSREAD IF DEVICE IS TAPE"
RETURN $ "END IOBSOPEN' INP"
IOBSINP. "ENTRANCE TO IOBS FOR 'INPUT' SERVICE CALL"
      TEMP2 = REM (POS, FUNCS'PER'REPW ($ 'FILENUM $) ) $
"TEMP 2 NOW CONTAINS THE ORDINAL OF THE FUNCTIONAL RECORD IN
THE BUFFER"

```

```

IF TEMP2 NQ 1$
  BEGIN "A READ WAS INITIATED PREVIOUSLY"
    WAIT'INP.
      IF SYSBUF ($ BFC ($ 'FILENUM $) $) EQ 0 $
        GOTO WAIT'INP $ "BUFFER IS STILL FILLING"
      "WHEN DONE, TEST FOR EOF"
      IF SYSBUF ($ BFC ($ ' FILENUM $) $) EQ 3 $
        BEGIN
          "CODE TO PROCESS END OF FILE"
        END
      "NOW TEST FOR TRANSMISSION ERROR "
      IF SYSBUF ($ BFC ($ ' FILENUM $) $) EQ 2 $
        BEGIN
          "CODE TO PROCESS ERROR"
        END
      END "BUFFER IS FILLED"
    "NOW COMPUTE BUFFER ADR"
    SYSBUF'INDEX = BCF ($ 'FILENUM $) + FUNC'REC'LENGTHW ($ 'FILENUM $) *
      TEMP2 + 1 $
    "INCREMENT POS"
      POS = POS + 1 $
    "NOW TEST TO SEE IF BUFFER NEEDS FILLING"
    TEMP 2 = REM (POS, FUNCS'PER'REPW ( $ 'FILENUM $)) $
    IF REM EQ 0 $
      BEGIN
        "IF DISK, COMPUTE NEW DISK ADDRESS AND STORE IN TEMP 2"
        IOBS READ ('FILENUM, TEMP2 ) $
      END
    "RETURN -- ALL DONE"
  END "IOBSINP"
  "READ COMPLETION ROUTINE"
  PROC ENDREAD ( ) $
    ITEM TEMP I 32 S $
  BEGIN
    "FIND OUT WHICH DEVICE CAUSED INTERRUPT"
    'EXECUTE WHICH ( = TEMP) ($ 0 $)
    "DEVICE PROCEDURE 'WHICH' IS INSTANTANEOUS. IT ASKS THE
    CPU 'DEVICE' WHICH OTHER DEVICE CAUSED THE LAST INTERRUPT.
    THE ANSWER IS PLACED IN TEMP"
    "NOW DO STATUS CHECK -- PLACE STATUS IN BUFFER CONTROL WORD"
    'EXECUTE STAT'CHECK
      ( = SYSBUF ($ BFC ($ DF ($ TEMP $) $) $) )
      ($ TEMP $)
    "RETURN"
  END

```

```

PROC IOBSREAD (NFILE, DISKADR)
  ITEM NFILE I 32 S $
  ITEM DISKADR I 32 U $
BEGIN
  ITEM TEMPI I 32 S $
  "FIRST SET BUFFER CONTROL WORD TO 0 -- BUFFER WILL BE FILLING WHEN WE
  EXIT THIS PROC"
  SYSBUF ($ BFC ($ NFILE $) $) = 0 $
  "IS DEVICE DISK OR TAPE"
  IF DEW ($ NFILE $) LS 8 $ "TAPE"
    GOTO TAPE'READ.
  DISK'READ. "WAIT FOR DISK READY"
    'EXECUTE STAT'CHECK ( = TEMPI)
      ($ DEW ($ NFILE$) $) $
  IF TEMPI EQ 0 $ GOTO DISK'READ $
  "NOW INITIATE DISK READ -- NOTE THAT THE DEVICE PROCEDURE
  READ DISK IS NON-INSTANTANEOUS"
  'EXECUTE READDISK ( DISKADR, "DISK ADDRESS"
    "NUM WORDS TO READ" FUNC'REC'LENGTHW ($NFILE$)*
      FUNCS'PER'REPW ($NFILE$),
    "CORE LOCATION OF BUFFER" 'LOC ( SYSBUF) + BFC ($ NFILE $) + 1 ) $
    "DEVICE" ($ DEW ($ NFILE$) $)
  "COMPLETION ROUTINE" ENDREAD ( ) $
  RETURN $
  TAPE'READ. "WAIT FOR TAPE READY"
    'EXECUTE STAT'CHECK ( = TEMPI)
      ($ DEW ($ NFILE$) $) $
  IF TEMPI EQ 0 $ GOTO TAPE'READ $
  "NOW INITIATE TAPE READ -- DEVICE PROCEDURE IS NON-INSTANTANEOUS"
  'EXECUTE READTAPE ( FUNC'REC'LENGTHW ($NFILE$)*
    "NUM WORDS TO READ" FUNCS'PER'REP ($ NFILE$),
    "CORE BUFFER ADDRESS" 'LOC (SYSBUF) + BFC ($NFILE$) + 1 ) $
    "DEVICE" ($ DEW ($ NFILE$) $)
    "COMPLETION ROUTINE" ENDREAD ( ) $
  "RETURN"
END "IOBSREAD"

```


3.2.2.12 The Stop:statement

The stop:statement has the form:

STOP θ \$

A stop:statement has the effect of terminating the sequence of executions initiated when the main:program began executing. This is true whether or not the stop:statement is executed in the main:program or in a closed subroutine subordinate to the main:program. The stop:statement in a closed subroutine does not cause control to pass to the calling program.

3.2.2.13 The Pause:statement

A pause:statement is:

PAUSE θ statement:name θ
optional:service:call θ \$

When a pause:statement is executed, the service:call, if included, is operated. Upon completion of the service:call execution, the sequence of executions initiated when the main:program began executing is suspended. This is true whether or not the pause:statement was executed in the main:program or in one of the closed subroutines subordinate to it. When execution is restarted by some external means, the first statement executed is the one labelled by statement:name. Note that the execution of a pause:statement which has no service:call takes place in the same manner as above except, of course, that there is no service:call operation. It is anticipated that a common use of the service:call will be to output some sort of identification information (e.g. to the printer).

3.2.2.14 Rounded:assignment:statements

A rounded:assignment:statement has the form:

numeric:variable $\theta = \theta$
numeric:formula θ 'ROUNDED θ \$

The rounded:assignment:statement produces exactly the same result when the evaluated numeric:formula is assigned to the numeric:variable as would be obtained if the numeric:variable were declared in an item:declaration with the abbreviation R. (ROUNDED has no effect if indeed numeric:variable was declared in an item:declaration with the abbreviation R.)

3.2.2.15 Extension to Program

Program means:

```
START  $\theta$  optional:origin  $\theta$  $  
       $\theta$  statement:list  $\theta$   
      TERM  $\theta$  optional:statement:name  $\theta$  $
```

or:

```
CLOSE  $\theta$  name  $\theta$   
      START  $\theta$  optional:origin  $\theta$  $  
      statement:list  $\theta$  TERM  $\theta$  $
```

or:

```
START optional:origin  $\theta$  procedure:declaration  $\theta$   
      TERM  $\theta$  $
```

The meaning of the first two forms is given in CED 2487.1. The third form provides the facility for compiling "stand-alone" procedures. The effect of optional:origin is the same as for the first two forms. The program defined by the third form has the same name as the procedure:name in the procedure:declaration. The program may also be referenced by the alternate:entrance:name which appears in any alternate:procedure:entrance:declaration. A program defined by the third form is invoked precisely as any procedure. Also, the procedure:name and any alternate:entrance:names are category 3 names.

3.2.3 Nucleus Features

Based on the criteria as described in Section 1 (1.3) and the users responses to the JOVIAL Application Questionnaire and Interview questions (contained in Appendix 3) as summarized in Appendix 2 and 4, respectively, the following features are recommended for inclusion in the nucleus. The references are to the appropriate sub-sections of the JOVIAL Application Questionnaire (JAQ), Section 3 of this report (Section 3), Interview Response Analysis in Appendix 4 (1), and the Approach for Change (AFC). Specification of each feature is contained in the JOVIAL Application Questionnaire for the current JOVIAL features considered and in Section 3 for recommended extentions. Justification for inclusion of features as a nucleus is contained in Appendix 2 (which is organized by AFC numbers) and Appendix 4 (which is organized by 1 numbers).

<u>Feature</u>	<u>JAQ/Sec. 3</u>	<u>AFC/I</u>
1) Item Type Hollerith	J3.5.1.1.11.1	A1.1.1.1
2) Hollerith Formula	J3.5.2.5	A1.1.1.1
3) Character Data Size Attribute	J3.5.1.1.14	A1.1.1.3
4) Operations on Literal Data	3.2.2.4	I 3, I 7
5) Explicit Size Attribute	J3.5.1.1.14	A1.1.3
6) Structure Table	J3.5.1.3.1	A1.2.2
7) Table Variability Attribute - Rigid	J3.5.1.3.4	A1.2.3
8) Table Variability Attribute - Variable	J3.5.1.3.4	A1.2.3
9) NENT	J3.5.2.2.2.6	A1.2.3
10) BYTE	J3.5.2.2.2.2	A2.2
11) Subscripting - Expressed as Complex Numeric Formula	J3.5.2.2.1	A2.4
12) Subscripting - Nested	J3.5.2.2.1	A2.4
13) Table Entry Referencing (ENTRY only may be used)	J3.5.2.6	A2.5
14) Computer Representation of Data	J3.5.1.1.15	A3
15) Computer Representation of Numeric Items and Constants	3.2.2.5	I 11
16) Octal Constants, Hexadecimal Constant (Must implement one or the other, may implement both)	J3.5.1.1.1 3.2.2.1	A3.1, I 12
17) Basic Table Structure Attribute-Parallel	J3.5.1.3.5	A3.3
18) Basic Table Structure Attribute - Serial	J3.5.1.3.5	A3.3
19) Defined (Table) Packing	J3.5.1.3.6.2	A3.5
20) Independent Overlay Declaration	J3.5.1.7.1	A3.6
21) 'LOC	J3.5.2.1.1	A3.8
22) Parenthesized Numeric Formulas	J3.5.2.3.1	A4
23) Mixed Item Types in Numeric Formulas	J3.5.2.3.5	A4
24) Item Type Integer	J3.5.1.1.4	A4
25) Item Type Fixed Point	J3.5.1.1.5	A4
26) Item Type Fixed Point - Scale in Declaration Blank or Zero	J3.5.1.1.5	A4
27) Item Type Floating Point	J3.5.1.1.9	A4
28) Sign Attribute	J3.5.1.1.6	A4.4
29) Prefix + and -	J3.5.2.3.2	A4.5
30) Precedence of Unary Operators (Alteration of precedence of +, -)	3.2.26	I 17
31) Exponentiation Operator (Notation**only)	J3.5.2.3.3	A4.5
32) Relational Formulas (excluding chained)	J3.5.2.7	A6.1
33) Boolean Formulas - with AND, OR, NOT	J3.5.2.8	A7.1
34) Item Type Boolean	J3.5.1.1.12	A7.2
35) Numeric Item Type Conversion in the Assignment Statement	J3.5.3.1	A8

<u>Feature</u>	<u>JAQ/Sec. 3</u>	<u>AFC/I</u>
36) Boolean Assignment Statement	J3.5.3.1	A8
37) Item Preset	J3.5.1.6.1	A8.1
38) Table Preset	J3.5.1.6.2	A8.1
39) Compound Statement	J3.5.4.2	A9.1
40) GOTO Statement	J3.5.5	A9.2
41) STOP Statement	3.2.2.12	A9.3, I 24
42) IF Statement	J3.5.5.3	A9.4.1
43) FOR-loop-One Factor	J3.5.5.5	A9.5
44) FOR-loop-Two Factor	J3.5.5.5	A9.5
45) FOR-loop-Three Factor	J3.5.5.5	A9.5
46) FOR-loop-Decrementing	J3.5.5.5	A9.5
47) Test Statement	J3.5.5.5.6	A9.5.4
48) Program	J3.5.4.10	A9.6, A9.7
49) Procedure (Excluding Function)	J3.5.4.6	A9.6.1
50) Function	J3.5.4.8	A9.6.1
51) Alternate Entrances to Procedures and Functions	3.2.2.9	I 28
52) Close	J3.5.4.5	A9.6.2
53) Return Statement	J3.5.5.6	A9.6.3
54) Direct Code	J3.5.4.4	A9.8
55) Data Editing and Conversion	3.2.2.11.3	I 31
56) Multiple Statement Labels	J3.5.4.3	A11.2
57) ALL	J3.5.5.5.5	A12.5
58) COMPOOL	J3.5.6.1	A12.6

3.2.4 Optional Features

Based on the same criteria and data as indicated in Section 3.2.3 above, the following features are recommended for the optional category.

<u>Feature</u>	<u>JAQ/Sec. 3</u>	<u>AFC/I</u>
1) Hexidecimal Constants	3.2.2.1	I 12
2) Simplified Hollerith Constant Form	3.2.2.2	-
3) User Definable Character Encoding Scheme ('CHARCODE)	3.2.2.3	I 2
4) Item Type Status	J3.5.1.1.13	A1.1.2
5) Status Formulas (Must be implemented together)	J3.5.2.5	A1.1.2
6) Explicit Status Size Attribute	J3.5.1.1.14	A1.1.3
7) Structure Array	J3.5.1.2	A1.2.1
8) NWDSN	J3.5.2.1.2	A1.2.4
9) BIT	J3.5.2.2.2.1	A2.1

<u>Feature</u>	<u>JAQ/Sec. 3</u>	<u>AFC/I</u>
10) Ordinary Packing (Dense only is allowed)	J3.5.1.3.6.1	A3.4
11) Absolute Value Operator (ABS only is allowed)	J3.5.2.3.4	A4.7
12) Extended Precision Numeric Items and Constants	3.2.2.10	I 16
13) Chained Relational Formulas	J3.5.2.7	A6.2
14) Array Preset	J3.5.1.6.3	A8.1
15) Exchange Statement	J3.5.3.2	A8.3
16) IFEITH/ORIF Statements	J3.5.5.4	A9.4.2
17) Index Switch	J3.5.5.2	A9.4.3
18) Item Switch	J3.5.5.1	A9.4.4
19) Close in Parameter List (of Procedure)	J3.5.4.6	A9.6.1
20) Statement Label in Parameter List (of Procedure)	J3.5.4.6	A9.6.1
21) 'PROGRAM Declaration	J3.5.4.11	A9.7
22) Item Declaration	J3.5.1.1.2.2	A11.1
23) Define Directive	J3.5.6.2	A12.1
24) Extended Define Directive	3.2.2.7	I 35
25) MODE Directive	J3.5.1.1.3	A12.2
26) Extended Mode Directive	3.2.2.8	A12.2, I 36
27) Like Table Declaration	J3.5.1.3.2	A12.4
28) Device oriented Input/Output Module	3.2.2.11.1	I 33
29) Functional File Input/Output Module	3.2.2.11.2	I 32
30) Pause Statement	3.2.2.13	I 25
31) Extension to Program ("Stand-alone" Procedure)	3.2.2.15	-

SECTION IV

ADDITIONAL RECOMMENDATIONS

4.1 Specific Study Topics

During the course of the interviews several unplanned for auxiliary topics of unusual interest were discussed. Some of these subjects were concerned with general language aspects, others with specific required features, and still others with implementation constraints. Following is a list of the more important concepts raised in the discussions, the details of which are included in Appendix 1. DDI strongly recommends in-depth studies of all of the concepts for possible inclusion in future JOVIAL language extensions.

4.2 General Concepts

1. JOVIAL Input/Output - Many comments were made indicating that some JOVIAL input/output specification should be included in the language. DDI has included a set input/output specifications in the Extension Specifications, however, further effort is required to increase the comprehensiveness of these specifications.
2. JOVIAL Library - It was recommended that a standard JOVIAL Source Language Library should be generated and available to all users.
3. JOVIAL COMPOOL - It was proposed that the COMPOOL be rigidly specified in the language rather than permitting individual interpretations of the concept by the implementors of the JOVIAL language.
4. Real Time Concepts - It was suggested that statements facilitating real time applications should be included in the language.
5. List Processing - It was proposed that some of the notions associated with the concepts of list processing should be included in this language.

4.3 Specific Concepts

The following specific functions were recommended for inclusion in the JOVIAL language:

1. Bit string items with associated bit string operations.
2. Expanded data structuring and associated data referencing concepts.
3. Loader and Compiler Directives.
4. Implied subjects and objects in IF statements.
5. Expanded character set.
6. Multiple assignment statement.
7. Non-linear loop incrementing.

4.4 Language/Compiler Design Goals

During discussions the following general language and compiler design goals were considered worthy of further investigation:

1. Attainment of absolute machine independence.
2. Improved readability of JOVIAL computer programs.
3. Less verbose.
4. Separate function from functional software.
5. Compatibility with manufacturer software.

APPENDIX I

INTERVIEW MATERIAL

This appendix contains a list of the agencies and firms who participated in the study together with information regarding their JOVIAL Application Questionnaire and interview responses. DDI interviewed ten military agencies, two industrial firms, and one non-military governmental agency in this study. Following is a list of those organizations interviewed together with associated abbreviations that are used to identify them throughout this and associated documents.

1. ADC Computer Programming and Analysis Center
Ent Air Force Base
Colorado Springs, Colorado -ADPAC
2. Computer Sciences Corporation
650 N. Sepulveda Boulevard
El Segundo, California -CSC
3. Federal Aviation Agency
800 Independence Avenue, S.W.
Washington, D.C. -FAA
4. FOCCPAC
U.S. Navy
Honolulu, Hawaii -FOCC
5. Operational Program Maintenance Branch
4th Air Force Combat Center
Hamilton Air Force Base
San Francisco, California -HAM
6. NAVCOSSACT
Washington Naval Yard Annex
Washington, D.C. -NAVC
7. National Civilian Defense Computer Facility
Olney, Maryland -NCD CF
8. Naval Electronics Laboratory
Command and Control Facility
San Diego, California -NEL

9. Defense Communication Agency
Naval Military Command System Support Center
Pentagon
Washington, D. C. -NMCSSC
10. Headquarters NORAD
NHCP
Ent Air Force Base
Colorado Springs, Colorado -NHCP
11. Rome Air Development Center
Griffiss Air Force Base
Rome, New York -RADC
12. Headquarters SAC
Offutt Air Force Base
Nebraska -SAC
13. Trans World Airways Data Processing Center
King Road
Rockleigh, New Jersey -TWA

The selected organizations responded to the JOVIAL Application Questionnaire (JAQ) in varying degrees of detail. Some responded qualitatively, some quantitatively, and some not at all. Following is a summary of the responses to the JAQ by the selected respondents:

1. ADPAC- Complete quantitative response to three JAQ's.
2. CSC- No JAQ usage rates because of CSC's role as a JOVIAL compiler writer rather than a JOVIAL user.
3. FAA- No JAQ usage rates were given; only qualitative results were provided.
4. FOCC- Complete quantitative responses to three JAQ's.
5. HAM- Complete quantitative responses to one JAQ.
6. NAVC- Complete quantitative responses to three JAQ's.
7. NCDCF- Complete quantitative responses to one JAQ.
8. NEL- Complete quantitative responses to one JAQ.
9. NMCSSC- Complete quantitative responses to one JAQ.
10. NHCP- Responses received too late to be considered for analysis.
11. RADC- Complete quantitative information was returned on three of the four JAQ's.
12. SAC- Complete quantitative information was provided on four JAQ's.
13. TWA- Because of a tight schedule, TWA did not complete their JAQ.

Each of the numbered sections that follow contain information summarizing the results obtained from each respondent. A complete description of the respondent is given, his response to the JAQ is noted, any written interview material is included and notes are provided that reflect general impressions gathered from the questionnaire and interview. The respondents are listed in each section according to the sequence indicated above. Each of the sections containing data from the respondent is organized according to the following outline:

- | | |
|-------------------------------|---|
| 1. Organization - | Particular application for specific agency. |
| 2. Contact - | Organization representative. |
| 3. Organization Identifier - | Organization abbreviation. |
| 4. Number of JAQ's returned - | Self Explanatory. |
| 5. Date of Interview - | Self Explanatory. |
| 6. JAQ Response Notes - | Comments about the response to the JAQ. |
| 7. Application - | Information about each application responded to in a JAQ. |
| 8. Interview Notes - | Statements of actual interview responses or interpretations of interview responses if required. |

1.0 ADPAC

1.1 Organization: ADC Computer Programming and Analysis Center
ADPAC
Ent Air Force Base
Colorado Springs, Colorado

1.2 Contact: Lt. Jon Feller

1.3 Organization Identifier: ADPAC

1.4 Number of JAQ's Returned: 4

1.5 Date of Interview: 14 May 1968

1.6 JAQ Response Notes:

Three of the JAQ's returned contain response data for three distinct sub-applications. These are identified by ADPAC-1, ADPAC-2, ADPAC-3. There is one compiler for all sub-applications. Compliance information was taken from the JAQ for ADPAC-3.

1.7 Application:

(1) Sub-application ADPAC-1

Sub-application Name: Special Programs Section

Sub-application Description:

Data reduction and analysis for radar evaluation and SNOWTIME. The sub-application includes editing, statistical calculation and data extraction. Sub-application currently consists of 7 computer programs with more expected to be started in the near future. Some computer programs were converted from COBOL.

Total Number of JOVIAL Statements: 5462

Sub-application Hardware Configuration: See under ADPAC-2.

Operating System: No information given.

Background:

- JOVIAL was used because it was the language best suited for the project.
- Average years experience of programmers: 3
- Average years JOVIAL experience: 1
- Dialect used: JOVIAL J3
- Compilation speed: Not given.
- Compiled code: Not relocatable.

(2) Sub-application ADPAC-2

Sub-application Name: SPACE TRACK

Sub-application description:

The objective of the sub-application is to maintain surveillance of all earth orbiting objects.

Total Number of JOVIAL Statements: 1400 (Approx.)

Computer manufacturer and model Philco 2000/212

High speed memory storage	32K words
Size	256K six bit bytes
Word size if applicable	48 bits

Input/Output devices

Drum	
Manufacturer and model	Bryant
Storage capacity	10,000,000 ₈ six bit bytes
Quantity on this configuration	2

Disc		
Manufacturer and model		0
Storage capacity		
Quantity on this configuration		
Magnetic tapes		
Manufacturer and model		Ampex
Recording densities permitted		
Quantity on this configuration		14
Card punches		
Manufacturer and model		0
Card punch speed		
Quantity on this configuration of this type		
Card reader		
Manufacturer and model		IBM
Card read speed		
Quantity on this configuration		
Printer		
Manufacturer and model		Philco
Printer speed		1,000 lines/min
Quantity on this configuration		1
Paper tape reader		
Manufacturer and model		0
Tape read speed		
Quantity on this configuration		
Paper tape punch		
Manufacturer and model		0
Tape punch speed		
Quantity on this configuration		

Operating system:

Name: DELTA

Developer: SDC

Mode of operating system: Batch Processing and Real Time

Background:

- JOVIAL was used because it was the official Air Force language.
- Average years experience of programmers: SDC - 5, Military - 3
- Average years JOVIAL experience: SDC - 5, Military - 1

Dialect used: JOVIAL J3

Compilation speed: Not given.

Compiled Code: "Applications are all RPL".

(3) Sub-application ADPAC-3

Sub-application Name: Satellite Archives - Storage and Retrieval

Sub-application Description:

The purpose of the sub-application is to store and retrieve satellite observations and element sets. The current Archives system programs are for the most part written in TAC. Only one short program has been written in JOVIAL but plans for the future involve rewriting of all programs in JOVIAL.

Total Number of JOVIAL Statements: 278

Sub-application Hardware Configuration: See under ADPAC-2

Operating System:

Name: COSMOS
Developer: SDC
Mode of operation: Batch processing

Background:

- JOVIAL was used to stay compatible with other projects and because it was the official Air Force language.
- Average years experience of programmers: 6
- Average years JOVIAL experience: Not given
- Dialect used: J3
- Compilation speed: Not given
- Compiled code: Not given

1.8 Interview Notes:

The following written information was received at the time of the interview.

General:

- We have three areas of application. The largest, 80% of our JOVIAL programs is SPADATS(496L). The Special Projects Section has 15%. The smallest is Archives with 5%.

- Technical data on the status of our compiler is contained in the Archives booklet. A description of JOVIAL library routines and diagnostic messages is included. Also included is a sample output and user instructions for the utility routine REFORM.
- Conclusions: Judging from our usage of JOVIAL two subsets would seem to be needed. One an extension of AFM 100-24 JOVIAL to facilitate scientific data analysis; the second a limited version without many of the features of AFM 100-24 for use in specific applications within real time systems which require extremely efficient code and minimum core utilization. It would also seem desirable to include specifications and minimum number of routines for the JOVIAL library.

HQ ADC, ADPAC, Special Programs Section

- JOVIAL programming in the Special Programs Section started in August 1967. The first implementation was the Radar Evaluation System. A limited version had been implemented in COBOL. It was desired to include more extensive calculations, which could not be done in COBOL. Both FORTRAN and JOVIAL were considered. Because of the greater power and flexibility of JOVIAL it was selected.
- The JOVIAL features which were significant in this decision were:
 - The ability to move and use data by BYTE modification.
 - The ability to pack data in core tables and in tape records.
 - The ability to embed arithmetic operations in subscripting and in operations, such as, $BYTE(\$10 + A*3\$)$ where A is a loop variable.
 - The ability to overlay data.
 - The define statement.
 - The ability to do manual conversion of input data enabling the same data fields to be used for both numeric and alphabetic data.
- Further JOVIAL programs are being written for the SNOWTIME data analysis series. Plans call for placing the ADPAC status report in JOVIAL.
- The following changes were proposed for JOVIAL:
 - Add carriage control option for printed output.
 - Expand features for conversion of numeric input/output to include signs and decimal points.

Greater detail in diagnostics. Too many merely state that there is an error without specifying the exact problems.

HQ ADC, ADPAC SPACE TRACK 496L

Computer programs are mostly in TAC (Philco 2000 assembly language) at the present. As major changes are made to the computer programs they are being written in JOVIAL. The following changes to JOVIAL were indicated to be highly desirable:

- The capability to overlay COMPOOL defined items and tables with program defined items and tables.
- For integer by integer division the fractional bits should be preserved according to the resulting defined integer.
- JOVIAL I/O should include the option of having automatic conversion of numeric items, like that in FORTRAN.
- AFM 100-24 should include specifications for a standard JOVIAL library.
- There should be a JOVIAL subset which is more limited than that of AFM 100-24. The purpose being to produce extremely efficient code, both core size and execution time. This is needed for real time systems.
- A traceback routine should be included for program error halts to aid in determining the cause.
- GEN2 diagnostics should give the line number.
- When possible diagnostics should indicate the specific column which contains the error.

Other Comments:

In addition to the written information, ADPAC personnel made the following statements:

- It would be desirable to have the facility to apply the BYTE functional modifier to an operand designated by the ENTRY functional modifier:

BYTE (\$3,5\$) (ENTRY(TABLE'NAME,3))

- The NENT word should be located in some place other than word 0 of the table.
- One ought to be able to use parentheses instead of the subscript brackets (\$ and \$).

- It is felt that more should be said in AFM 100-24 about COMPOOL and Libraries.
- A Hollerith constant form. - H(characters) ought to be provided.
- A facility for specifying assignment without automatic conversion should be supplied.
- The FOR loop feature should be set up in such a way as to provide for testing prior to the first iteration.
- "Stringed" item declarations are desired:

ITEM AA, BB, CC | 48 S \$
- Arrays with rows and column lengths and dimensions variable at run time would be helpful.

2.0 CSC

- 2.1 Organization: Computer Sciences Corporation
 650 N. Sepulveda
 El Segundo, California
- 2.2 Contact: Mr. Terry Dunbar
- 2.3 Organization Identifier: CSC
- 2.4 Number of JAQ's Returned: 1
- 2.5 Date of Interview: 11 July 1968
- 2.6 JAQ Response Notes:

The JAQ returned contained no usage notes. This is because, for the purposes of this study, CSC was considered only in its role of compiler writer; the implementation information in the returned JAQ pertains to a compiler produced by means of CSC's GENESIS system for the IBM System 360.

Compiler Hardware Configuration:

IBM System 360
 Storage (Memory) 256K 8 bit bytes

Operating System:

Name: OS
 Developer: IBM
 Mode of Operating System: Batch Processing

2.7 Application: Not applicable

2.8 Interview Notes:

Because CSC was being interviewed in the role of compiler implementor, the usual interview format was suspended. Instead, an informal discussion regarding the implementation of features took place. Acknowledgement is made to Mr. Dunbar for his excellent remarks in this area.

- The standard as described in AFM 100-24 implies an "all or nothing" implementation effort. Implementation "levels" or a feature or function module approach would be preferable.
- To achieve full compliance with AFM 100-24's "signed magnitude" fixed point number convention is an extremely expensive proposition.
- Full implementation of fixed point representation is very expensive.
- Implementation of rounding is expensive.
- Extended capabilities should be provided for "modelling" the computer memory, other than simply the notion of "words". System 360 has, in effect, four addressable units (bytes, half-, full-, and double-word) and also the notion of word boundaries, for example. An extended memory modelling capability would probably result in the generation of more efficient object code.
- Standard JOVIAL J3 array presetting is inefficient in that array elements are input in an order differing from the allocation order; and that there is no way for the programmer to "skip" over elements he does not desire to preset.
- AFM 100-24 should be rewritten. It is overly cumbersome for the compiler implementor; also certain "holes" exist in the presentation.
- In general, there are too many specific representation requirements to which the implementor must adhere. Frequently, a property of a given machine will lend itself to a more economic representation than that required in AFM 100-24.

In addition to the above remarks, Mr. Dunbar has provided a list of loop-holes and ambiguities which CSC has encountered in implementing JOVIAL compilers based on AFM 100-24. This list shows interpretations which CSC has made in order to implement compilers. Also shown are certain comments regarding the JOVIAL J3 compiler specifications, not included in this study. DDI feels that this list is very important in that it sheds light on problem areas which only an implementor might be expected to discover. For this reason, we incorporate the following comments from CSC into this report.

CED-2400 as a contractual document leaves much to be desired as there are many areas which are not covered at all or are ambiguous. The following is a list of these areas and the interpretation we (CSC) are taking. Also included are exceptions that must be taken because of incompatibilities with the hardware. Each item on the list references CED-2400 by section and page.

- 1) The floating point description has an error in it; part of a sentence is missing, leaving both the signicand and the exrad undefined. We use the hardware representation. (2416.7; p. 10)
- 2) Null literal constants, i.e., 0 H(), 0 T(), 0(), are not defined. We define them all to be illegal. (2418.5, .8, .9; p. 11).
- 3) The sign-magnitude effect is incompatible with most hardware. As filler bits arose as a complement to sign-magnitude, the compiler will ignore them and treat the bit to the left of BIT(\$ 1 \$) as BIT(\$ 0 \$) for reasons of efficient object programs. (2419.1, .2, 2426.6; p. 12, 16).
- 4) The precision and range of index variables are not defined. It should be stated that each implementation specify precisely these attributes in the form of an implied item declaration: ITEM letter I n S \$. (2426.5; p. 16).
- 5) ODD (variable) shall be interpreted as the rightmost bit of the variable (2429; p. 17-8).
- 6) Index computation should not follow rules which differ from normal arithmetic computation. It is unreasonable to compute a floating constant + fixed variable as fixed. (2434.6; p. 20).
- 7) Floating point rounding is not consistent with fixed-point rounding. (2434.6; p.20).
- 8) LOC of a TABLE is not described clearly since it is defined in terms of a 'control register' which is undefined. We are defining LOC (TABLE) to be the address of the first word in ENTRY (TABLE (\$ 0 \$)). (2434.10, .11; p. 20).
- 9) Scaling formulas are not complete nor well defined. In particular scaling for division has a missing formula, i.e., IR for A/I. Our interpretation changes formula 2435.11 (11) to read IR=IN+AD+1-MD., thus deleting the qualifying clause. (2435; p. 21-2).
- 10) As a point of interest, it should be noted that mixing of literal formula types in relational expressions, assignments, I/O, and presets involve no conversions. (2437.2, 2442.4, 2443, 2447-9, 2470; p. 23, 26-7, 29-31, 45).
- 11) Although ENTRY variables are not excepted, they are meaningless as actual parameters. (2446.1a., .4a.; p. 28).

- 12) Since "type" is referred to in several contexts without a definition, it is not apparent whether it is up to the programmer or the compiler to ensure parameter compatibility. A technical interpretation of CED-2400 would indicate that actual and formal parameters must agree, thus requiring no conversions. However, based upon historical JOVIAL and what we feel to be the intent of the author, the compiler will generate code to ensure type compatibility for value parameters. CED-2400 does not define the distinction between the data parameters passed by name (tables and arrays) used as input parameters and those used as output parameters. We choose to make no distinction. (2446.7..8; p. 28-9).
- 13) Since the manual is not clear as to whether iteration means loop:variable incrementation or iterative execution of the loop body, the definition of a one:factor:for:clause is ambiguous. In our compiler no iteration or looping will occur. (2455; p. 33-4).
- 14) Array presets should be specified in allocation order and expanded. (2467; p.43).
- 15) Type matching for literal item presetting seems overly restrictive. We intend to permit complete mixing of literal values with literal items. (2470; p. 45).
- 16) For defined entry table items, we will ignore the packing specifier. This interpretation was chosen to alleviate the problem arising when the specified allocation conflicts with the packing specifier. (2474-5; p. 50-2).
- 17) CED-2400 does not define COMPOOL resolutions of like table items. Given an explicit COMPOOL declaration and an implicit item declaration via a like table declaration within the source program, resolution will be as follows:
 - a) If a reference to the name occurs before the declaration, the COMPOOL declaration is used and a duplicate declaration exists.
 - b) If the like table declaration occurs before the reference, the implied declaration holds. (2476; p. 53).
- 18) Our interpretation of CED-2400 is that the manual precludes the use of optional origins. (2479, 2487; p. 55,60).
- 19) The index switch list syntax permits at most 2 consecutive null switch points. We feel this is an error and not the intent of the author. The compiler will not restrict the number consecutive null switch points except to the maximum switch points. (2481.1; p. 56).
- 20) Since the constants of an item switch list must be possible values of the switch item, we interpret CED-2400 to mean that no conversions will be necessary. (2482.1, .4; p. 56).
- 21) Although it is not excepted by the manual, we will not permit presetting formal parameters or a function result item. (2484-5; p. 57-9).

- 22) An "is" is interpreted as an "it". (2484.9; p. 58).
- 23) CED-2400 refers to a name before START; we interpret this to mean only the case of a compile above CLOSE. (2487.16., 2489.3)

in addition the following exceptions are taken to Part 1 of CED-2400 on compiler specifications.

- 1) All mention of library, procedure library and library maintenance programs should be taken in light of the fact that the standard computer manufacturer's operating system, library and library maintenance will be used.
- 2) Sequence numbers will not be assigned to statements as described; instead card numbers will be used for error reporting. (3.2.1.8; p. 69)
- 3) The compiler will not produce stand-alone programs. (3.2.4; p. 70).
- 4) Control input will obey computer operating system, not JOVIAL, conventions. (3.2.5; p. 70).
- 5) Error detection, reporting, and correction will not be as described in CED-2400. Error messages referencing card numbers will be output rather than error numbers. Not reporting errors caused by a previous error is not always possible. Calls to an error monitor procedure will not be generated. (3.2.6; p. 70-1).
- 6) Equipment, operator, and monitor - error checking will be performed by and obey computer operating system conventions.
- 7) Compiler capacities and efficiency will be as described in the contract (3.2.9, 3.2.10; p. 71-2).
- 8) Debug aids (listings) will not necessarily be as described in CED-2400. Alter update will be performed by standard computer operating system routines. (3.2.11; p.72-4).
- 9) COMPOOL will not conform exactly to the specifications in CED-2400. Arrays will be permitted in the COMPOOL. COMPOOL addresses will be determined by the standard compiler algorithms and will be relocatable. There will be no COMPOOL disassembly program as such. A storage map will be produced, optionally, as part of the assembly process.
- 10) Documentation and quality assurance will be as specified in the contract (3.5,4.;p.75-79).

3.0 FAA

3.1 Organization: FAA
800 Independence Avenue, S.W.
Washington, D.C.

3.2 Contact: Mr. Donald Scheffler

3.3 Organization Identifier: FAA

3.4 Number of JAQ's Returned: 1

3.5 Date of Interview: 9 May 1968

3.6 JAQ Response Notes:

- Application name: Not given
- Application description: Not given

NOTE: It was learned at interview that the chief use of JOVIAL in the FAA is in the National Air Space Program.

3.7 Application:

Hardware Configuration:

Computer manufacturer and model IBM 7090

High speed memory storage 32K words
Size
Word size if applicable

Input/Output devices

Drum

Manufacturer and model None
Storage capacity
Quantity on this configuration

Disc

Manufacturer and model None
Storage capacity
Quantity on this configuration

Magnetic tapes

Manufacturer and model
Recording densities permitted
Quantity on this configuration

Card punches
Manufacturer and model
Card punch speed
Quantity on this configuration

Card reader
Manufacturer and model
Card read speed
Quantity on this configuration

Printer
Manufacturer and model
Printer speed
Quantity on this configuration

Paper tape reader
Manufacturer and model
Tape read speed
Quantity on this configuration

Paper tape punch
Manufacturer and model
Tape punch speed
Quantity on this configuration

Operating System:

Name: FORTRAN Monitor System
Developer: IBM
Mode of Operation: Batch processing

Background:

- JOVIAL was used because it was regarded as the language best suited for the project.
- Average years experience of programmers: 5
- Average years JOVIAL experience: 2
- Dialect used: JOVIAL J2
- Compilation speed: Unknown
- Compiled code: Not relocatable.

Total Number of JOVIAL Statements: Not given.

NOTE: No counts or estimates of usages were given in the JAQ returned. Qualitative phrases were used instead. We have quantified these phrases, as shown below:

Occasionally	.2
Frequently	.8
Almost Always	.9
Never	0
Generally	.5
Constantly	.9
Always	.9
None	0
Not Available	0
Rarely	.1
Infrequently	.2
Continually	.9

3.8 Interview Notes:

FAA feels it would be desirable to allow: IF AA EQ BB,CC,DD ...
for IF AA EQ BB OR AA EQ CC OR AA EQ DD

4.0 FOCCPAC

4.1 Organization: FOCCPAC
U.S. Navy
Honolulu, Hawaii

4.2 Contact: Capt. Bishop

4.3 Organization Identifier: FOCC

4.4 Number of JAQ 's Returned: 2

4.5 Date of Interview: 22 May 1968

4.6 JAQ Response Notes:

Each of the two JAQ's returned contains responses for two distinct sub-applications. These are identified by FOCC-1 and FOCC-2. Both run on the same computer and use the same compiler. Compliance information was taken from the JAQ for FOCC-1.

4.7 Application:

(1) Sub-application FOCC-1

Sub-application Name: Patrol Report -- NAVCOSSACT
Report 0193 11P027

Sub-application Description:

The purpose of the sub-application is to maintain a history of data pertaining to the operations of Polaris Submarines and upon request to evaluate this data.

Total Number of JOVIAL Statements: 13,388

Sub-application Hardware Configuration:

Computer manufacturer and model	<u>Control Data Corporation 1604A & 160A</u>	
	<u>160A</u>	<u>1604A</u>
High speed memory storage	16K(4 banks)	32K words
Size	2/word(2 banks-160A; 2 banks-169 core storage only)	8/word six bit bytes
Word size if applicable	12 bits	48 bits
Input/Output devices		
Drum		
Manufacturer and model		None
Storage capacity		
Quantity on this configuration		
Disc		
Manufacturer and model		1301 IBM
Storage capacity		28X10 ⁶ six bit bytes
Quantity on this configuration		
Magnetic tapes		
Manufacturer and model		729 IV IBM
Recording densities permitted		556 bits/in
Quantity on this configuration		800
Card punches		
Manufacturer and model		1402 IBM
Card punch speed		250 cards/min
Quantity on this configuration of this type		
Card reader		
Manufacturer and model		1402 IBM
Card read speed		600 cards/min
Quantity on this configuration		

Printer

Manufacturer and model
Printer speed
Quantity on this configuration

Paper tape reader

Manufacturer and model
Tape read speed
Quantity on this configuration

Control Data 350
350 frames/sec

Paper tape punch

Manufacturer and model
Tape punch speed
Quantity on this configuration

Teletype BRPE 11
110 frames/sec

Operating System:

Name: MESIM
Developer: NAVCOSSACT
Mode of operation: "UNIT"
NOTE: With the exception of compilations, all programs are run individually,
each being manually loaded.

Background:

- JOVIAL was used because it was considered the language best suited for the project.
- Average years experience of programmers: 4
- Average years JOVIAL experience: 1
- Compilation speed: 210 statements per minute
- Compiled code: Optionally relocatable

(2) Sub-application FOCC-2

Sub-application Name: Afloat Cost, Consumption, Effectiveness Surveillance System

Sub-application Description:

The purpose of the sub-application is to collect and summarize basic financial consumption data generated in the normal course of supply operations afloat.

Total Number of JOVIAL Statements: 450

Sub-application Hardware Configuration: See under FOCC-1.

Operating System:

Name: Subsystem Executive
Developer: Nat given
Mode of operation: Batch processing.

Background:

- JOVIAL was used because it was the only high-level language available.
- Average years experience of programmers: 2
- Average years JOVIAL experience: 2
- Dialect used: JOVIAL J3
- Compilation speed: Nat given
- Compiled code: Relocatable.

4.8 Interview Notes:

The job of the FOCCPAC personnel is to maintain programs written by NAVCOSSACT and NAVCOSSACT contractors. The desire was expressed for more compiler directives or other mechanisms which would allow the programmer to effect a more efficient code generation.

5.0 HAM

5.1 Organization: Operational Program Maintenance Branch
4th Air Force Combat Center
Hamilton A.F.B.
San Francisco, California

5.2 Contact: Lt. Ruck

5.3 Organization Identifier: HAM

5.4 Number of JAQ's Returned: 1

5.5 Date of Interview: 23 May 1968

5.6 JAQ Response Notes: -

5.7 Application:

Application Description:

The application, WNR-CC, functions to monitor coordinate and direct air defense operations for WNR and coordinate air defense with adjacent regions.

Total Number JOVIAL Statements: 8850

Application Hardware Configuration:

Computer manufacturer and model Burroughs AN/GSA51

High speed memory storage 60,000g wards
Size 10,000 six bit bytes
Ward size if applicable 48 bits

Input/Output devices

Drum

Manufacturer and model Burroughs MU-469/GYK-4
Storage capacity 65,536 10 48 bit wards each
Quantity on this configuration 2 each

Disc

Manufacturer and model None
Storage capacity
Quantity on this configuration

Magnetic tapes

Manufacturer and model Burroughs AN/GSH-12
Recording densities permitted 800 bits/in
Quantity on this configuration 3 each

Card punches

Manufacturer and model IBM 026
Card punch speed
Quantity on this configuration on this type 1 each

Card reader

Manufacturer and model Burroughs AN/GSQ72
Card read speed 200 cards/min
Quantity on this configuration 1 each

Printer

Manufacturer and model
Printer speed
Quantity on this configuration

Paper tape reader

Manufacturer and model
Tape read speed
Quantity on this configuration

Paper tape punch
Manufacturer and model
Tape punch speed
Quantity on this configuration

Flexowriter - AN/GYQ-2
Controller Comparator - 2 I/O Modules C4634/GYK-4
Controller Comparator/Message Progresser
1 I/O Module 1 msg. Processor C4635/GYK-4

Operating System:

Name: Western NORAD Region CC
Developer: USAF/SDC
Mode of Operation: Real Time

Background: No background information supplied

5.8 Interview Notes: SDC is primarily responsible for the design of the HAM system.

6.0 NAVCOSSACT

6.1 Organization: NAVCOSSACT
Washington Naval Yard Annex
Washington, D.C.

6.2 Contact: Mr. Dan Foster

6.3 Organization Identifier: NAVC

6.4 Number of JAQ 's Returned: 3

6.5 Date of Interview: 8 May 1968

6.6 JAQ Response Notes:

- Each of the three JAQ 's returned contains response data for distinct sub-applications. These are identified by NAVC-1, NAVC-2, and NAVC-3.
- It is our understanding that each runs on the CDC 1604 and that the same compiler is used for each. Compliance information was obtained from the JAQ for NAVC-1.

6.7 Application:

(1) Sub-application NAVC-1:

Sub-application Name: JOVIAL Compiler

Sub-application Description:

The purpose of this sub-application is to compile programs written in JOVIAL for use on the CDC 160/1604 computer complex.

Total Number of JOVIAL Statements: Not given

NOTE: No counts or estimates of features were provided in the JAQ.

Sub-application Hardware Configuration:

Computer manufacturer and model CDC-1604

High speed memory storage	32,768 words
Size	8 six bit bytes
Word size if applicable	48 bits
Input/Output devices	
Drum	
Manufacturer and model	
Storage capacity	
Quantity on this configuration	
Disc	
Manufacturer and model	IBM 1301
Storage capacity	28,000,000 ₆ six bit bytes
Quantity on this configuration	4
Magnetic tapes	
Manufacturer and model	IBM 729
Recording densities permitted	556 bits/in
Quantity on this configuration	22
Card punches	
Manufacturer and model	IBM 1402
Card punch speed	250 cards/min
Quantity on this configuration of this type	2
Manufacturer and model	IBM 1402
Card punch speed	800 cords/min
Quantity on this configuration of this type	2
Card reader	
Manufacturer and model	Burroughs B-122
Card read speed	250 cards/min
Quantity on this configuration	1

Printer		
Manufacturer and model		Halley
Printer speed		150 lines/min
Quantity on this configuration		1
Printer		
Manufacturer and model		CDC
Printer speed		1000 lines/min
Quantity on this configuration		2
Paper tape reader		
Manufacturer and model		CDC
Tape read speed		350 frames/sec
Quantity on this configuration		3
Paper tape reader		
Manufacturer and model		CDC
Tape read speed		350 frames/sec
Quantity on this configuration		2
Paper tape punch		
Manufacturer and model		Teletype BRPE 11
Tape punch speed		110 frames/sec
Quantity on this configuration		5

Operating System:

Name: AN FYK-1(V) 1604A OPCON
 Developer: Auerbach Corporation
 Mode of Operation: Batch processing

Background:

- JOVIAL was used because it was the official NAVY language.
- Average years experience of programmers: 3
- Average years JOVIAL experience: 3
- Dialect used: JOVIAL J3
- Compilation speed: Not given.
- Compiled code: Relocatable.

(2) Sub-application NAVC-2:

Sub-application Name: None given

Sub-application Description: Compiler Maintenance Activity

Total Number of JOVIAL Statements: Not given

Sub-application Hardware Configuration: See under NAVC-1

Operating System: No information given.

Background:

- JOVIAL was used in order to remain compatible with related projects and because it was the best suited language for the project.
- No further background information was given.

(3) Sub-application NAVC-3

Sub-application Name: Route generation/Map generation (RGMG)

Sub-application Description:

Given a latitude and longitude, the sub-application RGMG generates a binary map of the world or a sea route (land-mass avoidance) according to input specifications. The binary map must reside in core before a route of turning points can be generated. RGMG is compiled as a closed subroutine and is operational as a part of a larger sea surveillance system. In addition it can be run independently and has been run using the PPS Test function to take advantage of the recording capability (RGMG does not write out the results).

Total Number of JOVIAL Statements: 2903

Sub-application Hardware Configuration: See under NAVC-1

Operating System: See under NAVC-1

Background:

- JOVIAL was used to remain compatible with other related projects.
- Average years experience of programmers: 0
- Average years JOVIAL Experience: 0
- Dialect used: JOVIAL J3
- Compiler speed: Not given
- Compiled Code: Not relocatable

6.8 Interview Notes:

- It was stressed that the major consideration in the use of JOVIAL is the ease of programming it provides.

- NAVCOSSACT feels that the readability of JOVIAL code in certain areas (e.g., item declarations) could stand some improvement.

7.0 NCDCF

- 7.1 Organization: National Civilian Defense Computer Facility
Olney, Maryland
- 7.2 Contact: Mr. Marvie Lee
- 7.3 Organization Identifier: NCDCF
- 7.4 Number of JAQ 's Returned: 1
- 7.5 Date of Interview: 7 May 1968
- 7.6 JAQ Response Notes:
- 7.7 Application:

Application Description:

The Dash Executive control system assists in using and monitoring the execution of a system of JOVIAL programs.

Total Number of JOVIAL Statements: 1600

Application Hardware Configuration:

Computer manufacturer and model CDC 3600

High speed memory storage 65536 words

Size

Word size if applicable 48 bits

Operating System:

Name: SCOPE

Developer: CDC

Mode of Operation: Batch Processing.

Background:

- JOVIAL was used to maintain source language compatibility with related projects.
- Average years experience of programmers: 4
- Average years JOVIAL experience: 1

- Dialect used: JOVIAL J3
- Compilation speed: Not given
- Compiled code: Not given

7.8 Interview Notes:

- NCDCF personnel interviewed are primarily responsible for maintaining a JOVIAL compiler.
- An opinion was expressed that list processing facilities might find some use in NCDCF's application.

8.0 NEL

8.1 Organization: Naval Electronics Laboratory
Command and Control Facility
San Diego, California

8.2 Contact: Mr. Barksdale

8.3 Organization Identifier: NEL

8.4 Number of JAQ's Returned: 1

8.5 Date of Interview: 24 April 1968

8.6 JAQ Response Notes:

8.7 Application:

Application Description:

The purpose of the Navy Command and Control application is to maintain and test compilers.

Total Number of JOVIAL Statements: 20K

Application Hardware Configuration:

Computer manufacturer and model UNIVAC CP 667

High speed memory storage	131K words
Size	6 six bit bytes
Word size if applicable	36 bits

Input/Output devices
Drum

Manufacturer and model	Univac Fastran
Storage capacity	132X10 ⁶ six bit bytes 702X10 ⁶ bits
Quantity on this configuration	1
Magnetic tapes	
Manufacturer and model	CDC 603's & 606's
Recording densities permitted	200 bits/in 556 bits/in
Quantity on this configuration	8
Card reader	
Manufacturer and model	Univac 1449
Card read speed	100 cards/min
Quantity on this configuration	1
Paper tape reader	
Manufacturer and model	Univac 1232
Tape read speed	350 frames/sec
Quantity on this configuration	1
Paper tape punch	
Manufacturer and model	Univac 1232
Tape punch speed	150 frames/sec
Quantity on this configuration	1

Operating System:

Name: JOVIAL
Developer: PRC and NELC
Mode of operation: Batch processing

Background:

- JOVIAL was used because it was regarded as the best suited language for the project.
- Average years experience of programmers: 5
- Average years JOVIAL experience: 1
- Dialect used: JOVIAL J3
- Compilation speed: 300-500 statements/minute
- Compiled code: Relocatable

8.8 Interview Notes:

- It is the intent of the Navy that the following philosophy be employed in the design and implementation of Command & Control systems: A clear separation of

functional software from (machine) system support software is to be maintained. This means that NEL would prefer a language as machine independent as possible. The discussion also produced the counter idea - the essential strong point of JOVIAL, as far as NEL is concerned, is the set of machine dependent features it contains. It was suggested that a possible resolution of these two conflicting notions would lie in a kind of (COBOL-like) "Environment Division". The Navy wants to be able to transfer programs between computers; in other words the functions performed by a given application may have to be executed on a variety of different machines.

- JOVIAL is too verbose. It would be desirable to have "compound" item declarations, for example (ITEM AA I 48 S BB I 48 S, etc.)
- NEL would (and does) like to make use of an expanded character set. They use braces for BEGIN END. This helps attack the verbosity problems.
- It was suggested that a "TYPE OTHER" (CDC FORTRAN 63) facility be considered.

9.0 NMCSSC

9.1 Organization: DCA
National Military Command
System Support Center
Pentagon
Washington, D.C.

9.2 Contact: Colonel Gallentine

9.3 Organization Identifier: NMCSSC

9.4 Number of JAQ's Returned: 1

9.5 Date of Interview: 20 June 1968

9.6 JAQ Response Notes:

9.7 Application:

Application Description:

The Damage and Contamination Model associates military nuclear weapons against all kinds of targets and predicts damage. The pertinent parameters are coordinates, wind data, height of weapon burst, weapon yield, vulnerability, number of targets, and target capacity.

Total Number of JOVIAL Statements: 5000

Application Hardware Configuration:

High speed memory storage	32K words
Size	8 six bit bytes
Word size if applicable	48 bits
Input/Output devices	
Drum	
Manufacturer and model	None
Storage capacity	
Quantity on this configuration	
Disc	
Manufacturer and model	None
Storage capacity	
Quantity on this configuration	
Magnetic tapes	
Manufacturer and model	CDC 606
Recording densities permitted	500 bits/in
Quantity on this configuration	12
Card punches	
Manufacturer and model	IBM
Card punch speed	50 cards/min
Quantity on this configuration of this type	1
Card reader	
Manufacturer and model	CPC 405
Card read speed	1200 cards/min
Quantity on this configuration	1
Printer	
Manufacturer and model	CDC 501
Printer speed	1000 lines/min
Quantity on this configuration	1
Paper tape reader	
Manufacturer and model	CDC
Tape read speed	
Quantity on this configuration	1
Paper tape punch	
Manufacturer and model	CDC
Tape punch speed	
Quantity on this configuration	1

Operating System:

Name: COOP
Developer: CDC
Mode of Operation: Batch Processing

Background:

- JOVIAL was used in order to stay compatible with other projects and because it was the language best suited for the project.
- Average years experience of programmers: 2
- Average years JOVIAL experience: 1
- Dialect used: JOVIAL J3
- Compilation speed: 44 statements per minute
- Compiled Code: CODAP symbolic which is then assembled in a relocatable fashion.

9.8 Interview Notes:

- A desire was expressed to have the compiler produce code in the form of the assembly languages supplied by the computer manufacturer; also, the subroutine linkages of the language should be in such a form as to be compatible with the computer manufacturer's system.
- NMCSSC also requires the ability to process machine interrupts.

10.0 NHCP

10.1 Organization: Headquarters NORAD
NHCP
Ent Air Force Base
Colorado Springs, Colorado

10.2 Contact: Mr. N. R. Gerhart

10.3 Organization Identifier: NHCP

10.4 Number of JAQ's returned: 2*

10.5 Date of Interview: -

10.6 JAQ Response Notes:

The JAQ's returned were not received in sufficient time for their contents to be processed.

10.7 Application: -

10.8 Interview Notes:

- NHCP desires facilities for tables containing variable length entries.
- Multiple assignment statements of the form
 $AA = BB = CC = 0\$$
would be found useful. The effect of the above statement is produced by
 $AA = 0\$$
 $BB = 0\$$
 $CC = 0\$$
- A need for a 'pro-word" such as "SELF" or "*" was voiced.
 $VARIABLE = \{SELF\} + 3 \$$ or $VARIABLE = * + 3 \$$
produces the effect
 $VARIABLE = VARIABLE + 3 \$$
- Extended use of ranged subscripts would be helpful:
 $TABLE'NAME (\$ 3 . . . 12 \$)$
has the effect of naming a table structure made up of entries 3 through 12.
- NHCP would like the ability to show a set of loop variable increment values to be taken in FOR loop processing:
 $FOR I = 1, (5, 3, 6, 19), NN$
Each time through the increment handling part of the loop, the next increment "to the right" in the list is used.
- NHCP would prefer the use of parentheses instead of the JOVIAL index brackets (\$ and \$) when writing subscripts.
- A desire to include \$ in comments was mentioned.
- NHCP feels that non-named variables should be allowed as operands of the BIT and BYTE functional modifier, e.g.,
 $BIT(\$ 3, 6 \$) (BYTE (\$5, 2\$) (AA))$

11.0 RADC

- 11.1 Organization: Rome Air Development Center, EMIIF
Griffis Air Force Base
Rome, New York
- 11.2 Contact: Mr. Dick Motto
- 11.3 Organization Identifier: RADC
- 11.4 Number of JAQ's Returned: 4
- 11.5 Date of Interview: 1 May 1968
- 11.6 JAQ Response Notes:

Each of the three JAQ's returned contains response data for three distinct sub-applications. These are identified as RADC-2, RADC-3, and RADC-4.

- The JAQ returned by RADC-1 did not give usage counts.
- RADC-1 is run on the CDC 1604B.
- RADC-2 is run on the Univoc M555.
- RADC-3 is run on the GE 635.
- RADC-4 is run on the GE 635.
- It is our understanding that RADC-3 and RADC-4 use the same compiler.

11.7 Application:

(1) Sub-application RADC-1

Sub-application Name: None given.

Sub-application Description:

- MADAPS - a data management system
- TINT - an on-line JOVIAL (subset) compiler
- JOVIAL COMPILER
- AUERBACH PROGRAM

Total Number JOVIAL Statements: Not given.

NOTE: No counts or estimates of usage were entered in this JAQ.

Sub-application Hardware Configuration: Not given.

Operating System: No information given.

Background: No information given.

(2) Sub-application RADC-2

Sub-application Name: Data Management I

Sub-application Description:

Data Management I is a data management system.

Total Number of JOVIAL Statements: Not given.

NOTE: It was stated in the JAQ for RADC-2 that because the compiler is in a continuous state of flux and improvement, and since no major programs have yet been written, usage counts have not been entered. Rather, qualitative statements have been made regarding

usage rates (e.g., very frequently, seldom, etc.) We have quantified these statements. The following is a list of the qualitative statements used with their associated quantifiers:

OFTEN	.7
VERY FREQUENTLY	.9
SOME	.3
SELDOM	.2
SOMETIMES	.2
ALWAYS	.9
NEVER	0
VERY SELDOM	.1
VERY OFTEN	.9
FREQUENTLY	.7

Sub-application Hardware Configuration:

Computer manufacturer and model: Univac M555

High speed memory storage 64K words
Word size: 18 bits

Operating System:

Name of operating system: ECP IT
Developer: Not given
Mode of operation: Not given

Background:

- JOVIAL was used in this project both because it was the official Air Force language and because it was the language best suited for the project.
- No other background information was entered.

(3) Sub-application RADC-3

Sub-application Name: TINT

Sub-application Description: "On-line compiler"

Total Number of JOVIAL Statements: 9142

Sub-Application Hardware Configuration:

Computer manufacturer and model GE 635

High speed memory storage	128K words
Size	
Word size if applicable	36 bits
Input/Output devices	
Drum	
Manufacturer and model	GS MDS 200
Storage capacity	30M bits
Quantity on this configuration	1
Disc	
Manufacturer and model	GS DSU 200
Storage capacity	138M bits
Quantity on this configuration	2
Magnetic tapes	
Manufacturer and model	GS MT #67
Recording densities permitted	200/566/800 bits/in
Quantity on this configuration	8
Card punches	
Manufacturer and model	GE 4WCPZ201A1
Card punch speed	300 cards/min
Quantity on this configuration of this type	1
Card reader	
Manufacturer and model	GS 4WCRZ201A1
Card read speed	800-1000 cards/min
Quantity on this configuration	
Printer	
Manufacturer and model	GE PRT 200
Printer speed	1200 lines/min
Quantity on this configuration	2

Operating System:

Name: GECOS
Developer: General Electric Co.
Mode of Operating System: Batch processing

Background:

- JOVIAL was used because it was the official Air Force language, because it was desired to maintain source language compatibility with other projects, and because it was the best suited language for the project.

- Average years experience of programmers: 3-5
- Average years JOVIAL experience: 3-5
- Dialect used: JOVIAL J3
- Compilation speed: 50 statements per minute
- Compiled code: Relocatable

(4) Sub-application RADC-4

Sub-application Name: Auxiliary Control Executive Processor

Sub-application Description: "Preprocessor for on Executive Control Program."

Total Number of JOVIAL Statements: 3951

Sub-application Hardware Configuration: No details given. It is our understanding, however, that the computer is the GE 635.

Operating System: No information given.

Background:

- JOVIAL was used to maintain compatibility with other related projects, because it is the official Air Force language and because it is the language best suited for the project.
- Average years experience of programmers: 4
- Average years JOVIAL experience: 1
- Dialect used: JOVIAL J3
- Compilation speed: None given.
- Compiled code: Relocatable.

11.8 Interview Notes:

RADC would like the ability to specify (of compile or load time) the allocation sequencing of both programs and data; this is felt a more important facility than dynamic allocation.

12.0 SAC

12.1 Organization: Headquarters SAC
Offut Air Force Base, Nebraska

12.2 Contacts: Lt Col F.L. Maloy, DOCODS
Major Samuel P. Herod

12.3 Organization Identifier: SAC

12.4 Number of JAQ 's Returned: 4

12.5 Date of Interview: 13 May 1968

12.6 JAQ Response Notes:

- Each of the four JAQ's returned contains response data for four distinct sub-applications. These sub-applications are identified as SAC-1, SAC-2, SAC-3, and SAC-4.
- Sub-applications SAC-1, SAC-2, and SAC-3 each run on the AN/FSQ-31 and each uses the same compiler. Implementation responses for SAC-1, SAC-2, and SAC-3 were taken from the JAQ for SAC-1.
- Sub-application SAC-4 is run on the IBM 7090; implementation responses are taken from the JAQ for SAC-4.

12.7 Application:

(1) Sub-application SAC-1

Sub-application Name: Utility Subsystem

Sub-application Description:

The Utility subsystem includes the following capabilities:

- executive
- compiler
- tape update
- COMPOOL
- data generator
- data reduction
- dump
- CHECKER
- task parameter assembler
- tape maintenance
- loader maintenance

Total Number JOVIAL Statements: 116,137

Sub-application Hardware Configuration:

Computer manufacturer and model IBM AN/FSQ-31 Data Processing Central

High speed memory storage	65,536 words
Size	524,288 six bit bytes
Word size if applicable	48 bits

Input/Output devices

Drum	19
Manufacturer and model	IBM (na model)
Storage capacity	1,114,112 six bit bytes 6,684,672 bits
Quantity on this configuration	2
Disc	
Manufacturer and model	Bryant Madel L
Storage capacity	103,809,024 six bit bytes 622,854,144 bits
Quantity on this configuration	1
Magnetic tapes	
Manufacturer and model	IBM 729 II, IBM 729 IV
Recording densities permitted	200/556 bits/in
Quantity on this configuration	12
*Card punches	
Manufacturer and model	IBM 729 II, IBM 729 IV
Card punch speed	250 cards/min
Quantity on this configuration of this type	1
*Card reader	
Manufacturer and model	IBM 1402
Card read speed	800 cards/min
Quantity on this configuration	1
*Printer	
Manufacturer and model	IBM 1403
Printer speed	600 lines/min
Quantity on this configuration	1

*Available as part of IBM 1401 systems which can be used on-line. However, printing and punching is normally accomplished off-line.

NOTE: There are two IBM AN/FSQ-31 computers. Each with an I/O typewriter, fix typewriter, Disc, 2 drums, 15 tape drives and 1401 system.

Operating System:

Name: EXECUTIVE
Developer: SDC/USAF
Mode of operation: Batch processing
Real Time

NOTE: "The utility subsystem operates only in the batch processing mode. However, it can operate 'interleaved' with the real time programs."

Background:

- JOVIAL was used both to maintain source language compatibility with other related projects and because it was the language best suited for the project. Also stated: "JOVIAL was devised for SACCS system by SDC - who wrote the utility subsystem."
- Average years experience of programmers: "Unknown"
- Average years JOVIAL experience: "Unknown - perhaps about 2"
- Dialect used: JOVIAL J2
- Compilation speed: 200 statements per minute
- Compiled code: Relocatable

(2) Sub-application SAC-2

Sub-application Name: Planning

Sub-application Description:

The Planning Branch furnishes the automated support for development and maintenance of the Single Integrated Operations Plan and other SAC planning functions. This includes missile and aircraft application against target systems with all the associated ramifications: bomber/tanker mating, fuel consumption, performance characteristics, optimum routine resolution of timing conflicts, etc. Also included is the production of planned and emergency execution documents and statistical documents.

Total Number of JOVIAL Statements: 300,000 (est)

Sub-application Hardware Configuration: See under SAC-1

Operating System:

Name: EXECUTIVE

Developer: SDC/USAF

Mode of operation: Batch Processing Real Time

NOTE: "The planning subsystem operates only in the batch processing mode. However, it can operate 'interleaved' with the real time programs."

Background:

- JOVIAL was used both to maintain source language compatibility with other related projects and because it was the best language suited for the project.
- Average years experience of programmers: 3
- Average years JOVIAL experience: 3
- Dialect used: See under SAC-1
- Compilation speed: See under SAC-1
- Compiled code: See under SAC-1

(3) Sub-application SAC-3

Sub-application Name: Force Control

Sub-application Description:

The Force Control sub-application modifies a static data base by processing message and card inputs to provide or present status of SAC SIOP Forces through various prints, wall displays and BOJO displays. DATA PREP modifies and error checks messages from units for use by V-1. V-1 updates tables and displays; 200 PACKAGES uses updated data to create displays and prints. DATA PRES creates the displays that are projected to wall screens.

Total Number of JOVIAL Statements: 153,789

Sub-application Hardware Configuration: See under SAC-1

Operating System:

Name: EXECUTIVE

Developer: SDC/USAF

Mode of operating system: Batch processing real time

Background:

- JOVIAL was used both to maintain source language compatibility with other related projects and because it was the language best suited for the project.
- Average years experience of programmers: 2
- Average years JOVIAL experience: 2
- Dialect used: See under SAC-1
- Compilation speed: See under SAC-1
- Compiled code: See under SAC-1

(4) Sub-application SAC-4

Sub-application Name: SIOP Gaming Simulation Analysis/ADA

Sub-application Description:

The SIOP Gaming Simulation extracts data from RISOP/SIOP tapes furnished by JWGA and prepares data for War Gaming Simulation and Analysis. The model includes trajectory computations and damage assessments.

The purpose of the ADA is to provide an AUSTERE DEMONSTRATION SYSTEM (Hardware and Software) in a PACCS aircraft to develop and demonstrate operational techniques in the actual environment using available automation hardware and a minimum of development effort.

Total Number of JOVIAL Statements: Gaming - 132,000
ADA - 90,000

Application Hardware Configuration:

Computer manufacturer and model IBM 7090

High speed memory storage	65,536 words
Size	393,216 six bit bytes
Word size if applicable	36 bits
Input/Output devices	17
Drum	
Manufacturer and model	
Storage capacity	
Quantity on this configuration	0
Disc	
Manufacturer and model	IBM 1301-1
Storage capacity	27,960,000 six bit bytes 167,760,000 bits
Quantity on this configuration	1
Magnetic tapes	
Manufacturer and model	IBM 729 IV
Recording densities permitted	200/556 bits/in
Quantity on this configuration	12
Card reader	
Manufacturer and model	IBM 711
Card read speed	250 cards/min
Quantity on this configuration	1
Printer	
Manufacturer and model	IBM 716
Printer speed	150 lines/min
Quantity on this configuration	1

NOTE: One I/O typewriter and one Hendrix 5101 crt are attached to the IBM 7090.

NOTE: 2 IBM 1460 computers (with MOD III printers, MOD III card reader-punches) plus a 5 track paper tape reader-punch are used off-line to print and punch 7090 outputs.

Operating System:

Name: (Gaming) None used
(ADA) ADA EXECUTIVE

Developer: (Gaming) N/A
(ADA) SAC

Mode of operating system: (Gaming) N/A
(ADA) Real Time

(Ncte: ADA does utility work under IBSYS)

Background:

- Both ADA and Gaming used JOVIAL to remain source language compatible with other projects and because it was the best suited language for the project.
- Average years experience programmers: Gaming, 2.5
ADA, 2.7
- Average years JOVIAL experience: Gaming, 2.2
ADA, 2.0
- Dialect used: JOVIAL J2 (Gaming and ADA)
- Compilation speed: 200 statements per minute (Gaming and ADA)
- Compiled code: Not relocatable (Gaming and ADA)

12.8 Interview Notes:

General:

SAC may be going over to third generation equipment within the next two years; it is expected that the corresponding software systems will operate in a time shared mode. SAC would like to transfer programs already written to the new equipment with as little reprogramming as possible, but admits that a lot of reprogramming may be required.

Specific:

- SAC feels that the notion of COMPOOL has outlived its usefulness, though it is admitted that this feeling is influenced by the fact that the COMPOOL implementation used is overly complex and very slow - both in terms of machine time and personnel interface required. It is our impression that SAC would like facilities for operationally imposing any conceivable kind of structuring over a relatively unstructured "free standing" data base in a short order. Toward this end SAC is currently examining TDMS (ADEPT).
- SAC requires the ability to manipulate disc and drum addresses as well as main memory addresses when doing I/O; further, they would like the capability of incrementing buffer addresses as required.

13.0 TWA

13.1 Organization:

TWA Data Processing Center
King Road, Rockleigh, N. J.

13.2 Contact:

Mr. Duane Witlaw

- 13.3 Organization Identifier: TWA
- 13.4 Number of JAQ's Returned: 0
- 13.5 Date of Interview: 3 May 1968
- 13.6 JAQ Response Notes:

Owing to an extremely tight schedule, TWA was unable to complete a JAQ. From our conversation at the interview and from a perusal of their JOVIAL Reference Manual, it appears that the TWA JOVIAL complies fairly closely to the standard of AFM 100-24. Particularly interesting are the several special features provided:

- The Burroughs computer on which the application is run is a multiprocessing machine. A BRANCH statement is provided to allow programmers to assign program segments to processors for execution (several segments may run in parallel). An ENDBRANCH Statement causes execution on a processor to cease.
- The AWAIT Statement permits the suspension of a parallel path of execution until the occurrence of some condition.
- READY, RELEASE, and SETUP statements provide the capabilities for dynamic allocation of data.
- ECI (Entry Control Item) and KEY allow for the manipulation of tables with both variable length entries and variable structure entries at execution time.
- The JOVIAL compiler was written by Burroughs.

13.7 Application: No data

13.8 Interview Notes:

General:

- TWA feels that the language (except as noted in the specific interview responses) is essentially sufficient for its application. TWA believes its big problem lies in the area of imposing control over all the diverse elements of the system at large. That is, TWA would like the ability to link COMPOOL, program libraries, and compiler, etc., together in such a way that any change to programs or data structures would result automatically in the output of messages stating what other parts of the system would be effected.
- TWA desires to keep the machine separated from the functions as much as possible. Toward this end, they have eliminated Direct Code from their system.

Specific:

- A need exists for "formal" bit string items and operators AND, OR, NOT, EITHER/OR.

- TWA would like to have PROC's in COMPOOL as well as data with both the definitional and external attributes.
- A desire was expressed for a kind of "DEFINE" facility:

ALPHA ('BETA)

where 'BETA (COMPOOL name) would be inserted in the text along with the ALPHA in subsequent references.

- A need for a "MAKE CURRENT" facility was voiced. Sometimes, programs will have many occurrences of the same table. It is desirable to write one segment of code which references items in one table, but then switch physical tables at execution time. (Only one physical table is being handled at a time.)

APPENDIX II

JOVIAL FEATURE RESOLUTION ANALYSIS

The following pages contain the resolution analyses for all of the features investigated in the JOVIAL Evaluation Study. The JOVIAL Evaluation Questionnaire (JAQ) and the approach for change reference numbers are provided for each feature for cross-referencing purposes. The resolution inequality, resolution equations and roles leading to each particular resolution are in accordance with the discussion in Section 1.

The application is described in Appendix 1. In the compliance column, if the feature was implemented in accordance with AFM 100-24, it is noted as CFI; if not implemented, CFNI; or if implemented differently, CFID. The usage is calculated according to the resolution equation.

Resolution equations and usage rates are given frequently in this appendix for features which are "accepted as nucleus." This is done only to provide the reader with a means of comparing the use of such features with respect to other related features.

The resolution analysis of the features are organized according to the organization of the Approach for Change (AFC) reference numbers.

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.11.1 AFC Reference: A 1.1.1.1

Feature Name: ITEM TYPE HOLLERITH

Resolution Inequality: N/A

Resolution Equation: $U = Q46 / (Q46 + Q50)$

Number of users with usage rates: $GQ TI = N/A$

Resolution: Accepted as nucleus (Note optional simplified constant form, 3.2.2.8)
Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.1333
ADPAC-2	CFI	1
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	1
FOCC-2	CFI	1
HAM	CFI	1
NAVC-1	CFI	(N/C)
NAVC-2	CFI	1
NAVC-3	CFI	1
NCDCF	CFI	1
NEL	CFI	.9803
NMCSSC	CFI	1
RADC-1	CFI	(N/C)
RADC-2	CFID	.9
RADC-3	CFI	.9524
RADC-4	CFI	1
SAC-1	CFI	1
SAC-2	CFI	1
SAC-3	CFI	.7214
SAC-4	CFI	1

N(CFI) = 21

N(CFID) = 1

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.5 AFC Reference: A 1.1.1.1
 Feature Name: HOLLERITH FORMULA
 Resolution Inequality: N/A
 Resolution Equation: N/A
 Number of users with usage rates: GQ TI = N/A
 Resolution: Retain as nucleus (Note modification to Hollerith assignment and comparisons, 3.2.2.4)
 Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFI	
NMCSSC	CFI	
RADC-1	CFI	
RADC-2	CFNI	
RADC-3	CFI	
RADC-4	CFI	
SAC-1	CFI	
SAC-2	CFI	
SAC-3	CFI	
SAC-4	CFI	

N(CFI) = 21 N(CFID) = 0 N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.11.2 AFC Reference: A 1.1.1.2

Feature Name: ITEM TYPE STANDARD TRANSMISSION CODE

Resolution Inequality: $U \text{ GQ TI} = .10$

Resolution Equation: $U = Q50/(Q46 + Q50)$

Number of users with usage rates: $\text{GQ TI} = 2$

Resolution: Delete (Note also CHARCODE directive, 3.2.2.3)

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.8666
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFID	.0196
NMCSSC	CFI	0
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	.0476
RADC-4	CFI	0
SAC-1	CFI	0
SAC-2	CFI	0
SAC-3	CFI	.2785
SAC-4	CFI	0

N(CFI) = 18

N(CFID) = 1

N(CFNI) = 2

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.5 AFC Reference: A 1.1.1.2
 Feature Name: STANDARD TRANSMISSION CODE FORMULA
 Resolution Inequality: N/A
 Resolution Equation: N/A
 Number of users with usage rates: GQ TI = N/A
 Resolution: Delete (Because of deletion of Item Type STC)

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFI	
NMCSSC	CFI	
RADC-1	CFI	
RADC-2	CFNI	
RADC-3	CFI	
RADC-4	CFI	
SAC-1	CFI	
SAC-2	CFI	
SAC-3	CFI	
SAC-4	CFI	

N(CFI) = 21

N(CFID) = 0

N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.14 AFC Reference: A4, A1.1.1.3
 Feature Name: CHARACTER DATA SIZE ATTRIBUTE
 Resolution Inequality: N/A
 Resolution Equation: N/A
 Number of users with usage rates: GQ T1 = N/A
 Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFI	
NMCSSC	CFI	
RADC-1	CFI	
RADC-2	CFI	
RADC-3	CFI	
RADC-4	CFI	
SAC-1	CFI	
SAC-2	CFI	
SAC-3	CFI	
SAC-4	CFI	

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.13 AFC Reference: A 1.1.2
 Feature Name: ITEM TYPE STATUS
 Resolution Inequality: $U \text{ GQ} = T1 = .05$
 Resolution Equation: $U = Q58/Q14$
 Number of users with usage rates: $\text{GQ T1} = 7$
 Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.0416
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	.0040
FOCC-2	CFI	.4000
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.0500
NAVC-3	CFI	.0100
NCDCF	CFI	.0200
NEL	CFI	.0080
NMCSSC	CFI	.1666
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	.0100
RADC-4	CFI	.0303
SAC-1	CFI	.1812
SAC-2	CFI	.0673
SAC-3	CFI	.1924
SAC-4	CFI	.0532

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.5 AFC Reference: A 1.1.2
Feature Name: STATUS FORMULA
Resolution Inequality: N/A
Resolution Equation: N/A
Number of users with usage rates: GQ T1 = N/A
Resolution: Retain as optional (Required if Status Items implemented)

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFI	
NMCSSC	CFI	
RADC-1	CFI	
RADC-2	CFNI	
RADC-3	CFI	
RADC-4	CFI	
SAC-1	CFI	
SAC-2	CFI	
SAC-3	CFI	
SAC-4	CFI	

N(CFI) = 21 N(CFID) = 0 N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.14 AFC Reference: A 1.1.3
 Feature Name: EXPLICIT STATUS SIZE ATTRIBUTE
 Resolution Inequality: U GQ T1 = .01
 Resolution Equation: U = Q70/Q58
 Number of users with usage rates: GQ T1 = 12
 Resolution: Retain as nucleus (Meaningful only if Item Type Status is
 Data: implemented)

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.6000
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	0
FOCC-2	CFI	.0100
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.2000
NAVC-3	CFI	0
NCDCF	CFI	0
NEL	CFI	.0500
NMCSSC	CFI	.2000
RADC-1	CFID	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	1
RADC-4	CFI	0
SAC-1	CFI	.0500
SAC-2	CFI	.1484
SAC-3	CFI	.0543
SAC-4	CFI	.0487

N(CFI) = 21

N(CFID) = 1

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.2

AFC Reference: A 1.2.1

Feature Name: STRUCTURE ARRAY

Resolution Inequality: N/A

Resolution Equation: $Q74(Q74 + Q78)$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Retain as optional (Note: In the AFC, we accepted ARRAY as a nucleus data structure. Since the number of users implementing ARRAY is relatively small, and because the usage where implemented is low, we are altering the status of this feature to "optional").

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFID	(N/A)
FAA	CFID	.2
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFID	.2500
NMCSSC	CFNI	0
RADC-1	CFID	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	.0322
RADC-4	CFI	0
SAC-1	CFID	0
SAC-2	CFID	.0181
SAC-3	CFID	.0417
SAC-4	CFID	.0676

N(CFI) = 5

N(CFID) = 8

N(CFNI) = 9

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.1 AFC Reference: A 1.2.2
 Feature Name: STRUCTURE TABLE
 Resolution Inequality: $U GQ T1 = .10$
 Resolution Equation: $U = Q78 / (Q78 + Q74)$
 Number of users with usage rates: $GQ T1 = 18$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	1
ADPAC-2	CFI	1
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFI	.9
FOCC-1	CFI	1
FOCC-2	CFI	1
HAM	CFI	1
NAVC-1	CFI	(N/C)
NAVC-2	CFI	1
NAVC-3	CFI	1
NCDCF	CFI	1
NEL	CFI	.7500
NMCSSC	CFI	1
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	.9677
RADC-4	CFI	1
SAC-1	CFI	1
SAC-2	CFI	.9818
SAC-3	CFI	.9582
SAC-4	CFI	.9323

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.4 AFC Reference: A 1.2.3
 Feature Name: TABLE VARIABILITY ATTRIBUTE - RIGID
 Resolution Inequality: $U \text{ GQ T1} = .10$
 Resolution Equation: $U = Q86/Q78$
 Number of users with usage rates: $\text{GQ T1} = 18$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.4736
ADPAC-2	CFI	.8750
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFID	.2608
FOCC-2	CFID	.7500
HAM	CFI	.0625
NAVC-1	CFI	(N/C)
NAVC-2	CFI	1
NAVC-3	CFI	.9534
NCDCF	CFI	.4000
NEL	CFI	.666
NMCSSC	CFI	0
RADC-1	CFID	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	.1666
RADC-4	CFI	.5555
SAC-1	CFI	.5952
SAC-2	CFI	.4615
SAC-3	CFI	.9556
SAC-4	CFI	.6113

N(CFI) = 19

N(CFID) = 3

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.4 AFC Reference: A 1.2.3

Feature Name: TABLE VARIABILITY ATTRIBUTE - VARIABLE

Resolution Inequality: $U GQ T1 = .10$

Resolution Equation: $1 - (Q86/Q87)$

Number of users with usage rates: $GQ T1 = 18$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.5264
ADPAC-2	CFI	.1250
ADPAC-3	CFI	0
CSC	CFI	N/A
FAA	CFI	.2
FOCC-1	CFID	.7392
FOCC-2	CFID	.2500
HAM	CFI	.9375
NAVC-1	CFI	N/C
NAVC-2	CFI	0
NAVC-3	CFI	.0466
NCDCF	CFI	.6000
NEL	CFI	.334
NMCSSC	CFI	1
RADC-1	CFID	N/C
RADC-2	CFI	.1
RADC-3	CFI	.8334
RADC-4	CFI	.4445
SAC-1	CFI	.4048
SAC-2	CFI	.5385
SAC-3	CFI	.0444
SAC-4	CFI	.3887

N(CFI) = 19

N(CFID) = 3

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.2.6 AFC Reference: A 1.2.3
 Feature Name: NENT
 Resolution Inequality: $U GQ T1 = .10$
 Resolution Equation: $U = Q169/Q78$
 Number of users with usage rates: $GQ T1 = 15$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.6315
ADPAC-2	CFI	.1250
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	1.304
FOCC-2	CFI	.2500
HAM	CFI	.9375
NAVC-1	CFI	(N/C)
NAVC-2	CFI	1
NAVC-3	CFI	.8139
NCD CF	CFI	1
NEL	CFI	.0166
NMCSSC	CFI	.6666
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	0
RADC-4	CFI	.0555
SAC-1	CFI	1.976
SAC-2	CFI	.3230
SAC-3	CFI	.8076
SAC-4	CFI	1.279

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.1.2

AFC Reference: A 1.2.4

Feature Name: NWDSN

Resolution Inequality: $U \text{ GQ T1} = .02$

Resolution Equation: $U = Q140/Q78$

Number of users with usage rates: $\text{GQ T1} = 0$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFID	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.0173
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFI	.0166
NMCSSC	CFNI	0
RADC-1	CFI	(N/C)
RADC-2	CFI	.1
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFI	.0119
SAC-2	CFI	.0115
SAC-3	CFI	.0087
SAC-4	CFI	.0010

N(CFI) = 19

N(CFID) = 1

N(CFNI) = 2

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.4

AFC Reference: A 1.2.5

Feature Name: STRUCTURE STRING

Resolution Inequality: U GQ T1 = .10

Resolution Equation: U = Q103/Q78

Number of users with usage rates: GQ T1 = 2

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFI	0
SAC-2	CFI	.0307
SAC-3	CFI	.2697
SAC-4	CFI	.0357

N(CFI) = 6

N(CFID) = 0

N(CFNI) = 16

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.2.1 AFC Reference: A 2.1
 Feature Name: BIT
 Resolution Inequality: $U \leq GQ T1 = .03$
 Resolution Equation: $U = Q149/Q14$
 Number of users with usage rates: $GQ T1 = 10$
 Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.0416
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.0200
FOCC-2	CFI	.0200
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.1000
NAVC-3	CFI	0
NCDCF	CFI	.0800
NEL	CFI	.0020
NMCSSC	CFI	.6666
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	.0100
RADC-4	CFI	.1635
SAC-1	CFID	.0966
SAC-2	CFID	.0210
SAC-3	CFID	.0689
SAC-4	CFID	.0649

N(CFI) = 18

N(CFID) = 4

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.2.2 AFC Reference: A 2.2
 Feature Name: BYTE
 Resolution Inequality: $U GQ T1 = .03$
 Resolution Equation: $U = Q153/(Q46 + Q50)$
 Number of users with usage rates: $GQ T1 = 15$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	2.883
ADPAC-2	CFI	2.428
ADPAC-3	CFI	1.291
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.3076
FOCC-2	CFI	.0150
HAM	CFI	4.000
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.1500
NAVC-3	CFI	0
NCDCF	CFI	.8571
NEL	CFI	.0294
NMCSSC	CFI	.6666
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	.7500
RADC-4	CFI	.6666
SAC-1	CFID	4.235
SAC-2	CFID	2.058
SAC-3	CFID	.9856
SAC-4	CFI	.9605

N(CFI) = 17

N(CFID) = 4

N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.2.3-4

AFC Reference: A 2.3

Feature Name: CHAR/MANT

Resolution Inequality: $U \text{ GQ T1} = .03$

Resolution Equation: $U = (Q157 + Q161)/Q38$

Number of users with usage rates: $\text{GQ T1} = 1$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.2000
ADPAC-3	CFI	0
CSC	CFNI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFI	.0200
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFI	0
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 7

N(CFID) = 0

N(CFNI) = 15

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.1 AFC Reference: A 2.4
 Feature Name: SUBSCRIBING - Expressed as Complex Numeric Formula
 Resolution Inequality: N/A
 Resolution Equation: U = Q144/(Q74 + Q78 + Q103)
 Number of users with usage rates: GQ T1 = N/A
 Resolution: Accepted as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.3684
ADPAC-2	CFI	.1250
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.1304
FOCC-2	CFI	.6250
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.2000
NAVC-3	CFI	.5813
NCDCF	CFI	.4000
NEL	CFI	.0750
NMCSSC	CFI	6.666
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	32.25
RADC-4	CFI	4.444
SAC-1	CFID	1.500
SAC-2	CFID	.5425
SAC-3	CFID	1.229
SAC-4	CFID	.8433

N(CFI) = 18 N(CFID) = 4 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.1

AFC Reference: A 2.4

Feature Name: SUBSCRIBING - Nested

Resolution Inequality: N/A

Resolution Equation: $U = Q145/(Q74 + Q78 + Q103)$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Accepted as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	.8000
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.0869
FOCC-2	CFI	1.250
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.0666
NAVC-3	CFI	.2325
NCDCE	CFI	1
NEL	CFI	.0050
NMCSSC	CFI	1.333
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	16.12
RADC-4	CFI	.1111
SAC-1	CFID	.3095
SAC-2	CFID	.2932
SAC-3	CFID	2.030
SAC-4	CFID	.5927

$N(\text{CFI}) = 18$

$N(\text{CFID}) = 4$

$N(\text{CFNI}) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.6 AFC Reference: A 2.5
 Feature Name: TABLE ENTRY REFERENCING (ENTRY/ENT)
 Resolution Inequality: $U \text{ GQ T1} = .02$
 Resolution Equation: $U = Q200/Q78$
 Number of users with usage rates: GQ T1 - 13
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFID	.6315
ADPAC-2	CFID	.2500
ADPAC-3	CFID	0
CSC	CFI	(N/A)
FAA	CFID	.8
FOCC-1	CFI	.0652
FOCC-2	CFI	2.500
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.3333
NAVC-3	CFI	0
NCDCF	CFI	.0400
NEL	CFI	.0200
NMCSSC	CFID	.6666
RADC-1	CFID	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFID	.6904
SAC-2	CFID	.2307
SAC-3	CFID	.4125
SAC-4	CFID	.2202

N(CFI) = 9

N(CFID) = 9

N(CFNI) = 3

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.1

AFC Reference: A 2,6

Feature Name:

Resolution Inequality: N/A

Resolution Equation: ENT: $U = Q201/Q200$; ENTRY: $U = 1 - Q201/Q200$

Number of users with usage rates: GQ T1 = N/A

Resolution: Retain ENTRY as nucleus; delete ENT (Basis of resolution: As shown below, the majority of applications implementing both ENT and ENTRY and which use table entry referencing at all, use ENTRY either exclusively or in most cases.)

Application	Compliance		Usage	
	ENT	ENTRY	ENT	ENTRY
ADPAC-1	CFNI	CFI	0	1
ADPAC-2	CFNI	CFI	0	1
ADPAC-3	CFNI	CFI	0	1
CSC	CFI	CFI	(N/A)	(N/A)
FAA	CFI	CFNI	1	0
FOCC-1	CFI	CFI	0	1
FOCC-2	CFI	CFI	0	1
HAM	CFI	CFI	0	0
NAVC-1	CFI	CFI	(N/C)	(N/C)
NAVC-2	CFI	CFI	0	1
NAVC-3	CFI	CFI	0	0
NCDCF	CFI	CFI	0	1
NEL	CFI	CFI	.8333	.1667
NMCSSC	CFNI	CFI	0	1
RADC-1	CFI	CFI	(N/C)	(N/C)
RADC-2	CFNI	CFNI	(N/C)	(N/C)
RADC-3	CFI	CFI	0	0
RADC-4	CFI	CFI	0	0
SAC-1	CFI	CFNI	1	0
SAC-2	CFI	CFNI	1	0
SAC-3	CFI	CFNI	1	0
SAC-4	CFI	CFNI	1	0

ENT:	N(CFI) = 17	N(CFID) = 0	N(CFNI) = 5
ENTRY:	N(CFI) = 16	N(CFID) = 0	N(CFNI) = 6

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.15 AFC Reference: A 3

Feature Name: COMPUTER REPRESENTATION OF DATA

Resolution Inequality: N/A

Resolution Equation: N/A

Number of users with usage rates: GQ T1 = N/A

Resolution: Modified as specified in 3.2.2.1 (hexadecimal constant), 3.2.2.3 (CHARCODE), 3.2.2.5 (computer representation of numeric variables and constants, and 3.2.2.10 (extended precision)

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFID	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFID	
NMCSSC	CFID	
RADC-1	CFID	
RADC-2	CFI	
RADC-3	CFI	
RADC-4	CFID	
SAC-1	CFID	
SAC-2	CFID	
SAC-3	CFID	
SAC-4	CFI	

N(CFI) = 14

N(CFID) = 8

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.1 AFC Reference: A 3.1
 Feature Name: OCTAL CONSTANT
 Resolution Inequality: $U \geq T1 = 1$
 Resolution Equation: $U = Q4$
 Number of users with usage rates: $GQ T1 = 18$

Resolution: The "Bit String Constant" Facility provided by Octal Constant is retained as nucleus; however, octal is optional if hexadecimal (3.2.2.1) is chosen for implementation.
 Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	48
ADPAC-2	CFI	60
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	70
FOCC-2	CFI	70
HAM	CFI	130
NAVC-1	CFI	(N/C)
NAVC-2	CFI	12
NAVC-3	CFI	1
NCD CF	CFI	30
NEL	CFI	1,000
NMCSSC	CFI	50
RADC-1	CFI	(N/C)
RADC-2	CFI	.7
RADC-3	CFI	30
RADC-4	CFI	25
SAC-1	CFID	520
SAC-2	CFID	5000
SAC-3	CFID	453
SAC-4	CFID	860

N(CFI) = 18 N(CFID) = 4 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.5 AFC Reference: A 3.3
 Feature Name: BASIC TABLE STRUCTURE ATTRIBUTE - PARALLEL
 Resolution Inequality: $U \leq T1 = .10$
 Resolution Equation: $U = Q89 / (Q89A + Q89)$
 Number of users with usage rates: $GQ T1 = 10$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.8157
ADPAC-2	CFI	.0625
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFID	0
FOCC-1	CFI	0
FOCC-2	CFI	.1428
HAM	CFI	1
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	.9767
NCDCF	CFI	0
NEL	CFI	.8000
NMCSSC	CFI	.3333
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	.5000
RADC-4	CFI	.9090
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFID	0

N(CFI) = 17 N(CFID) = 2 N(CFNI) = 3

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.5 AFC Reference: A 3.3
 Feature Name: BASIC TABLE STRUCTURE ATTRIBUTE - SERIAL
 Resolution Inequality: $U G Q = .10$
 Resolution Equation: $U = Q89A / (Q89A + Q89)$
 Number of users with usage rates: $G Q T1 = 16$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.1842
ADPAC-2	CFI	.9375
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFID	.9
FOCC-1	CFI	1
FOCC-2	CFI	.8572
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	1
NAVC-3	CFI	.0232
NCDCF	CFI	1
NEL	CFI	.2000
NMCSSC	CFI	.6666
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	.5000
RADC-4	CFI	.9090
SAC-1	CFI	1
SAC-2	CFI	1
SAC-3	CFI	1
SAC-4	CFID	1

$N(\text{CFI}) = 20$ $N(\text{CFID}) = 2$ $N(\text{CFNI}) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.6.1 AFC Reference: A 3.4
Feature Name: ORDINARY PACKING - Medium and/or Dense
Resolution Inequality: $U GQ T1 = .02$
Resolution Equation: $U = (Q95 + Q95A) +/Q78$
Number of users with usage rates: GQ T1 = 14
Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.0625
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFID	0
FOCC-2	CFID	0
HAM	CFI	0
NAVC-1	CFID	(N/C)
NAVC-2	CFID	0
NAVC-3	CFID	0
NCDCF	CFID	0
NEL	CFID	.3333
NMCSSC	CFID	0
RADC-1	CFI	(N/C)
RADC-2	CFID	.2222
RADC-3	CFI	.1666
RADC-4	CFID	.0222
SAC-1	CFID	(N/C)
SAC-2	CFID	(N/C)
SAC-3	CFID	(N/C)
SAC-4	CFID	0

N(CFI) = 8

N(CFID) = 14

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.6.1

AFC Reference: A 3.4

Feature Name: ORDINARY PACKING - Medium

Resolution Inequality: $U GQ T1 = .05$

Resolution Equation: $U = Q95/(Q95 + Q95A)$

Number of users with usage rates: $GQ T1 = 0$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	0
RADC-4	CFNI	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

N(CFI) = 8

N(CFID) = 14

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.6.1 AFC Reference: A 3.4
 Feature Name: ORDINARY PACKING - Dense
 Resolution Inequality: U GQ T1 = .05
 Resolution Equation: U = Q95A/(Q95 + Q95A)
 Number of users with usage rates: GQ T1 = 5
 Resolution: Retain as optional (must be implemented if Ordinary Packing
 Data: is implemented)

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	1
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFI	1
NMCSSC	CFI	0
RADC-1	CFI	(N/C)
RADC-2	CFI	1
RADC-3	CFI	1
RADC-4	CFI	1
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

N(CFI) = 8

N(CFID) = 14

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.6.1 AFC Reference: A 3.4
 Feature Name: NO" PACKING NOTATION - "N"
 Resolution Inequality: $U G Q T1 = .05$
 Resolution Equation: $U = Q94 / (Q93 + Q94)$
 Number of users with usage rates: $G Q T1 = 1$
 Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFID	0
FOCC-2	CFID	0
HAM	CFI	0
NAVC-1	CFID	(N/C)
NAVC-2	CFID	0
NAVC-3	CFID	0
NCDCF	CFID	0
NEL	CFID	0
NMCSSC	CFID	0
RADC-1	CFI	(N/C)
RADC-2	CFID	.818
RADC-3	CFI	1
RADC-4	CFID	.027
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

N(CFI) = 8 N(CFID) = 14 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.3.6.2 AFC Reference: A 3.5
Feature Name: DEFINED PACKING
Resolution Inequality: U GQ T1 = .10
Resolution Equation: U = Q99/Q78
Number of users with usage rates: GQ T1 = 17
Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.4473
ADPAC-2	CFI	.9375
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	1
FOCC-2	CFI	1
HAM	CFI	.0625
NAVC-1	CFI	(N/C)
NAVC-2	CFI	1
NAVC-3	CFI	.9534
NCDCF	CFI	.6000
NEL	CFI	.1666
NMCSSC	CFI	.4666
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	0
RADC-4	CFI	.222
SAC-1	CFID	.3095
SAC-2	CFID	.5384
SAC-3	CFID	.7423
SAC-4	CFID	.7875

N(CFI) = 18

N(CFID) = 4

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.7.1 AFC Reference: A 3.6

Feature Name: INDEPENDENT OVERLAY DECLARATION

Resolution Inequality: $U \leq GQ \leq T1 = 1$

Resolution Equation: $U = Q129$

Number of users with usage rates: $GQ \leq T1 = 15$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	19
ADPAC-2	CFI	7
ADPAC-3	CFI	0
CSC	CFID	(N/A)
FAA	CFNI	0
FOCC-1	CFI	30
FOCC-2	CFI	10
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	2
NAVC-3	CFI	0
NCDCF	CFI	5
NEL	CFI	30
NMCSSC	CFI	25
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	3
RADC-4	CFI	25
SAC-1	CFID	10
SAC-2	CFID	250
SAC-3	CFID	77
SAC-4	CFID	1370

$N(\text{CFI}) = 16$

$N(\text{CFID}) = 5$

$N(\text{CFNI}) = 1$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.7.2 AFC Reference: A 3.6
 Feature Name: SUBORDINATE OVERLAY DECLARATION
 Resolution Inequality: $U \text{ GQ T1} = 1$
 Resolution Equation: $U = Q133$
 Number of users with usage rates: $\text{GQ T1} = 0$
 Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	4
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFID	10
SAC-2	CFID	20
SAC-3	CFID	10
SAC-4	CFID	10

N(CFI) = 1

N(CFID) = 4

N(CFNI) = 17

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.1.1

AFC Reference: A 3.8

Feature Name: 'LOC

Resolution Inequality: U GQ T1 = 1

Resolution Equation: U = Q136

Number of users with usage rates: GQ T1 = 4

Resolution: Retain as nucleus*

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFID	(N/A)
FAA	CFNI	0
FOCC-1	CFI	5
FOCC-2	CFI	3
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	3
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFI	73
NMCSSC	CFNI	0
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFID	0

N(CFI) = 9 N(CFID) = 2 N(CFNI) = 11

*These data indicate that 'LOC be retained as optional. Responses to Interview Question I-15 (2) indicate nucleus, however.

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.3.1 AFC Reference: A 4
 Feature Name: PARENTHEZIZED NUMERIC FORMULAS
 Resolution Inequality: N/A
 Resolution Equation: U = Q173
 Number of users with usage rates: GQ T1 = N/A
 Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	14
ADPAC-2	CFI	15
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	30
FOCC-2	CFI	100
HAM	CFI	230
NAVC-1	CFI	(N/C)
NAVC-2	CFI	20
NAVC-3	CFI	50
NCDCF	CFI	20
NEL	CFI	10
NMCSSC	CFI	1000
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	500
RADC-4	CFI	20
SAC-1	CFI	240
SAC-2	CFI	2700
SAC-3	CFI	968
SAC-4	CFI	2290

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.3.5 AFC Reference: A 4
 Feature Name: MIXED ITEM TYPES IN NUMERIC FORMULAS
 Resolution Inequality: N/A
 Resolution Equation: $U = Q189$
 Number of users with usage rates: GQ T1 = N/A
 Resolution: Accept as nucleus
 Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	20
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	1 or more
FOCC-1	CFI	150
FOCC-2	CFI	50
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	5
NAVC-3	CFI	0
NCDCF	CFID	10
NEL	CFI	150
NMCSSC	CFI	100
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	20
RADC-4	CFI	10
SAC-1	CFI	100
SAC-2	CFI	4500
SAC-3	CFI	0
SAC-4	CFI	80

$N(\text{CFI}) = 21$

 $N(\text{CFID}) = 1$

 $N(\text{CFNI}) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.4

AFC Reference: A 4

Feature Name: ITEM TYPE INTEGER

Resolution Inequality: N/A

Resolution Equation: $U = Q18 / (Q18 + Q22 + Q38)$

Number of users with usage rates: GQ T1 = 15

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFID	.7525
ADPAC-2	CFID	.1428
ADPAC-3	CFID	.4772
CSC	CFI	(N/A)
FAA	CFI	.9
FOCC-1	CFI	.0963
FOCC-2	CFI	.1428
HAM	CFI	.0303
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.2857
NAVC-3	CFI	.1000
NCDCF	CFI	.8000
NEL	CFI	.5000
NMCSSC	CFI	.0476
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	.9937
RADC-4	CFI	.7142
SAC-1	CFID	.9539
SAC-2	CFID	.8791
SAC-3	CFID	.9649
SAC-4	CFID	.9570

N(CFI) = 14

N(CFID) = 7

N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.5

AFC Reference: A 4

Feature Name: ITEM TYPE FIXED POINT - Scale in Declaration Blank or Zero

Resolution Inequality: $U GQ T1 = .02$

Resolution Equation: $U = Q23/Q22$

Number of users with usage rates: $GQ T1 = 16$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	(N/C)
ADPAC-2	CFI	.7692
ADPAC-3	CFI	.6521
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	1
FOCC-2	CFI	0
HAM	CFI	1.166
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	1
NCDCF	CFI	0
NEL	CFI	.0285
NMCSSC	CFI	.6666
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	1
RADC-4	CFI	.4166
SAC-1	CFI	0
SAC-2	CFI	1.156
SAC-3	CFI	.1666
SAC-4	CFI	0

$N(CFI) = 16$

$N(CFID) = 0$

$N(CFNI) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.8
 Feature Name: RANGE ATTRIBUTE
 Resolution Inequality: $U \leq Q_{T1} = .10$
 Resolution Equation: $U = Q_{34} / (Q_{18} + Q_{22})$
 Number of users with usage rates: $Q_{T1} = 0$
 Resolution: Delete

AFC Reference: A 4.3

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFID	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(\text{CFI}) = 2$

$N(\text{CFID}) = 1$

$N(\text{CFNi}) = 19$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.9

AFC Reference: A 4.4

Feature Name: ITEM TYPE FLOATING POINT

Resolution Inequality: N/A

Resolution Equation: $U = Q38 / (Q18 + Q22 + Q38)$

Number of users with usage rates: GQ T1 = N/A

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.2380
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.9
FOCC-1	CFI	.1204
FOCC-2	CFI	0
HAM	CFI	.0606
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	.3000
NCDCF	CFI	.0400
NEL	CFI	.1500
NMCSSC	CFI	.3809
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFI	.0276
SAC-2	CFI	.0915
SAC-3	CFI	.0194
SAC-4	CFI	.0245

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.6 AFC Reference: A 4.4
 Feature Name: SIGN ATTRIBUTE
 Resolution Inequality: U GE T1 = .01
 Resolution Equation: U = Q26/(Q18 + Q22)
 Number of users with usage rates: GQ T1 = 14
 Resolution: Retain as nucleus (Note: Usage data refers to "signed" items)
 Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.04
ADPAC-2	CFI	.03
ADPAC-3	CFI	.05
CSC	CFI	(N/A)
FAA	CFI	.02
FOCC-1	CFI	1.095
FOCC-2	CFI	.04
HAM	CFI	.004
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	1
NCDCF	CFI	.008
NEL	CFI	.03
NMCSSC	CFI	.01
RADC-1	CFI	(N/C)
RADC-2	CFI	.03
RADC-3	CFI	.09
RADC-4	CFI	.01
SAC-1	CFI	.02
SAC-2	CFI	.008
SAC-3	CFI	.003
SAC-4	CFI	.02

N(CFI) =	22	N(CFID) =	0	N(CFNI) =	0
----------	----	-----------	---	-----------	---

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.3.2 AFC Reference: A 4.5

Feature Name: PREFIX + AND -

Resolution Inequality: U GQ T1 = 1

Resolution Equation: U = Q177

Number of users with usage rates: GQ T1 = 13

Resolution: Retain as nucleus (Note modification of precedence (3.2.2.6))

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	2
CSC	CFI	(N/A)
FAA	*	(N/C)
FOCC-1	CFI	30
FOCC-2	CFI	0
HAM	CFI	55
NAVC-1	CFI	(N/C)
NAVC-2	CFI	20
NAVC-3	CFI	15
NCD CF	CFI	5
NEL	CFI	3
NMCSSC	CFID	10
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	0
RADC-4	CFI	20
SAC-1	CFID	140
SAC-2	CFID	9000
SAC-3	CFID	64
SAC-4	CFID	1300
	N(CFI) = 15	N(CFID) = 5
		N(CFNI) = 1

*Misprinted page in JAQ made it impossible for FAA to respond.

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.3.3

AFC Reference: A 4.5

Feature Name: EXPONENTIATION OPERATOR

Resolution Inequality: N/A

Resolution Equation: N/A

Number of users with usage rates: GQ TI = N/A

Resolution: Accept as Nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFID	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFID	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFI	
NMCSSC	CFID	
RADC-1	CFI	
RADC-2	CFID	
RADC-3	CFI	
RADC-4	CFI	
SAC-1	CFID	
SAC-2	CFID	
SAC-3	CFID	
SAC-4	CFID	

N(CFI) = 14

N(CFID) = 8

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J3.5.2.3.3

AFC Reference: A 4.6

Feature Name: EXPONENTIATION NOTATION - (* *)

Resolution Inequality: U G Q T1 = 1

Resolution Equaltion: U - Q181

Number of users with usage rates: G Q T1 = 10

Resolution: Delete

Data:

Note: Because of an oversight in the preparation of the JAQ, insufficient information regarding the use of exponentiation operators was obtained. We are recommending that (* *) be dropped, since ** is widely used in other high level languages, and since only one operator is required.

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	3
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	1 or more
FOCC-1	CFI	5
FOCC-2	CFI	0
HAM	CFNI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	5
NAVC-3	CFI	0
NCDCF	CFI	2
NEL	CFI	20
NMCSSC	CFNI	0
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	5
RADC-4	CFI	0
SAC-1	CFI	20
SAC-2	CFI	225
SAC-3	CFI	15
SAC-4	CFI	20

N(CFI) = 19

N(CFID) = 0

N(CFNI) = 3

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.3.4

AFC Reference: A 4.7

Feature Name: Absolute Value Operator

Resolution Inequality: N/A

Resolution Equation: N/A

Number of users with usage rates: GQ T1 = N/A

Resolution: Retain as optional

Data:

Application

Compliance

Usage

ADPAC-1	CFID
ADPAC-2	CFID
ADPAC-3	CFID
CSC	CFI
FAA	CFID
FOCC-1	CFI
FOCC-2	CFI
HAM	CFI
NAVC-1	CFI
NAVC-2	CFI
NAVC-3	CFI
NCDCF	CFI
NEL	CFI
NMCSSC	CFID
RADC-1	CFI
RADC-2	CFID
RADC-3	CFI
RADC-4	CFI
SAC-1	CFID
SAC-2	CFID
SAC-3	CFID
SAC-4	CFID

Note: Owing to an oversight in the preparation of the JAQ, insufficient data regarding use of absolute value operator were solicited. In the interests of being fair, we have declared this feature as optional.

N(CFI) = 12

N(CFID) = 10

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.3.4

AFC Reference: A 4.8

Feature Name: ABSOLUTE VALUE NOTATION - (✓ /)

Resolution Inequality: N/A

Resolution Equation: Q185

Number of users with usage rates: GE T1 = N/A

Resolution: Delete Basis of Resolution: Only one notation for absolute value is required. The number of application which do not implement (✓ /) or which have 0 usage is 15. Therefore (✓ /) is dropped in favor of ABS.

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	5
NAVC-3	CFI	2
NCDCF	CFI	0
NEL	CFI	0
NMCSSC	CFI	20
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 14

N(CFID) = 0

N(CFNI) = 8

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.9

AFC Reference: A 4.9

Feature Name: REM

Resolution Inequality: $U G Q T1 = .05$

Resolution Equation: $U = Q265/Q18$

Number of users with usage rates: $G Q T1 = 0$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 1 N(CFID) = 0 N(CFNI) = 21

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.7

AFC Reference: A 4.9

Feature Name: REMQUO

Resolution Inequality: $U G Q T1 = .05$

Resolution Equation: $U = Q257/Q18$

Number of users with usage rates: $G Q T1 = 7$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.001
ADPAC-2	CFI	0
ADPAC-3	CFI	.009
CSC	CFI	(N/A)
FAA	CFI	.08
FOCC-1	CFID	0
FOCC-2	CFID	0
HAM	CFI	.05
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	.01
RADC-4	CFNI	0
SAC-1	CFID	.001
SAC-2	CFID	.003
SAC-3	CFID	.001
SAC-4	CFID	.001

N(CFI) = 9

N(CFID) = 6

N(CFNI) = 7

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.7

AFC Reference: A 5

Feature Name: ITEM TYPE DUAL

Resolution Inequality: $U \leq Q \leq T1 = .10$

Resolution Equation: $U = Q30/Q14$

Number of users with usage rates: $G \leq Q \leq T1 = 0$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFNI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 1

N(CFID) = 0

N(CFNI) = 21

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.4

AFC Reference: A5

Feature Name: DUAL FORMULAS

Resolution Inequality: $U GQ T1 = 0$

Resolution Equation: $U = Q193/Q14$

Number of users with usage rates: $GQ T1 = T2 = 0$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFNI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 0 N(CFID) = 0 N(CFNI) = 22

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.7

AFC Reference: A 6.1

Feature Name: RELATIONAL FORMULAS (excluding "Chained")

Resolution Inequality: N/A

Resolution Equation: N/A

Number of users with usage rates: GQ T1 = N/A

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCD CF	CFI	
NEL	CFID	
NMCSSC	CFID	
RADC-1	CFI	
RADC-2	CFID	
RADC-3	CFI	
RADC-4	CFI	
SAC-1	CFID	
SAC-2	CFID	
SAC-3	CFID	
SAC-4	CFID	

N(CFI) = 15

N(CFID) = 7

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.7

AFC Reference: A 6.2

Feature Name: "CHAINED" RELATIONAL FORMULAS

Resolution Inequality: $U \text{ GQ } 77 = 1$

Resolution Equation: $U = \text{Q205}$

Number of users with usage rates: $\text{GQ T1} = 9$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	37
ADPAC-2	CFI	15
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	1 or more
FOCC-1	CFI	10
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	5
NAVC-3	CFI	0
NCDCF	CFI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	10
RADC-4	CFI	0
SAC-1	CFI	100
SAC-2	CFI	700
SAC-3	CFI	540
SAC-4	CFI	600

$N(\text{CFI}) = 19$

$N(\text{CFID}) = 0$

$N(\text{CFNI}) = 3$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.8

AFC Reference: A 7.1

Feature Name: BOOLEAN FORMULAS -- with AND, OR, NOT

Resolution Inequality: $U \text{ GQ } T1 = 1$

Resolution Equation: $U = Q209$

Number of users with usage rates: $\text{GQ } T1 = 15$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	18
ADPAC-2	CFI	15
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	500
FOCC-2	CFI	15
HAM	CFI	300
NAVC-1	CFI	(N/C)
NAVC-2	CFI	10
NAVC-3	CFI	20
NCDCF	CFI	20
NEL	CFI	30
NMCSSC	CFI	700
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	400
RADC-4	CFI	0
SAC-1	CFI	1480
SAC-2	CFI	7000
SAC-3	CFI	1700
SAC-4	CFI	8050

$N(\text{CFI}) = 21$

$N(\text{CFID}) = 0$

$N(\text{CFNI}) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.12

AFC Reference: A 7.2

Feature Name: ITEM TYPE BOOLEAN

Resolution Inequality: $U \text{ GQ T1} = .02$

Resolution Equation: $U = Q54/Q14$

Number of users with usage rates: $\text{GQ T1} = 8$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.0500
ADPAC-2	CFI	.1083
ADPAC-3	CFI	.2380
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	.0480
FOCC-2	CFI	0
HAM	CFNI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.0500
NAVC-3	CFI	.0050
NCDCF	CFI	.2000
NEL	CFID	.0100
NMCSSC	CFI	.3333
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 15

N(CFID) = 1

N(CFNI) = 6

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.1

AFC Reference: A 8

Feature Name: NUMERIC ITEM TYPE CONVERSION IN THE ASSIGNMENT STATEMENT

Resolution Inequality: $U \leq GQ \leq T1 = .01$

Resolution Equation: $U = Q213/Q217A$

Number of users with usage rates: $GQ \leq T1 = 12$

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	(N/C)
ADPAC-2	CFI	.0428
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFID	.2
FOCC-1	CFI	.0125
FOCC-2	CFI	.0500
HAM	CFID	.0600
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.6000
NAVC-3	CFI	0
NCDCF	CFI	.0500
NEL	CFI	.0350
NMCSSC	CFI	.2000
RADC-1	CFI	(N/C)
RADC-2	CFID	.2
RADC-3	CFI	.0083
RADC-4	CFI	.0250
SAC-1	CFID	.0018
SAC-2	CFID	.0700
SAC-3	CFID	.0014
SAC-4	CFID	.0023

N(CFI) = 15

N(CFID) = 7

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.1

AFC Reference: A 8

Feature Name: BOOLEAN ASSIGNMENT STATEMENT

Resolution Inequality: N/A

Resolution Equation: $U = Q217/Q217A$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	(N/C)
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFID	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFID	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFI	0125
NEL	CFI	.0025
NMCSSC	CFI	.2000
RADC-1	CFI	(N/C)
RADC-2	CFID	0
RADC-3	CFID	0
RADC-4	CFID	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

$N(CFI) = 14$

$N(CFID) = 7$

$N(CFNI) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.6.1

AFC Reference: A 8.1

Feature Name: ITEM PRESET

Resolution Inequality: $U \text{ GQ T1} = .02$

Resolution Equation: $U = Q113/Q14$

Number of users with usage rates: $\text{GQ T1} = 13$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.1714
ADPAC-2	CFI	.0416
ADPAC-3	CFI	.3809
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	.0720
FOCC-2	CFI	.6000
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.1000
NAVC-3	CFI	.0650
NCDCF	CFI	.0400
NEL	CFI	.0400
NMCSSC	CFI	1.333
RADC-1	CFI	(N/C)
RADC-2	CFI	2
RADC-3	CFI	.1500
RADC-4	CFI	.3738
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(\text{CFI}) = 17$

$N(\text{CFID}) = 0$

$N(\text{CFNI}) = 5$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.6.2

AFC Reference: A 8.1

Feature Name: TABLE PRESET

Resolution Inequality: $U G Q T = .02$

Resolution Equation: $U = Q116/Q78$

Number of users with usage rates: $G Q T1 = 16$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.1315
ADPAC-2	CFI	.1875
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	.4347
FOCC-2	CFI	.5000
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.3333
NAVC-3	CFI	0
NCD CF	CFI	.1000
NEL	CFI	.3333
NMCSSC	CFI	.2000
RADC-1	CFI	(N/C)
RADC-2	CFID	.2
RADC-3	CFI	2.000
RADC-4	CFI	.0777
SAC-1	CFID	.3333
SAC-2	CFID	1
SAC-3	CFID	.1061
SAC-4	CFI	.1088

N(CFI) = 18 N(CFID) = 4 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.6.3

AFC Reference: A 8.1

Feature Name: ARRAY PRESET

Resolution Inequality: $U \text{ GQ T1} = .02$

Resolution Equation: $U = Q120/Q74$

Number of users with usage rates: $\text{GQ T1} = 2$

Resolution: Retain as option -- Must be implemented if ARRAY is implemented

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	(N/C)
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFI	.3000
NMCSSC	CFNI	0
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	1.000
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 7 N(CFID) = 0 N(CFNI) = 14

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.6.4

AFC Reference: A 8.1

Feature Name: STRING PRESET

Resolution Inequality: $U GQ T1 = .02$

Resolution Equation: $U = Q124/Q103$

Number of users with usage rates: $GQ T1 = 0$

Resolution: Delete

Data: Note: STRING is deleted.

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	(N/C)
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 1 N(CFID) = 0 N(CFNI) = 21

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.10

AFC Reference: A 8.2

Feature Name: ROUNDED NUMERIC ASSIGNMENT -- Round Attribute

Resolution Inequality: $U G Q T1 = .05$

Resolution Equation: $U = Q42 / (Q18 + Q22 + Q38)$

Number of users with usage rates: $G Q T1 = 0$

Resolution: Delete Note: Extended Numeric Round (3.2.2.14) is made optional.

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFNI	0
NEL	CFNI	0
NMCSSC	CFNI	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 2

N(CFID) = 0

N(CFNI) = 20

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.2

AFC Reference: A 8.3

Feature Name: EXCHANGE STATEMENT

Resolution Inequality: $U \leq GQ \leq T1 = .01$

Resolution Equation: $U = Q221/Q217A$

Number of users with usage rates: $GQ \leq T1 = 3$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	(N/C)
ADPAC-2	CFI	.0142
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	.0012
FOCC-2	CFI	.0100
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	(N/C)
NCDCF	CFI	.0025
NEL	CFI	.0035
NMCSSC	CFI	.1000
RADC-1	CFI	(N/C)
RADC-2	CFI	.1
RADC-3	CFI	0
RADC-4	CFI	.0150
SAC-1	CFI	.0030
SAC-2	CFI	.0240
SAC-3	CFI	.0158
SAC-4	CFI	.0103

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.2

AFC Reference: A 9.1

Feature Name: COMPUND STATEMENT

Resolution Inequality: N/A

Resolution Equation: $U = Q234$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	100
ADPAC-2	CFI	55
ADPAC-3	CFI	30
CSC	CFI	(N/A)
FAA	CFI	1 or more
FOCC-1	CFI	2000
FOCC-2	CFI	200
HAM	CFI	750
NAVC-1	CFI	(N/C)
NAVC-2	CFI	30
NAVC-3	CFI	0
NCDCF	CFI	40
NEL	CFI	200
NMCSSC	CFI	700
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	500
RADC-4	CFI	100
SAC-1	CFI	9360
SAC-2	CFI	11,000
SAC-3	CFI	4140
SAC-4	CFI	17,500

N(CFI) = 22 N(CFID) = 0 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5

AFC Reference: A 9.2

Feature Name: GOTO STATEMENT

Resolution Inequality: N/A

Resolution Equation: $U = Q276$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Accept as nucleus Note: The form containing a subscript is implemented only if Index Switch is implemented.

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	420
ADPAC-3	CFI	33
CSC	CFI	(N/A)
FAA	CFI	1 or more
FOCC-1	CFI	0
FOCC-2	CFI	100
HAM	CFI	250
NAVC-1	CFI	(N/C)
NAVC-2	CFI	100
NAVC-3	CFI	250
NDCDF	CFI	60
NEL	CFI	400
NMCSSC	CFI	200
RADC-1	CFI	(N/C)
RADC-2	CFI	1 or more
RADC-3	CFI	1000
RADC-4	CFI	40
SAC-1	CFI	10080
SAC-2	CFI	28,000
SAC-3	CFI	5708
SAC-4	CFI	7120

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.7

AFC Reference: A 9.3

Feature Name: STOP

Resolution Inequality: $U \text{ GQ T1} = 1$

Resolution Equation: $U = \text{Q319}$

Number of users with usage rates: $\text{GQ T1} = 9$

Resolution: Retain as nucleus

Note: Modification in 3.2.2.12
Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	11
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFID	(N/A)
FAA	CFID	1 or more
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	7
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFID	15
NMCSSC	CFI	10
RADC-1	CFID	(N/C)
RADC-2	CFNI	0
RADC-3	CFID	1
RADC-4	CFID	0
SAC-1	CFID	220
SAC-2	CFID	10
SAC-3	CFID	160
SAC-4	CFI	80

N(CFI) = 11

N(CFID) = 9

N(CFNI) = 2

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.7

AFC Reference: A 9.3

Feature Name: STOP Statement Label (Pausing)

Resolution Inequality: $U \text{ GQ T1} = 1$

Resolution Equation: $U = \text{Q318}$

Number of users with usage rates: $\text{GQ T1} = 1$

Resolution: Delete

Note: Pause statement in 3.2.2.13

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFID	(N/A)
FAA	CFID	1 or more
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFID	5
NMCSSC	CFI	0
RADC-1	CFID	(N/C)
RADC-2	CFNI	0
RADC-3	CFID	0
RADC-4	CFID	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFI	0

N(CFI) = 11

N(CFID) = 9

N(CFNI) = 2

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.3

AFC Reference: A 9.4.1

Feature Name: IF STATEMENT

Resolution Inequality: N/A

Resolution Equation: $U = Q291 / (Q291 + Q295 + Q284 + Q280)$

Number of users with usage rates: GQ T1 = N/A

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.9742
ADPAC-2	CFI	.9090
ADPAC-3	CFI	.9756
CSC	CFI	(N/A)
FAA	CFI	.9
FOCC-1	CFI	.9615
FOCC-2	CFI	.8196
HAM	CFI	.9615
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.9708
NAVC-3	CFI	.9708
NCDCF	CFI	.6250
NEL	CFI	.9523
NMCSSC	CFI	.7407
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	.9685
RADC-4	CFI	1
SAC-1	CFI	.9361
SAC-2	CFI	.9529
SAC-3	CFI	.9492
SAC-4	CFI	.9906

N(CFI) = 22

N(CFID) = 0

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.4

AFC Reference: A 9.4.2

Feature Name: IFEITH/ORIF STATEMENT

Resolution Inequality: $U \leq GQ T1 = .02$

Resolution Equation: $U = Q295 / (Q291 + Q295 + Q284 + Q280)$

Number of users with usage rates: $GQ T1 = 7$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.0103
ADPAC-2	CFI	.0363
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	.0384
FOCC-2	CFI	.8196
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.0097
NAVC-3	CFI	.0291
NCDCF	CFI	.1250
NEL	CFI	.0068
NMCSSC	CFI	.1111
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	.0169
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 17

N(CFID) = 0

N(CFNI) = 5

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.2

AFC Reference: A 9.4.3

Feature Name: INDEX SWITCHES

Resolution Inequality: $U \text{ GQ T1} = .02$

Resolution Equation: $U = Q284 / (Q291 + Q295 + Q284 + Q280)$

Number of users with usage rates: $\text{GQ T1} = 6$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.0181
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	0
FOCC-2	CFI	.1639
HAM	CFI	.0192
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.0097
NAVC-3	CFI	0
NCDCF	CFI	.1250
NEL	CFI	.0272
NMCSSC	CFI	.0740
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	.0096
RADC-4	CFI	0
SAC-1	CFID	.0036
SAC-2	CFID	.0071
SAC-3	CFID	.0187
SAC-4	CFID	.0025

N(CFI) = 18

N(CFID) = 4

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.2

AFC Reference: A 9.4.3

Feature Name: SWITCH NAMES within SWITCH DECLARATIONS

Resolution Inequality: $U \leq Q T1 = .05$

Resolution Equation: $U = Q285/Q284$

Number of users with usage rates: $G Q T1 = 1$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFI	0
NEL	CFI	0
NMCSSC	CFI	0
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

N(CFI) = 18 N(CFID) = 4 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.2

AFC Reference: A 9.4.3

Feature Name: CLOSE NAMES within Switch Declarations

Resolution Inequality: $U GQ T1 = .05$

Resolution Equation: $U = Q286/Q284$

Number of users with usage rates: $GQ T1 = 0$

Resolution: Delete

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFI	0
NEL	CFI	0
NMCSSC	CFI	0
RADC-1	CFI	(N/C)
RADC-2	CFI	0
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

N(CFI) = 18 N(CFID) = 4 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.1

AFC Reference: A 9.4.4

Feature Name: ITEM SWITCH

Resolution Inequality: $U \leq Q_{T1} = .01$

Resolution Equation: $U = Q_{280} / (Q_{291} + Q_{295} + Q_{284} + Q_{280})$

Number of users with usage rates: $Q_{T1} = 7$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.5309
ADPAC-2	CFI	.0363
ADPAC-3	CFI	.0243
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	0
FOCC-2	CFI	.8196
HAM	CFI	.0192
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.0097
NAVC-3	CFI	0
NCDCF	CFI	.1250
NEL	CFI	.0136
NMCSSC	CFI	.0740
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	.0048
RADC-4	CFI	0
SAC-1	CFID	.0602
SAC-2	CFID	.0399
SAC-3	CFI	.0319
SAC-4	CFI	.0067

N(CFI) = 18

N(CFID) = 4

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.5

AFC Reference: A 9.5

Feature Name: FOR LOOP - One Factor

Resolution Inequality: $U \leq GQ \leq T1 = .05$

Resolution Equation: $U = Q299 / (Q74 + Q78 + Q103)$

Number of users with usage rates: $GQ \leq T1 = 15$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	2.289
ADPAC-2	CFI	.0625
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	.1739
FOCC-2	CFI	.2500
HAM	CFI	.0625
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.1333
NAVC-3	CFI	0
NCDCF	CFI	0
NEL	CFI	.0750
NMCSSC	CFI	.4000
RADC-1	CFID	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	12.90
RADC-4	CFI	.0222
SAC-1	CFI	2.976
SAC-2	CFI	.2492
SAC-3	CFI	.4751
SAC-4	CFI	.4578

N(CFI) = 21

N(CFID) = 1

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.5 AFC Reference: A 9.5
 Feature Name: FOR LOOP - Two Factor
 Resolution Inequality: $U \leq GQ \leq T1 = .05$
 Resolution Equation: $U = Q300 / (Q74 + Q78 + Q103)$
 Number of users with usage rates: $GQ \leq T1 = 17$
 Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>	
ADPAC-1	CFI	.0789	
ADPAC-2	CFI	.1875	
ADPAC-3	CFI	.6000	
CSC	CFI	(N/A)	
FAA	CFI	.2	
FOCC-1	CFI	.6521	
FOCC-2	CFI	.5000	
HAM	CFI	.3125	
NAVC-1	CFI	(N/C)	
NAVC-2	CFI	.0666	
NAVC-3	CFI	0	
NCDCF	CFI	.0400	
NEL	CFI	.1000	
NMCSSC	CFI	.2000	
RADC-1	CFID	(N/C)	
RADC-2	CFI	.7	
RADC-3	CFI	1.129	
RADC-4	CFI	.1666	
SAC-1	CFI	.1904	
SAC-2	CFI	.2419	
SAC-3	CFI	.7408	
SAC-4	CFI	.3421	
	N(CFI) = 21	N(CFID) = 1	N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.5

AFC Reference: A 9.5

Feature Name: FOR LOOP - Three Factor

Resolution Inequality: $U \leq Q_{T1} = .05$

Resolution Equation: $U = Q_{301} / (Q_{74} + Q_{78} + Q_{103})$

Number of users with usage rates: $Q_{T1} = 17$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.6250
ADPAC-3	CFI	1.600
CSC	CFI	(N/A)
FAA	CFI	.9
FOCC-1	CFI	3.478
FOCC-2	CFI	1.250
HAM	CFI	3.125
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.6666
NAVC-3	CFI	0
NCDCF	CFI	.4000
NEL	CFI	.1750
NMCSSC	CFI	2.666
RADC-1	CFID	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	6.451
RADC-4	CFI	1.222
SAC-1	CFI	2.761
SAC-2	CFI	.8211
SAC-3	CFI	1.754
SAC-4	CFI	3.026

$N(\text{CFI}) = 21$

$N(\text{CFID}) = 1$

$N(\text{CFNI}) = 0$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.5

AFC Reference: A 9.5

Feature Name: FOR LOOP - Decrementing

Resolution Inequality: $U \text{ GQ T1} = .05$

Resolution Equation: $U = Q302/Q301$

Number of users with usage rates: $\text{GQ T1} = 15$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	(N/C)
ADPAC-2	CFI	.1000
ADPAC-3	CFI	.2500
CSC	CFI	(N/A)
FAA	CFI	.2
FOCC-1	CFI	.0625
FOCC-2	CFI	.6250
HAM	CFI	.0200
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.2000
NAVC-3	CFI	0
NCDCF	CFI	.2500
NEL	CFI	.1428
NMCSSC	CFI	.5000
RADC-1	CFID	(N/C)
RADC-2	CFI	.7
RADC-3	CFI	.0500
RADC-4	CFI	0
SAC-1	CFI	.3809
SAC-2	CFI	.0892
SAC-3	CFI	.2405
SAC-4	CFI	.1528

N(CFI) = 21

N(CFID) = 1

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.5.6

AFC Reference: A 9.5.4

Feature Name: TEST STATEMENT

Resolution Inequality: $U \leq GQ \leq T1 = .05$

Resolution Equation: $U = Q310 / (Q299 + Q300 + Q301)$

Number of users with usage rates: $GQ \leq T1 = 14$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.1555
ADPAC-2	CFI	.1428
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.0404
FOCC-2	CFI	.3125
HAM	CFI	.3571
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.2307
NAVC-3	CFI	0
NCDCF	CFI	.9090
NEL	CFI	.0214
NMCSSC	CFI	.2857
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	.1181
RADC-4	CFI	.0787
SAC-1	CFI	.2610
SAC-2	CFI	.5251
SAC-3	CFI	.7140
SAC-4	CFI	1.011

N(CFI) = 21

N(CFID) = 0

N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.10

AFC Reference: A 9.6, A9.7

Feature Name: PROGRAM

Resolution Inequality: N/A

Resolution Equation: N/A

Number of users with usage rates: GQ T1 = N/A

Resolution: Retain as nucleus
Note: Extension proposed in 3.2.2.15.
Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFID	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFID	
NMCSSC	CFID	
RADC-1	CFI	
RADC-2	CFID	
RADC-3	CFID	
RADC-4	CFID	
SAC-1	CFID	
SAC-2	CFID	
SAC-3	CFID	
SAC-4	CFID	

N(CFI) = 12 N(CFID) = 10 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.6

AFC Reference: A 9.6.1

Feature Name: PROCEDURE (Excluding Function)

Resolution Inequality: $U \geq Q T1 = .05$

Resolution Equation: $U = Q251 / (Q251 + Q257 + Q261)$

Number of users with usage rates: $G Q T1 = 17$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	1
ADPAC-2	CFI	.2916
ADPAC-3	CFI	.2857
CSC	CFI	(N/A)
FAA	CFID	.9
FOCC-1	CFI	.8823
FOCC-2	CFI	.8196
HAM	CFI	1
NAVC-1	CFID	(N/C)
NAVC-2	CFID	(N/C)
NAVC-3	CFID	(N/C)
NCDCF	CFI	.7500
NEL	CFID	.8571
NMCSSC	CFI	.6250
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	.1250
RADC-4	CFI	1
SAC-1	CFID	.5714
SAC-2	CFID	.6060
SAC-3	CFID	.8153
SAC-4	CFID	.7128

N(CFI) = 13

N(CFID) = 9

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.6

AFC Reference: A 9.6.1

Feature Name: Close in Parameter List

Resolution Inequality: $U \leq Q \leq T1 = .05$

Resolution Equation: $U = Q252/Q251$

Number of users with usage rates: $Q \leq T1 = 1$

Resolution: Retain as Optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	(N/C)
NCDCF	CFI	0
NEL	CFI	.0833
NMCSSC	CFI	0
RADC-1	CFI	(N/C)
RADC-2	CFI	0
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 17 N(CFID) = 0 N(CFNI) = 5

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.6

AFC Reference: A 9.6.1

Feature Name: Statement label in Parameter List

Resolution Inequality: $U \leq T1 = .05$

Resolution Equation: $U = Q253/Q251$

Number of users with usage rates: $GE T1 = T2 = 5$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.0270
ADPAC-2	CFI	.1428
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	(N/C)
NCDCF	CFI	.0666
NEL	CFID	.1666
NMCSSC	CFI	.4000
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 17

N(CFID) = 0

N(CFNI) = 5

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.8

AFC Reference: A 9.6.1

Feature Name: FUNCTION

Resolution Inequality: $U \leq GQ \leq T1 = .05$

Resolution Equation: $U = Q261 / (Q251 + Q247 + Q261)$

Number of users with usage rates: $GQ \leq T1 = 12$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.0416
ADPAC-3	CFI	.7142
CSC	CFI	(N/A)
FAA	CFID	.2
FOCC-1	CFI	.1176
FOCC-2	CFI.1639	
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.3125
NAVC-3	CFI	0
NCDCF	CFI	.2500
NEL	CFI	0
NMCSSC	CFI	.1875
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	.7500
RADC-4	CFI	0
SAC-1	CFID	.2571
SAC-2	CFID	.2121
SAC-3	CFID	.0119
SAC-4	CFID	.2666

N(CFI) = 17

N(CFID) = 5

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.5

AFC Reference: A 9.6.2

Feature Name: CLOSE

Resolution Inequality: $U \text{ GQ T1} = .05$

Resolution Equation: $U = Q247 / (Q251 + Q247 + Q261)$

Number of users with usage rates: $\text{GQ T1} = 11$

Resolution: Retain as nucleus *

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>	
ADPAC-1	CFI	0	
ADPAC-2	CFI	.6666	
ADPAC-3	CFI	0	
CSC	CFI	(N/A)	
FAA	CFI	.2	
FOCC-1	CFI	0	
FOCC-2	CFI	.1639	
HAM	CFI	0	
NAVC-1	CFI	(N/C)	
NAVC-2	CFI	.0625	
NAVC-3	CFI	0	
NCDCF	CFI	0	
NEL	CFI	.1428	
NMCSSC	CFI	.1875	
RADC-1	CFI	(N/C)	
RADC-2	CFI	.1	
RADC-3	CFI	.1250	
RADC-4	CFI	0	
SAC-1	CFI	.1714	
SAC-2	CFI	.1818	
SAC-3	CFI	.1027	
SAC-4	CFI	.0205	
	N(CFI) = 22	N(CFID) = 0	N(CFNI) = 0

*These data indicate retention as optional, Interview responses to question I-27 require retention as nucleus

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.6

AFC Reference: A 9.6.3

Feature Name: RETURN STATEMENT

Resolution Inequality: N/A

Resolution Equation: Q134

Number of users with usage rates: GQ T1 = N/A

Resolution: Accept as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	48
ADPAC-2	CFI	60
ADPAC-3	CFI	3
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	300
FOCC-2	CFI	25
HAM	CFI	50
NAVC-1	CFI	(N/C)
NAVC-2	CFI	25
NAVC-3	CFI	40
NCDCF	CFI	30
NEL	CFID	100
NMCSSC	CFI	75
RADC-1	CFI	(N/C)
RADC-2	CFI	.9
RADC-3	CFI	50
RADC-4	CFI	20
SAC-1	CFI	700
SAC-2	CFI	1200
SAC-3	CFI	686
SAC-4	CFI	1520

N(CFI) = 21 N(CFID) = 1 N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.11

AFC Reference: A 9.7

Feature Name: PROGRAM DECLARATION

Resolution Inequality: N/A

Resolution Equation: $U = Q275$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFNI	0
ADPAC-2	CFNI	0
ADPAC-3	CFNI	0
CSC	CFNI	0
FAA	CFI	1 or more
FOCC-1	CFI	1
FOCC-2	CFI	0
HAM	CFNI	0
NAVC-1	CFI	0
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFNI	0
NEL	CFID	1
NMCSSC	CFNI	0
RADC-1	CFI	0
RADC-2	CFNI	0
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFNI	1260
SAC-2	CFNI	16
SAC-3	CFNI	110
SAC-4	CFI	50

N(CFI) = 10 N(CFID) = 1 N(CFNI) = 11

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.4

AFC Reference: A 9.8

Feature Name: DIRECT CODE

Resolution Inequality: U GQ T1 = 1

Resolution Equation: U = Q242

Number of users with usage rates: GQ T1 = 16

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	5
ADPAC-2	CFI	130
ADPAC-3	CFI	12
CSC	CFI	(N/A)
FAA	CFI	1 or more
FOCC-1	CFI	60
FOCC-2	CFI	50
HAM	CFI	15
NAVC-1	CFI	(N/C)
NAVC-2	CFI	10
NAVC-3	CFI	5
NCDCE	CFI	15
NEL	CFI	50
NMCSSC	CFI	50
RADC-1	CFI	(N/C)
RADC-2	CFID	1 or more
RADC-3	CFI	150
RADC-4	CFI	0
SAC-1	CFID	880
SAC-2	CFID	2000
SAC-3	CFID	620
SAC-4	CFID	620

N(CFI) = 17

N(CFID) = 5

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.5

AFC Reference: A 10

Feature Name: STRUCTURE FILE

Resolution Inequality: N/A

Resolution Equation: $U = Q107$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	34
ADPAC-2	CFI	0
ADPAC-3	CFI	2
CSC	CFID	(N/A)
FAA	CFNI	0
FOCC-1	CFID	80
FOCC-2	CFID	20
HAM	CFNI	0
NAVC-1	CFID	(N/C)
NAVC-2	CFID	6
NAVC-3	CFID	2
NCDCF	CFID	38
NEL	CFID	300
NMCSSC	CFI	20
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFID	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(CFI) = 4$

$N(CFID) = 9$

$N(CFNI) = 9$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.5

AFC Reference: A 10

Feature Name: STRUCTURE FILE - Hollerith or Binary

Resolution Inequality: N/A

Resolution Equation: $U = Q108/Q107$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	1
ADPAC-2	CFI	0
ADPAC-3	CFI	1
CSC	CFID	(N/A)
FAA	CFNI	0
FOCC-1	CFID	.0500
FOCC-2	CFID	0
HAM	CFNI	0
NAVC-1	CFID	(N/C)
NAVC-2	CFID	0
NAVC-3	CFID	0
NCDCF	CFID	.4736
NEL	CFID	0
NMCSSC	CFI	.5000
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 4 N(CFID) = 9 N(CFNI) = 9

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.5

AFC Reference: A 10

Feature Name: STRUCTURE FILE - Record Length Variability

Resolution Inequality: N/A

Resolution Equation: $U = Q109/Q107$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.3235
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFID	(N/A)
FAA	CFNI	0
FOCC-1	CFID	.0500
FOCC-2	CFID	0
HAM	CFNI	0
NAVC-1	CFID	(N/C)
NAVC-2	CFID	0
NAVC-3	CFID	0
NCDCF	CFID	.5263
NEL	CFID	0
NMCSSC	CFI	.7500
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFID	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(\text{CFI}) = 4$ $N(\text{CFID}) = 9$ $N(\text{CFNI}) = 9$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.2.2.2.5

AFC Reference: A 10

Feature Name: INPUT/OUTPUT -- POS

Resolution Inequality: N/A

Resolution Equation: $U = Q165$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Modify as specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFID	7
ADPAC-2	CFID	0
ADPAC-3	CFID	0
CSC	CFI	(N/A)
FAA	CFNI	1 or more
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	10
NEL	CFNI	0
NMCSSC	CFI	100
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(CFI) = 2$

$N(CFID) = 4$

$N(CFNI) = 16$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.3

AFC Reference: A 10

Feature Name: INPUT/OUTPUT - OPEN Statement with Operand

Resolution Inequality: N/A

Resolution Equation: $U = Q225$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	2
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	0
NEL	CFNI	0
NMCSSC	CFID	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 4 N(CFID) = 2 N(CFNI) = 16

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.3

AFC Reference: A 10

Feature Name: INPUT/OUTPUT - OPEN Statement without Operand

Resolution Inequality: N/A

Resolution Equation: $U = Q226$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	18
ADPAC-2	CFI	0
ADPAC-3	CFI	1
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	0
NEL	CFNI	0
NMCSSC	CFID	20
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(CFI) = 4$ $N(CFID) = 2$ $N(CFNI) = 16$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.3

AFC Reference: A 10

Feature Name: INPUT/OUTPUT - SHUT Statement with Operand

Resolution Inequality: N/A

Resolution Equation: $U = Q227$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	7
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	10
NEL	CFNI	0
NMCSSC	CFID	0
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

N(CFI) = 4 N(CFID) = 2 N(CFNI) = 16

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.3

AFC Reference: A 10

Feature Name: INPUT/OUTPUT - SHUT Statement Without Operand

Resolution Inequality: N/A

Resolution Equation: $U = Q228$

Number of users with usage rates: GQ T1 = N/A

Resolution: Delete in favor of Features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	18
ADPAC-2	CFI	0
ADPAC-3	CFI	2
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	0
NEL	CFNI	0
NMCSSC	CFID	20
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(\text{CFI}) = 4$

$N(\text{CFID}) = 2$

$N(\text{CFNI}) = 16$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.3

AFC Reference: A 10

Feature Name: INPUT/OUTPUT - INPUT Statement

Resolution Inequality: N/A

Resolution Equation: $U = Q229$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	18
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	20
NEL	CFNI	0
NMCSSC	CFID	50
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFI	0

$N(CFI) = 4$

$N(CFID) = 2$

$N(CFNI) = 16$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.3.3 AFC Reference: A 10

Feature Name: INPUT/OUTPUT - OUTPUT Statement

Resolution Inequality: N/A

Resolution Equation: $U = Q230$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Delete in favor of features specified in 3.2.2.11

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	18
ADPAC-2	CFI	0
ADPAC-3	CFI	14
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFNI	0
FOCC-2	CFNI	0
HAM	CFNI	0
NAVC-1	CFNI	(N/C)
NAVC-2	CFNI	0
NAVC-3	CFNI	0
NCDCF	CFID	20
NEL	CFNI	0
NMCSSC	CFID	50
RADC-1	CFNI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	0
RADC-4	CFNI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFNI	0

$N(CFI) = 4$

$N(CFID) = 2$

$N(CFNI) = 16$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.2.2

AFC Reference: A 11.1

Feature Name: ITEM Declaration by Preset Constant

Resolution Inequality: $U GQ T1 = .05$

Resolution Equation: $U = Q8/Q14$

Number of users with usage rates: $GQ T1 = N/A$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	1
ADPAC-2	CFI	0
ADPAC-3	CFI	.0476
CSC	CFI	(N/A)
FAA	CFI	.8
FOCC-1	CFI	.0012
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.1250
NAVC-3	CFI	0
NCDCF	CFI	.0200
NEL	CFI	.2000
NMCSSC	CFI	.1333
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	0
RADC-4	CFNI	.2757
SAC-1	CFID	.0362
SAC-2	CFID	.1789
SAC-3	CFID	.0428
SAC-4	CFI	.0367

N(CFI) = 17

N(CFID) = 3

N(CFNI) = 2

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.4.3

AFC Reference: A 11.2

Feature Name: Multiple Statement Labels

Resolution Inequality: $U \text{ GQ T1} = .05$

Resolution Equation: $U = Q238 / \text{Total Number of Statements}$

Number of users with usage rates: $\text{GQ T1} = \text{N/A}$

Resolution: Retain as nucleus

Data:

Note: These data indicate that the multiple statement labels feature ought to be deleted. Because of the high degree of interplay between the various kinds of Statements in JOVIAL and the ease of syntactic specification obtained by allowing this feature (CED 2448), we are recommending that this feature be retained as nucleus.

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	0
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	.0597
FOCC-2	CFI	0
HAM	CFID	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	0
NCDCF	CFI	.0031
NEL	CFI	0
NMCSSC	CFI	0
RADC-1	CFI	(N/C)
RADC-2	CFID	0
RADC-3	CFID	.0010
RADC-4	CFI	0
SAC-1	CFID	0
SAC-2	CFID	.0001
SAC-3	CFID	0
SAC-4	CFID	0

N(CFI) = 14 N(CFID) = 7 N(CFNI) = 1

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.6.2

AFC Reference: A 12.1

Feature Name: DEFINE Directive

Resolution Inequality: $U G Q T1 = .001$

Resolution Equation: $U = Q236 / \text{Total Number of Statements}$

Number of users with usage rates: $G Q T1 = 8$

Resolution: Retain as optional

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	.0035
ADPAC-2	CFI	.0071
ADPAC-3	CFI	(N/C)
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	.0037
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	.0148
NCDCF	CFI	.0031
NEL	CFI	.0100
NMCSSC	CFI	.0010
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFI	.0010
RADC-4	CFI	0
SAC-1	CFNI	0
SAC-2	CFNI	0
SAC-3	CFNI	0
SAC-4	CFID	0

N(CFI) = 16

N(CFID) = 1

N(CFNI) = 5

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.3

AFC Reference: A 12.2

Feature Name: MODE DIRECTIVE - Automatic and Explicit

Resolution Inequality: $U \leq GQ T1 = .02$

Resolution Equation: $U = Q12/Q14$

Number of users with usage rates: $GQ T1 = 5$

Resolution: Retain as optional (Note extension as specified in 3.2.2.8)

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	.0200
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFNI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	1
NCDCF	CFNI	0
NEL	CFI	.4000
NMCSSC	CFI	.0666
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	.0500
RADC-4	CFNI	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

$N(\text{CFI}) = 12$

$N(\text{CFID}) = 4$

$N(\text{CFNI}) = 6$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.1.1.3 AFC Reference: A 12.2
 Feature Name: MODE DIRECTIVE - Explicit
 Resolution Inequality: U GQ T1 = 1
 Resolution Equation: U = Q13
 Number of users with usage rates: GQ T1 = 4
 Resolution: Retain as optional (Note extension as specified in 3.2.2.8)

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	0
ADPAC-2	CFI	10
ADPAC-3	CFI	0
CSC	CFI	(N/A)
FAA	CFNI	0
FOCC-1	CFI	0
FOCC-2	CFI	0
HAM	CFNI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	0
NAVC-3	CFI	200
NCDCF	CFNI	0
NEL	CFI	40
NMCSSC	CFI	5
RADC-1	CFI	(N/C)
RADC-2	CFNI	0
RADC-3	CFNI	1
RADC-4	CFNI	0
SAC-1	CFID	0
SAC-2	CFID	0
SAC-3	CFID	0
SAC-4	CFID	0

$N(\text{CFI}) = 12$

$N(\text{CFID}) = 4$

$N(\text{CFNI}) = 6$

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.5.5.5

AFC Reference: A 12.5

Feature Name: ALL

Resolution Inequality: $U GQ T = .05$

Resolution Equation: $U = Q306/Q78$

Number of users with usage rates: $GQ T1 = 12$

Resolution: Retain as nucleus

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFID	.3421
ADPAC-2	CFID	.0625
ADPAC-3	CFID	0
CSC	CFI	(N/A)
FAA	CFID	.8
FOCC-1	CFI	.0434
FOCC-2	CFI	0
HAM	CFI	0
NAVC-1	CFI	(N/C)
NAVC-2	CFI	.06666
NAVC-3	CFI	0
NCDCF	CFI	.1000
NEL	CFID	.1000
NMCSSC	CFI	.2000
RADC-1	CFI	(N/C)
RADC-2	CFI	.2
RADC-3	CFI	0
RADC-4	CFI	0
SAC-1	CFID	.1726
SAC-2	CFID	.1230
SAC-3	CFID	.4038
SAC-4	CFID	.4455

N(CFI) = 13

N(CFID) = 9

N(CFNI) = 0

RESOLUTION ANALYSIS

JAQ Reference: J 3.5.6.1
Feature Name: COMPOOL
Resolution Inequality: N/A
Resolution Equation: N/A
Number of users with usage rates: GQ T1 = N/A
Resolution: Retain as nucleus

AFC Reference: A 12.6

Data:

<u>Application</u>	<u>Compliance</u>	<u>Usage</u>
ADPAC-1	CFI	
ADPAC-2	CFI	
ADPAC-3	CFI	
CSC	CFI	
FAA	CFI	
FOCC-1	CFI	
FOCC-2	CFI	
HAM	CFI	
NAVC-1	CFI	
NAVC-2	CFI	
NAVC-3	CFI	
NCDCF	CFI	
NEL	CFI	
NMCSSC	CFID	
RADC-1	CFI	
RADC-2	CFNI	
RADC-3	CFI	
RADC-4	CFNI	
SAC-1	CFI	
SAC-2	CFI	
SAC-3	CFI	
SAC-4	CFI	

N(CFI) = 19

N(CFID) = 1

N(CFNI) = 2

J 3.5.7

OTHER FEATURES

Following is the collection of response to the other feature section of the JAQ that included potential JOVIAL language additions not specifically mentioned in the previous sections of the JAQ:

- | | |
|--------|--|
| CSC | (1) Hexadecimal constant feature provided (as well as octal) |
| | (2) Double precision floating point constants and variables |
| | (3) An ability to declare a PROC as a closed routine using START PROC . . is provided to augment the standards |
| FAA | "EXIT" - Return to monitor instruction |
| FOCC-1 | (1) A formatting and editing capability is included (FORMAT', ENCODE', DECODE') |
| | (2) "LOAD'" is used to bring a program in loader format into core |
| | (3) "MESSAGE'" enables a character string to be output to the system console |
| | (4) "FAULT' and NOFAULT'" are used to enable and disable fault interrupts |

APPENDIX III
INTERVIEW QUESTIONS

This Appendix contains the questions asked of participants at interview. The symbols "Y", "N", "NSF", "CT", denote "Yes", "Na", "No Strang Feeling", and "Can't Tell". "No Strang Feeling" was used by respondents when they felt that the proposed addition, extension, or modification would be only marginally helpful. "Can't Tell" was used when respondents felt they required more specific information regarding implementation of a proposed feature in order to decide whether or not it would be worthwhile. The number in the parenthesis is the related "Approach for Change" reference number.

The interview responses to these questions are included in sequential order in Appendix 4, along with their recommended disposition or resolution.

1) Expanded Character Sets (A1.1.1)

Equipments which permit the use of expanded character sets are currently available. Using these equipments it is possible in writing JOVIAL programs to use, for example, the symbol "<" for the primitive LS, or "{" for BEGIN.

- (1) Do you use expanded character sets now or will you within the next two years?

Y N NSF CT

- (2) Are you in favor of including an expanded set of signs in JOVIAL?

Y N NSF CT

2) User Definable Character Set/Encoding Schemes (A1.1.1)

STC data are composed of characters which are a subset of the Hollerith characters and which admit to a different ("bit pattern") encoding scheme. Are you in favor of a mechanism in the language which would allow the user to define a character set and encoding scheme?

Y N NSF CT

3) Hollerith Comparisons (A6.1)

The Standard (AFM 100-24) does not define a meaning for LS, LQ, GR, and GQ when these relational operators occur between Hollerith operands. Are you in favor of establishing such a meaning?

Y N NSF CT

FIGURE 1

"GROUP" Data Structure

Declaration:

```
GROUP TARGET'COMPLEX $
  BEGIN
    GROUP COMPLEX'ID $
      BEGIN
        ITEM SECTOR'NAME H 24 $
        ITEM THREAT'LEVEL
          I 5 U $
      END "COMPLEX'ID"
    ITEM COMMAND'GROUP H 8 $
    TABLE . . .
    :
  END "TARGET'COMPLEX"
```

FIGURE 2

"SET" Data Structure

```
ITEM AA I 23 S $
ITEM BB I 23 U $
GROUP CC $
  BEGIN
    ITEM CC'1 . . . $
    ITEM CC'2 . . . $
  END
ITEM DD F $ . . .
SET ALPHA $
  BEGIN BB, CC, AA END
```

4) Bit Strings (A 1.1)

In JOVIAL there are four formal item types: Numeric, character (literal), status, and logical (Boolean).

- (1) In your application, do you sometimes deal with data which are none of these formal types, but which are simply strings of bits?

Y N NSF CT

- (2) When you apply functional modifier BIT to a numeric item are you always trying to access a substring of digits, or are there occasions where you are more interested in accessing a string of bits with no particular numeric aspect?

Y N NSF CT

5) GROUP (A1.2)

In JOVIAL, aggregates of data are recurrences of simple items as in arrays or recurrences of sets of simple items as in tables. Sometimes it is convenient to work with sets of items which do not recur (e.g. the example in Figure 1). The figure shows a declaration for data structure called GROUP. A group may contain simple items, arrays, tables or other group. A group might be used as an I/O record. Also, groups may be assigned to groups using only one assignment statement - a group name refers collectively to all the data which comprises it. Note that (serial) tables may be thought of as one dimensional array of groups, where a group is the table entry.

Are you in favor of having such a facility?

Y N NSF CT

6) SET (A 1.2)

A SET (depicted in Figure 2) is similar to a group, but is primarily a naming facility and does not cause allocation to take place. That is, a set declaration provides a collective name for groups, tables, arrays, or other sets which have been declared elsewhere in the program and which do not necessarily occupy contiguous blocks of words. The chief use of set would be to perform assignments in the manner of the group assignment. Are you in favor of having such a facility?

Y N NSF CT

7) Dynamic Allocation (A 1)

Data structures in JOVIAL are allocated prior to execution of the object program. Are you in favor of facilities which allow the programmer to allocate data structures at program execution time.

Y N NSF CT

FIGURE 3

SIGNED MAGNITUDE
SIGNED MAG -3:
1 000 0011
ONES COMPLEMENT -3:
1 111 1100

FIGURE 4

OVERLAY -- USAGES

(1) Sequence:

OVERLAY AA, BB, CC

(2) To reclaim Space

OVERLAY AA = BB \$

(3) To provide names for substructures:

ARRAY MATRIX 2 2 F \$

ARRAY COLUMN'1 2 F \$

ARRAY COLUMN'2 2 F \$

OVERLAY MATRIX = COLUMN'1, COLUMN'2

(4) To assign different attributes to same data

ITEM XX I 48 S \$

ITEM YY F \$

OVERLAY XX = YY \$

8) Storage Allocation Facilities (A 3)

JOVIAL provides much more in the way of explicit storage allocation facilities (e.g., defined packed tables) than most other high level language. In your application, are these facilities regarded as important?

Y N NSF CT

9) Conversion of Programs (A 3)

With respect to your application, have you already, or do you expect in the future to transfer programs written for one computer to another dissimilar computer?

Y N NSF CT

10) Memory Model (A 3)

In the standard specifications a model of the Standard's computer memory is described. This model is made up of words, bits within words, bytes within words, etc. Is this model accurate with respect to the computer (s) used in your application? (This model as opposed to, say, a purely byte oriented memory?)

Y N NSF CT

11) Signed Magnitude (A 3.2)

The Standard requires that JOVIAL compilers produce code which effects a simulation of a signed magnitude numeric representation when the machine's numeric representation is other than signed magnitude. (Shown in Figure 3). This requirement enables compatibility to be maintained in the use of BIT and OVERLAY, but may cause great expense in terms of the compiled code generated. Are you in favor of dropping absolute adherence to signed magnitude?

Y N NSF CT

12) Hexadecimal Constants (A 3.1)

In your application, would hexadecimal constants, as opposed to octal constants, be useful?

Y N

13) Packing (A 3)

Packing in JOVIAL can be specified only with respect to tables. Packing, however, is only peripherally associated with tables. Are you in favor of allowing packing specifications for other data aggregates as well? (e.g., Arrays, groups)

Y N NSF CT

14) OVERLAY (A 3.6)

There are two distinct usages for OVERLAY (e.g., see Figure 4). Examples showing sequencing and space reclaiming usages are shown above the dashed line in the Figure; examples showing naming and multiple interpretation of the same bit string are shown below the dashed line. In your application, is OVERLAY used more in the area characterized by examples (1) and (2) or more in the area characterized by (3) and (4)?

(1)/(2) More (3)/(4) More Both the Same CT

15) Absolute Addresses and 'LOC (A 3.7, A 3.8)

(1) Are you in favor of retaining absolute address specifications, as in OVERLAY and 'PROGRAM declarations?

Y N NSF CT

(2) Are you in favor of retaining the 'LOC facility for manipulating absolute addresses at execution time?

Y N NSF CT

16) Double Precision (A 4.1)

(1) Is your hardware capable of double precision arithmetic?

Y N

(2) Are you in favor of incorporating a double precision capability into the language?

Y N NSF CT

17) Prefix Minus (A 4.5)

The precedence of prefix minus in the JOVIAL Standard is different from the convention taken in algebra. (See Figure 5). Are you in favor of altering the Standard Specification so that it will comply with the algebraic convention?

Y N NSF CT

18) Functional Modifier ODD (A 7)

When applied to a numeric item, ODD produces the value "true" if the least significant bit of the item is 1; otherwise, ODD produces "false".

(1) Do you have functional modifier ODD in your version of JOVIAL?

Y N

(2) Are you in favor of retaining ODD?

Y N NSF CT

19) Structure Operators - ARRAYS, TABLES (A 5)

It is possible to define generalized arithmetic operators which act on array or table elements when their operands are non-subscripted array or table names (e.g. Example in Figure 6). Are you in favor of generalizing arithmetic operators in this way?

Y N NSF CT

20) Structure Operator - MATRIX (A 5)

If a new data structure called MATRIX is introduced, along with extended arithmetic operators it is possible to specify matrix arithmetic computations without the use of FOR-loops (See Figure 7). Note that a MATRIX is a 2 dimensional ARRAY and that "*" in the figure implies matrix multiplication. Would you be in favor of incorporating such features?

Y N NSF CT

21) Structure Operator - COMPLEX (A 8.2)

A data structure of type COMPLEX would be composed of two numeric items -- a "real" item and an "imaginary" item. It is possible to generalize the arithmetic operators in such a way that whenever they appear with operands declared as type COMPLEX, they cause complex arithmetic operations to take place. Are you in favor of such features as these?

Y N NSF CT

22) Rounding (A 8.2)

In the Standard, numeric items may be declared with a rounding specifier, implying that any value being assigned to such items must be rounded prior to assignment. Sometimes it is desirable to round in some assignments and not in others. In such cases, a rounding specification is more properly an attribute of assignment statements. Are you in favor of switching to the rounding specification mechanism as shown in Figure 8?

Y N NSF CT

23) Parallel "Monitoring" (A 9)

Sometimes it is desirable to monitor the status of the hardware in parallel with other processing and to take action immediately upon a signal that the hardware status has been changed (see Figure 9). "ON" causes a monitoring process which tests for the condition "UNIT READY" to be activated during the execution of the code between BEGIN and END. Monitoring is suspended when control passes "through" the END. If the condition UNIT READY occurs while the monitoring process is active, the statement immediately following the ON

statement is executed. In the figure, this statement is a CLOSE invocation; control will be returned to the program step which was being processed when the condition occurred, once the CLOSE terminate execution. Are you in favor of such a feature as the ON statement?

Y N NSF CT

24) STOP (A 9.3)

STOP in a closed subprogram causes control to pass back to the calling program; when executed in the main program, it causes control to pass to the monitor (if there is one). Are you in favor of always having STOP return control to the monitor as opposed to retaining the two different actions caused by STOP, as described above?

Y N NSF CT

25) Pausing (A 9.3)

Are you in favor of having a pausing facility?

Y N NSF CT

26) Label Items (A 9)

It is possible to introduce a data type, called a label item, which takes statement labels as values. (See Figure 10). Label items permit a greater flexibility (and are cleaner!) in switch manipulation. Are you in favor of adopting label items?

Y N NSF CT

27) Closed Subprograms (A 9.6)

In the Standard JOVIAL there are two kinds of closed subroutines -- the Close and the Procedure. Closes may be defined within FOR loops and may reference loop variables; closes may be defined within closes. With the exception of the above two features, procedures do everything a close does and also allow the user to have input and output parameters. Are you in favor of relying solely on procedures and deleting closes?

Y N NSF CT

28) Alternate Procedure Entrances (A 9.6.1)

It is sometimes convenient to have more than one point of entry into a procedure (See Figure 11). The figure shows alternate entrance declarations. Note that each alternate entrance may have a different calling sequence if desired. An alternate entrance is invoked exactly as if it were a procedure but entrance is made at the associated statement label. Are you in favor of having alternate entrances?

Y N NSF CT

FIGURE 5

PREFIX MINUS

In JOVIAL:

$$- XX ** 2 = 9 \text{ if } XX = 3$$

In ALGEBRA:

$$- X^2 = -9 \text{ if } X = 3$$

FIGURE 6

"STRUCTURE" OPERATOR - ARRAY

ARRAY AA 10 F \$

ARRAY BB 10 F \$

ARRAY CC 10 F \$

AA = BB * CC \$ Means

"FOR I = 0, 1, 9, \$

AA (\$I\$) = BB (\$I\$) * CC (\$I\$) \$ "

FIGURE 7

"Structure" Operator -- MATRIX

New data type MATRIX

MATRIX AA 2 2 F \$

MATRIX BB 2 2 F \$

MATRIX CC 2 2 F \$

Then

AA = BB * CC \$ means

matrix multiplication.

FIGURE 8

Rounded Assignment

AA = BB + CC * (DD - EE) ROUNDED \$

FIGURE 9

Parallel Monitoring

ON UNIT READY \$

GOTO END'INPUT'RTN \$

BEGIN

- { SCOPE OVER WHICH UNIT
- { STATUS BEING MONITORED

END MONITORING DEACTIVATED HERE

FIGURE 10

Label Items

Declaration:

```
ITEM AA LABEL $
```

USE:

```
AA = ADD'EM'UP $  
    "ADD'EM'UP is a statement  
    label; it is now the value of AA'  
GOTO AA $ "NON INDEXED SWITCH"
```

In an array:

```
ARRAY AA 3 LABEL $  
GOTO AA ($ 2 $) "INDEXED SWITCH"
```

FIGURE 11

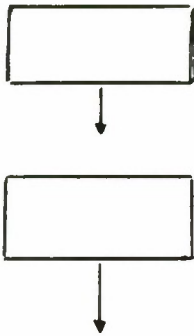
Alternate Entrances

```
PROC AA (-----)  
ENTRANCE BB ( o o o )  
ENTRANCE CC ( o o o )  
BEGIN  
BB. { PROC BODY  
CC. {  
END
```

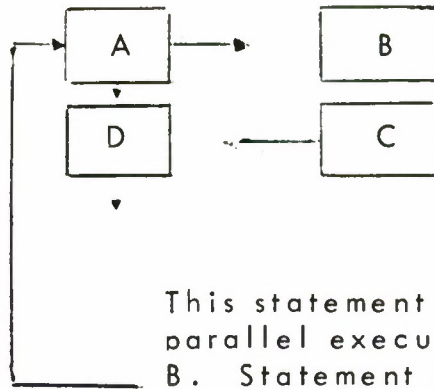

FIGURE 12

SERIAL/PARALLEL EXECUTION:

SERIAL



PARALLEL



This statement initiates a parallel execution of statement B. Statement C terminates the processing stream begun at B and signals "completion" to D.

FIGURE 13

"FORTRAN"

```
100      FORMAT      (F 12.3, 14, A6)
        READ (100) AA, BB, CC
```

variables to be
evaluated by reading.

FIGURE 14

"COBOL"

ITEM AA I 48 S \$

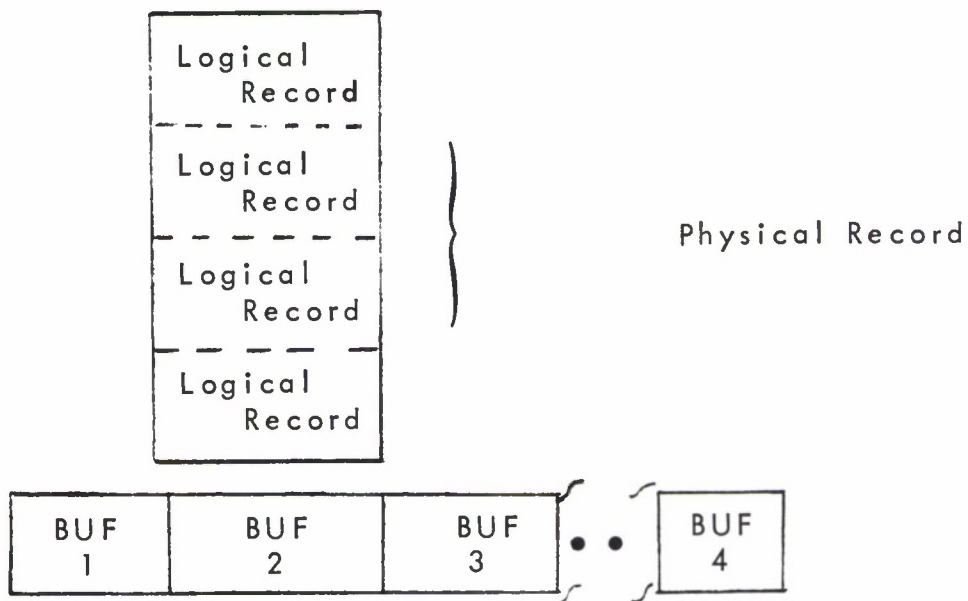
ITEM BB PICTURE (9 (7).9 (3)) \$

AA = BB \$ causes conversion from character string
to binary

BB = AA \$ causes conversion from binary to character
string

FIGURE 15

Automatic Blocking/Deblocking/Buffering



User allocates as many
buffers as he wants - I/O
system automatically fills
(input) or empties (output)
buffers.

FIGURE 16

NAME SCOPES

LOCAL \$

 BEGIN

-
-
-

} Names declared in this block take precedence over names declared externally, when they are referenced in the block

 END

FIGURE 17

EXTENDED DEFINE

 DEFINE SUM (XX, YY, ZZ) "XX = YY + ZZ" \$

 SUM (AA, BB, CC) \$

 would produce

 AA = BB + CC \$

FIGURE 18

Extended MODE

```
MODE I I 48 S $
MODE Q F $
```

The non-declared identifier INDEX appearing after these extended MODE directives would be automatically declared as I 48 S. The non-declared identifier QUOVADIS would be automatically declared as F.

FIGURE 19

"MACROS"

```
% ITEM NN I 5 U $
% NN = 9 $
$ FOR I = 0, 1, NN
AA ($ I $) = BB ($ I $) * CC ($ I $)
```

COMPILE
TIME
STATEMENTS

When compile time program is executed, the program segment below is produced (AND COMPILED)

```
AA ($ 0 $) = BB ($ 0 $) * CC ($ 0 $) $
AA ($ 1 $) = BB ($ 1 $) * CC ($ 1 $) $ •••
AA ($ 9 $) = BB ($ 9 $) * CC ($ 9 $) $
```

29) Recursive Subprograms (A 9.6)

Are you in favor of providing recursive routines, facilities, i.e., routines which may call themselves?

Y N NSF CT

30) Parallel Processing (A 9)

JOVIAL statements are executed in a basically serial mode -- one statement is executed by the processor, then the next, etc. When computer systems with more than one processor are used, it is possible to execute several different program segments in parallel by assigning different program segments to different processors (See Figure 12). Are you in favor of incorporating formal language elements into JOVIAL which would allow the programmer to assign program statements to different processors?

Y N NSF CT

31) Stream Oriented I/O (A 10)

- (1) An important part of data processing involves converting data in character string representations to and from data in internal (binary) representations. Do you currently employ formal mechanisms in the language to perform this task?

Y N

- (2) Stream Oriented I/O Mechanisms

Two approaches are widely used to perform conversions. The "FORTRAN method" causes conversions to take place under the control of a FORMAT (see Figure 13). The "COBOL approach" allows, in essence, for item declarations to be made with character string attributes; conversion is then performed by assignment of an item declared with an internal representation to an item declared with a character string representation, and vice-versa (see Figure 14). Are you in favor of incorporating FORTRAN type mechanisms, COBOL type mechanisms, or neither?

FORTRAN COBOL NEITHER

32) Functional Files (A 10)

Functional file systems allow the user to describe functional records and record manipulations independently of any actual storage medium or transmission devices. Typically these systems provide for the automatic blocking and deblocking of "logical" records and for automatically controlled multiple buffer (see Figure 15). With respect to such "classical" functional file systems, programmers simply specify functional manipulations of logical records; all details are taken care of automatically. Are you in favor of adopting a functional file system for JOVIAL?

Y N NSF CT

33) Device Oriented I/O (A10)

Device oriented I/O refers to the processes by which transmission devices are commanded to perform certain tasks, e.g. REWIND, INITIATE, READ. The I/O devices will operate in parallel with the central processor, and the Central Processor will be interrupted by the devices. Are you in favor of incorporating mechanisms in JOVIAL which allow the programmer to deal with these processes?

Y N NSF CT

34) Proc-like Name Scopes (A 11)

(See Figure 16). This figure illustrates a possible name scope facility which is similar to that provided by PROCS. The primitive LOCAL declares that all declarations made in the next statement are local to it. Are you in favor of such a facility?

Y N NSF CT

35) Extended DEFINE (A 12.1)

(See Figure 17). It is possible to extend the capabilities of the DEFINE directive to allow for the automatic insertion of actual calling sequence parameters in the DEFINE string, before the string replaces the defined identifier itself. Are you in favor of such a facility?

Y N NSF CT

36) Extended MODE (A 12.2)

(See Figure 18). The extended MODE facility would allow the programmer to mode declare items on the basis of the first character of their names. Are you in favor of the extended MODE facility?

Y N NSF CT

37) Macro Facilities (A12)

Macro facilities allow the programmer to make definitions for source text strings and have them automatically inserted into the source stream when they are referenced by their definition name; they also enable the programmer to direct different source text strings to be desired on the basis of certain conditions which may hold true at compile time. (See Figure 19). In the system shown, there are two kinds of program statements, both of which have the same form except for a beginning sign, %. Statements marked with % are compile time statements. The compiler translates compile time statements and then executes them -- at compile time. The example shown causes the non-compile time statement to be replicated 10 times as indicated in the figure. It is these replicated statements which will be operated at execution time. Are you in favor of such Macro facilities?

Y N NSF CT

APPENDIX IV

INTERVIEW RESPONSE ANALYSIS

This appendix contains interview question response summaries and analyses. The symbols "Y", "N", "NSF", and "CT" are described in Appendix 3. Resolutions are based upon criteria fully described in Sections I and II of this document. The meaning of resolutions of the form "No recommendation at this time" is given in Section II.

INTERVIEW ANALYSIS

Interview Question: I-1

AFC Reference: A 1.1.1

Question Content: (1) Use Expanded Character Sets Now or in Future?
 (2) In Favor of Adopting Expanded Set in Standard?

Number of Users with Response Y:	(1) 2	(2) 2
Number of Users with Response N:	(1) 10	(2) 10
Number of Users with Response NSF:	(1) 0	(2) 0
Number of Users with Response CT:	(1) 0	(2) 0

Data:

Application	Response	
ADPAC	(1) N	(2) N
FAA	N	Y
FOCC	N	N
HAM	N	N
NAVC	N	N
NCDCF	N	N
NEL	Y	Y
NHCP	N	N
NMCSSC	Y	N
RADC	N	N
SAC	N	N
TWA	N	N

Resolution: Do not adopt a standard expanded character set.

INTERVIEW ANALYSIS

Interview Question: 1-2

AFC Reference: A1.1.1

Question Content: In Favor of User Definable Character Set/Encoding Scheme Facility?

Number of Users with Response Y: 6

Number of Users with Response N: 6

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	N
NAVC	N
NCDCF	Y
NEL	Y
NHCP	N
NMCSSC	Y
RADC	N
SAC	N
TWA	N

Resolution: Adopt 'CHARCODE directive (3.2.2.3) (optional)

INTERVIEW ANALYSIS

Interview Question: 1-3

AFC Reference: A 6.1

Question Content: In Favor of Hollerith ">", "<" Comparisons?

Number of Users with Response Y: 10

Number of Users with Response N: 1

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	NSF
FAA	Y
FOCC	Y
HAM	N
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	Y
TWA	Y

Resolution: Adopt literal comparison feature (3.2.2.4.2) (nucleus)

INTERVIEW ANALYSIS

Interview Question: 1-4

AFC Reference: A 1.1

Question Content: (1) Deal With Data Which are Simply BIT Strings?
 (2) When BIT is Used, Is Referenced Data Just BIT String?

Number of Users with Response Y: (1) 8 (2) 9

Number of Users with Response N: (1) 4 (2) 3

Number of Users with Response NSF: (1) 0 (2) 0

Number of Users with Response CT: (1) 0 (2) 0

Data:

<u>Application</u>	<u>Response</u>	
	(1)	(2)
ADPAC	Y	Y
FAA	Y	Y
FOCC	N	N
HAM	N	Y
NAVC	Y	Y
NCDCF	N	N
NEL	Y	Y
NHCP	Y	Y
NMCSSC	Y	Y
RADC	N	N
SAC	Y	Y
TWA	Y	Y

Note: The purpose of Questions 1-4 (1), (2) was to ascertain whether or not a formal item type "bit string" ought to be incorporated into JOVIAL. Operations on bit strings might include assignment, concatenation, "and," "or," "not," etc.

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: 1-5

AFC Reference: A 1.2

Question Content: In Favor of GROUP Data Structure ?

Number of Users with Response Y: 7

Number of Users with Response N: 4

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	NSF
FAA	Y
FOCC	Y
HAM	N
NAVC	N
NCDCF	N
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	N
TWA	Y

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: I-6

AFC Reference: A 1.2

Question Content: In Favor of SET Data Structure?

Number of Users with Response Y: 8

Number of Users with Response N: 2

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	NSF
FAA	Y
FOCC	NSF
HAM	N
NAVC	N
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	Y
TWA	Y

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: 1-7

AFC Reference: A 1

Question Content: In Favor of Dynamic Allocation Facilities?

Number of Users with Response Y: 4

Number of Users with Response N: 6

Number of Users with Response NSF: 0

Number of Users with Response CT: 2

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	CT
FOCC	N
HAM	N
NAVC	N
NCDCF	Y
NEL	CT
NHCP	N
NMCSSC	Y
RADC	N
SAC	Y
TWA	Y

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: I-8

AFC Reference: A 3

Question Content: Explicit Storage Allocation Facilities Important?

Number of Users with Response Y: 12

Number of Users with Response N: 0

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	Y
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	Y
TWA	Y

Note: This question was not intended to determine the need for any specific feature or features, but rather to measure, in general terms, command and control programming requirements. Users felt strongly that these explicit storage allocation facilities are among the most powerful of the JOVIAL capabilities.

Resolution: N/A

INTERVIEW ANALYSIS

Interview Question: I-9

AFC Reference: A 3

Question Content: Transfer Already, Or In Future, Programs Between Different Computer Systems?

Number of Users with Response to Y: 10

Number of Users with Response to N: 2

Number of Users with Response to NSF: 0

Number of Users with Response to CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	N
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	N
NMCSSC	Y
RADC	Y
SAC	Y
TWA	Y

Note: This question, like I-9, was not intended to determine the need for any specific feature or features, but rather to ascertain user requirements. The fact that the great majority of users responded with "yes" implies that the command and control language should be very machine independent. This conflicts with the highly machine dependent explicit storage allocation facilities deemed so important by users. It is important to note that most users commented that they were willing to sacrifice machine independence for allocation facilities.

Resolution: N/A

INTERVIEW ANALYSIS

Interview Question: 1-10

AFC Reference: A 3

Question Content: Is "Memory Model" Sufficient?

Number of Users with Response Y: 12

Number of Users with Response N: 0

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	Y
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	N
SAC	Y
TWA	Y

Resolution: Retain the memory model -- words, bits, bytes -- as is.

INTERVIEW ANALYSIS

Interview Question: I-11

AFC Reference: A 3.2

Question Content: In Favor of Dropping Absolute Adherence to Signed Magnitude Representation?

Number of Users with Response Y: 9

Number of Users with Response N: 1

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	N
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	NSF
TWA	NSF

Resolution: Adopt "relaxed" signed magnitude interpretation (3.2.2.5).

INTERVIEW ANALYSIS

Interview Question: I-12

AFC Reference: A 3.1

Question Content: In Favor of Hexadecimal Constants?

Number of Users with Response Y: 5

Number of Users with Response N: 7

Number of Users with Response NSF: N/A

Number of Users with Response CT: N/A

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	Y
FOCC	Y
HAM	N
NAVC	N
NCDCF	N
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	N
SAC	Y
TWA	N

Note: It is recommended that compilers must implement either the octal constant or the hexadecimal constant, but that they may implement both.

Resolution: Adopt a hexadecimal constant capability (3.2.2.1) (optional)

INTERVIEW ANALYSIS

Interview Question: I-13

AFC Reference: A 3

Question Content: In Favor of Generalized Packing?

Number of Users with Response Y: 9

Number of Users with Response N: 3

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	N
NAVC	N
NCDCF	N
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	Y
TWA	Y

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: 1-14

AFC Reference: A 3.6

Question Content: Use of OVERLAY

Number of Users with Response Y: "(1)/(2) more"

Number of Users with Response N: "(3)/(4) more"

Number of Users with Response NSF: "Both the Same"

Number of Users with Response CT: 1

Data:

<u>Application</u>	<u>Response</u>
ADPAC	(3)/(4) more
FAA	Both the same
FOCC	Both the same
HAM	CT
NAVC	(1)/(2) more
NCDCF	(3)/(4) more
NEL	(3)/(4) more
NHCP	Both the same
NMCSSC	Both the same
RADC	(3)/(4) more
SAC	(1)/(2) more
TWA	Both the same

Note: (See next page)

Resolution: No recommendation at this time.

Note:

The purpose of this question was to determine whether OVERLAY is used more for directing allocation -- its immediately apparent function -- or for applying multiple interpretations to data and data structures. DDI contended in the AFC that it would be preferable to provide distinct features for the two fundamental functions performed by OVERLAY because this would provide greater ease in training novice programmers and because the ambiguity of the intent of an OVERLAY (is the programmer directing allocation or is he assigning more than one set of attributes to a given datum or data structure?) would be avoided. The responses indicate that nine of the 12 applications use OVERLAY at least as much for performing the "multiple interpretations" function as for the allocation direction function. Based upon these responses, we should recommend distinct features to replace OVERLAY; however, owing to the fact that we are proposing no action be taken as yet in implementing new data structures, we feel that it would be counter-productive to propose features to replace OVERLAY at this time.

INTERVIEW ANALYSIS

Interview Question: I-15

AFC Reference: A 3.7/A 3.8

Question Content: (1) In Favor of Obtaining Absolute Address Specifications?
 (2) In Favor of Retaining 'LOC?

Number of Users with Response Y: (1) 8 (2) 9

Number of Users with Response N: (1) 4 (2) 3

Number of Users with Response NSF: (1) 0 (2) 0

Number of Users with Response CT: (1) 0 (2) 0

Data:

<u>Application</u>	<u>Response</u>	
	(1)	(2)
ADPAC	Y	Y
FAA	Y	Y
FOCC	Y	Y
HAM	N	N
NAVC	Y	N
NCDCF	N	N
NEL	N	Y
NHCP	Y	Y
NMCSSC	N	Y
RADC	Y	Y
SAC	Y	Y
TWA	Y	Y

Resolution: (1) Retain absolute address specification facility as nucleus.

(2) Retain 'LOC feature as nucleus.

INTERVIEW ANALYSIS

Interview Question: I-16

AFC Reference: A 4.1

Question Content: (1) Does Hardware Have Double Precision?
 (2) In Favor of Double Precision?

Number of Users with Response Y: (1) 8 (2) 8

Number of Users with Response N: (1) 4 (2) 4

Number of Users with Response NSF: (1) N/A (2) 0

Number of Users with Response CT: (1) N/A (2) 0

Data:

<u>Application</u>	<u>Response</u>	
	(1)	(2)
ADPAC	Y	Y
FAA	Y	Y
FOCC	N	N
HAM	N	N
NAVC	Y	N
NCDCF	Y	Y
NEL	Y	Y
NHCP	Y	Y
NMCSSC	Y	Y
RADC	Y	Y
SAC	N	Y
TWA	N	N

Resolution: Adopt extended precision feature (3.2.2.10) (optional)

Note: The extended precision facility is made optional since it is a function of hardware.

INTERVIEW ANALYSIS

Interview Question: 1-17

AFC Reference: A 4.5

Question Content: In Favor of Algebra-Like Unary Operator Precedence?

Number of Users with Response Y: 6

Number of Users with Response N: 4

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	Y
FOCC	N
HAM	N
NAVC	Y
NCDCF	Y
NEL	NSF
NHCP	Y
NMCSSC	NSF
RADC	N
SAC	Y
TWA	Y

Resolution: Adopt Modification of Unary Operator Precedence (3.2.2.6) (nucleus)

INTERVIEW ANALYSIS

Interview Question: I-18

AFC Reference: A 7

Question Content: (1) Is ODD implemented?
 (2) Should ODD be retained?

Number of Users with Response Y: (1) 7 (2) 5

Number of Users with Response N: (1) 5 (2) 6

Number of Users with Response NSF: (1) N/A (2) 1

Number of Users with Response CT: (1) N/A (2) 0

Data:

<u>Application</u>	<u>Response</u>	
	(1)	(2)
ADPAC	Y	Y
FAA	N	Y
FOCC	Y	Y
HAM	N	N
NAVC	Y	N
NCDCF	N	N
NEL	Y	NSF
NHCP	Y	Y
NMCSSC	N	N
RADC	Y	Y
SAC	N	N
TWA	Y	N

Resolution: Retain ODD as optional.

INTERVIEW ANALYSIS

Interview Question: I-19

AFC Reference: A 5

Question Content: In Favor of Structure Operators for Arrays and Tables?

Number of Users with Response Y: 2

Number of Users with Response N: 8

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	N
HAM	N
NAVC	N
NCDCF	NSF
NEL	NSF
NHCP	N
NMCSSC	N
RADC	N
SAC	N
TWA	N

Resolution: Do not adopt structure operators.

INTERVIEW ANALYSIS

Interview Question: 1-20

AFC Reference: A 5

Question Content: In favor of MATRIX and MATRIX operators?

Number of Users with Response Y: 3

Number of Users with Response N: 8

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	Y
FOCC	N
HAM	N
NAVC	N
NCDCF	NSF
NEL	N
NHCP	Y
NMCSSC	Y
RADC	N
SAC	N
TWA	N

Resolution: Do not adopt matrix and matrix operators.

INTERVIEW ANALYSIS

Interview Question: 1-21

AFC Reference: A 5

Question Content: In favor of COMPLEX and operators?

Number of Users with Response Y: 3

Number of Users with Response N: 9

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	Y
FOCC	N
HAM	N
NAVC	N
NCDCF	Y
NEL	Y
NHCP	N
NMCSSC	N
RADC	N
SAC	N
TWA	N

Resolution: Do not adopt a data structure type COMPLEX and complex arithmetic.

INTERVIEW ANALYSIS

Interview Question: I-22

AFC Reference: A 8.2

Question Content: In Favor of ROUND in Assignment Statement?

Number of Users with Response Y: 5

Number of Users with Response N: 6

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	Y
FOCC	N
HAM	N
NAVC	N
NCDCF	Y
NEL	NSF
NHCP	Y
NMCSSC	Y
RADC	N
SAC	Y
TWA	N

Resolution: Adopt Round Assignment Statement (3.2.2.14) (optional)

INTERVIEW ANALYSIS

Interview Question: 1-23

AFC Reference: A 9

Question Content: In Favor of Parallel Monitoring?

Number of Users with Response Y: 6

Number of Users with Response N: 5

Number of Users with Response NSF: 0

Number of Users with Response CT: 1

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	N
HAM	N
NAVC	Y
NCDCF	N
NEL	Y
NHCP	CT
NMCSSC	Y
RADC	N
SAC	Y
TWA	N

Resolution: Implementors may effect such a feature as this by use of the proposed EXECUTE Statement (3.2.2.11.1). Therefore no explicit parallel monitoring feature is proposed.

INTERVIEW ANALYSIS

Interview Question: I-23

AFC Reference: A 9

Question Content: In Favor of Parallel Monitoring?

Number of Users with Response Y: 6

Number of Users with Response N: 5

Number of Users with Response NSF: 0

Number of Users with Response CT: 1

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	N
HAM	N
NAVC	Y
NCDCF	N
NEL	Y
NHCP	CT
NMCSSC	Y
RADC	N
SAC	Y
TWA	N

Resolution: Implementors may effect such a feature as this by use of the proposed EXECUTE Statement (3.2.2.11.1). Therefore no explicit parallel monitoring feature is proposed.

INTERVIEW ANALYSIS

Interview Question: I-24

AFC Reference: A 9.3

Question Content: In Favor of Altering STOP?

Number of Users with Response Y: 9

Number of Users with Response N: 3

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	N
FOCC	N
HAM	Y
NAVC	Y
NCDCF	Y
NEL	Y
NHCPQ	Y
NMCSSC	Y
RADC	N
SAC	Y
TWA	Y

Note: The STOP statement, as modified, causes termination of the operating program no matter where it occurs. Thus, a STOP never returns control to the calling program when executed in a closed subprogram.

Resolution: Adopt the modification to STOP (3.2.2.12) (nucleus)

INTERVIEW ANALYSIS

Interview Question: I-25

AFC Reference: A 9.3

Question Content: In Favor of a Pausing Feature?

Number of Users with Response Y: 5

Number of Users with Response N: 6

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	N
FOCC	N
HAM	N
NAVC	N
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	N
RADC	NSF
SAC	Y
TWA	Y

Resolution: Adopt the Pause Statement (3.2.2.13) (optional)

INTERVIEW ANALYSIS

Interview Question: I-26

AFC Reference: A 9

Question Content: In Favor of Label Item?

Number of Users with Response Y: 5

Number of Users with Response N: 5

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	N
FOCC	NSF
HAM	N
NAVC	N
NCDCF	N
NEL	N
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	NSF
TWA	Y

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: I-27

AFC Reference: A 9.6

Question Content: In Favor of Deleting Closes?

Number of Users with Response Y: 2

Number of Users with Response N: 9

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	N
FOCC	N
HAM	Y
NAVC	N
NCDCF	N
NEL	NSF
NHCP	N
NMCSSC	Y
RADC	N
SAC	N
TWA	N

Note: Users feel that the ability to define Closes within FOR loops and the ability to define Closes within Closes are important.

Resolution: Retain both Procedures and Closes.

INTERVIEW ANALYSIS

Interview Question: I-28

AFC Reference: A 9.6.1

Question Content: In Favor of Alternate Procedures Entrances?

Number of Users with Response Y: 8

Number of Users with Response N: 3

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	N
HAM	N
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	N
SAC	NSF
TWA	Y

Resolution: Adopt Alternate Procedure Entrances (3.2.2.9) (nucleus)

INTERVIEW ANALYSIS

Interview Question: 1-29

AFC Reference: A 9.6

Question Content: In Favor of Recursive Subprograms?

Number of Users with Response Y: 5

Number of Users with Response N: 7

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	N
FOCC	N
HAM	Y
NAVC	N
NCDCF	N
NEL	Y
NHCP	N
NMCSSC	Y
RADC	Y
SAC	N
TWA	Y

Resolution: No recommendation at this time.

INTERVIEW ANALYSIS

Interview Question: I-30

AFC Reference: A 9

Question Content: In Favor of Parallel Processing Features?

Number of Users with Response Y: 3

Number of Users with Response N: 7

Number of Users with Response NSF: 1

Number of Users with Response CT: 1

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	NSF
FOCC	Y
HAM	Y
NAVC	N
NCDCF	N
NEL	N
NHCP	CT
NMCSSC	N
RADC	N
SAC	N
TWA	Y

Resolution: Do not adopt parallel processing features.

INTERVIEW ANALYSIS

Interview Question: 1-31

AFC Reference: A 10

Question Content: (1) Data editing and conversion features now?
 (2) Prefer FORTRAN, COBOL, or NEITHER?

Number of Users with Response Y: (1) 2 (2) FORTRAN - 9

Number of Users with Response N: (1) 10 (2) COBOL - 2

Number of Users with Response NSF: (1) N/A (2) NEITHER - 1

Number of Users with Response CT: (1) N/A

Data:

<u>Application</u>	<u>Response</u>	
	(1)	(2)
ADPAC	N	FTN
FAA	N	FTN
FOCC	Y	FTN
HAM	N	NEITHER
NAVC	N	FTN
NCDCF	N	FTN
NEL	N	CBL
NHCP	N	FTN
NMCSSC	N	FTN
RADC	N	FTN
SAC	Y	FTN
TWA	N	CBL

Resolution: Adopt Editing and Conversion Module (3.2.2.11.3) (nucleus)

INTERVIEW ANALYSIS

Interview Question: 1-32

AFC Reference: A 10

Question Content: In Favor of Functional Files?

Number of Users with Response Y: 7

Number of Users with Response N: 5

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	N
NAVC	Y
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	N
RADC	N
SAC	N
TWA	N

Resolution: Adopt Functional File Module (3.2.2.11.2) (optional)

INTERVIEW ANALYSIS

Interview Question: I-33

AFC Reference: A 10

Question Content: In Favor of Device Oriented I/O?

Number of Users with Response Y: 8

Number of Users with Response N: 4

Number of Users with Response NSF: 0

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	N
FOCC	Y
HAM	N
NAVC	N
NCDCF	Y
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	Y
SAC	Y
TWA	N

Resolution: Adopt Device Module (3.2.2.11.1) (optional)

INTERVIEW ANALYSIS

Interview Question: I-34

AFC Reference: A 11

Question Content: In Favor of Proc-Like Name Scopes

Number of Users with Response Y: 1

Number of Users with Response N: 9

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	Y
FOCC	N
HAM	N
NAVC	N
NCDCF	N
NEL	NSF
NHCP	N
NMCSSC	N
RADC	NSF
SAC	N
TWA	N

Resolution: Do not adopt Proc-Like Name Scopes.

INTERVIEW ANALYSIS

Interview Question: I-35

AFC Reference: A 12.1

Question Content: In Favor of Extended Define?

Number of Users with Response Y: 7

Number of Users with Response N: 3

Number of Users with Response NSF: 2

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	NSF
HAM	N
NAVC	Y
NCDCF	N
NEL	Y
NHCP	Y
NMCSSC	Y
RADC	NSF
SAC	N
TWA	Y

Resolution: Adopt Extended Define Facility (3.2.2.7) (optional)

INTERVIEW ANALYSIS

Interview Question: I-36

AFC Reference: A 12.2

Question Content: In Favor of Extended Mode?

Number of Users with Response Y: 7

Number of Users with Response N: 4

Number of Users with Response NSF: 1

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	Y
FAA	Y
FOCC	Y
HAM	N
NAVC	Y
NCDCF	NSF
NEL	Y
NHCP	N
NMCSSC	Y
RADC	N
SAC	Y
TWA	N

Resolution: Adopt Extended Mode Facility (3.2.2.8) (optional)

INTERVIEW ANALYSIS

Interview Question: I-37

AFC Reference: A 12

Question Content: In Favor of Macro Facilities?

Number of Users with Response Y: 2

Number of Users with Response N: 6

Number of Users with Response NSF: 4

Number of Users with Response CT: 0

Data:

<u>Application</u>	<u>Response</u>
ADPAC	N
FAA	NSF
FOCC	Y
HAM	N
NAVC	N
NCDCF	Y
NEL	NSF
NHCP	NSF
NMCSSC	N
RADC	NSF
SAC	N
TWA	N

Resolution: Do not adopt Macro Facilities.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Data Dynamics, Inc. 9800 S. Sepulveda Boulevard Los Angeles, California 90045		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP N/A	
3. REPORT TITLE JOVIAL EVALUATION PROJECT			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Final Report			
5. AUTHOR(S) (First name, middle initial, last name) William M. O'Brien			
6. REPORT DATE 15 October 1968	7a. TOTAL NO. OF PAGES 269	7b. NO. OF REFS 3	
8a. CONTRACT OR GRANT NO. AF 19628-68-C-0110	9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TR-68-452		
b. PROJECT NO.			
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)		
d.			
10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Command Systems Division, Electronic Systems Division, Air Force Systems Command, USAF, L G Hanscom Field, Bedford, Mass. 01730	
13. ABSTRACT The results of the evaluation of the JOVIAL Language as specified in Air Force Manual (AFM) 100-24 are contained in this report. This evaluation was based primarily on experience of users of JOVIAL Language dialects. The goal of this evaluation was to recommend deletions, retentions, modifications, and extensions to the JOVIAL language as specified in AFM 100-24 based on the users experience. The methodology of the evaluation consisted of collecting user experience data by means of a "JOVIAL Application Questionnaire" and interviews, and evaluating this data based on criteria established and documented in the "Approach for Change". This report contains a list of JOVIAL features recommended for deletion and retention and detailed specifications of recommended modifications and extensions to the JOVIAL language. In addition, the report contains the detailed interview notes and questionnaire responses which were the basic data used to arrive at the recommendations.			

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer Programming Language Programming Language Evaluation JOVIAL Evaluation						