JI 15-MAY-74 12:58 30746

response to Marcia

Marcia,

(1) I recieved your message today requesting an early response - however, you never said response to what - did part of the message get lost?

(2) Could you also please delete the idents JEAN and NEWS from thee ident system as they are no longer active nor do they have any functional utility at this time,

Thanks,,,,,,,Jean 1

response to Marcia

(J30746) 15-MAY-74 12:58; Title: Author(s): Jean Iseli/JI; Distribution: /MLK; Sub-Collections: NIC; Clerk: JI;

NIC # nnnnn
RFC # rrr
references: RFC #624, 625

Mark Krilanovich
UCSB
May 15, 1974 1

More on On-Line Hostnames: a Response to RFC 625

2

One of the reasons why I feel FTP inappropriate for the hostnames service is that the application at hand is more of a data transfer than a file transfer (admitedly partially personal opinion), and there currently exists no data transfer protocol. I would like to point out that there is a precedent for creating a new server process for this type of application, namely the host status protocol (socket 17). It seems to me that the hostnames service is very similar to the host status service, and that a separate socket for it is just as appropriate. 3

RFC 625 defends the use of FTP for the on-line hostnames service by advancing the opinion that FTP has been set up with the intent of providing a "file transfer mechanism that everyone can use for a variety of needs without further programming required." This indeed was the goal, but whether FTP be good or bad, it clearly cannot solve EVERY need. The point was also made that if one felt FTP fell short of its goals, one should work to improve it, rather than "making end runs around it." With this I heartily agree; there are few things worse than someone who gripes about the current state of affairs but makes no suggestions to better it. I am at present doing the best I can to correct what I believe to be the inadequacies of FTP; witness RFC 624. 4

The claim was made that it is easier to check out data or data transfer problems if the data is in ASCII than if it were in binary. This is undeniably true, but ease of checkout, it seems to me, should nearly always take a back seat to efficiency of operation once checked out. The claim that characters are easier than binary data for humans to look at could be applied equally well to the host-to-host protocol, but I doubt that anyone, even when NCP's were being checked out, would have proposed use of ASCII or EBCDIC for host-to-host protocol. 5

More on On-Line Hostnames: a Response to RFC 625

(J30747) 15-MAY-74 13:28; Title: Author(s): Mark C. Krilanovich/MCK;
Distribution: /JAKE; Sub-Collections: NIC; Clerk: MCK;
Origin: <UCSB>MOREHOSTNAMES.NLS;4, 15-MAY-74 13:21 MCK ;

NJN 16-MAY-74 07:23 30748

USING and the new NIC

We may have just enough time to arrange the next meeting before they sweep the rug out from under us.

NJN 16-MAY-74 07:23 30748

USING and the new NIC

Dave--
By now you have seen what is happening to the NIC services come July.
CRASH!!!! We may both be out of network work very soon. With no NIC
nls services, journal services, etc. it will be well-nigh impossible
to do any USING work, and I get the strong feeling that ARPA-IPT not
only does not want to support the individuals inn the group, but will
probably be reluctant to even support a USING account on OFFICE-1.
Christ, they are not even going to publish RFC's anymore. This all
makes me very nervous, but we can talk more about it when I see you.
--N. 1

USING and the new NIC

(J30748) 16-MAY-74 07:23; Title: Author(s): Nancy J. Neigus/NJN;
Distribution: /DHC; Sub-Collections: NIC; Clerk: NJN;

Dirk--
I found out about the NIC minutes before reading your note. I knew they had lost their funding and then, as now, I was very upset. I really like using NLS. Is there any chance it could be made available at BBN? I intend to ask people here about that.
I looked at statement 0 of your message and saw all the new features the journal system has. I wish I could use it.
Dreams: Senoi-Wise ?
Closings: I am sure it is not true for Craig.
Since this is the old NLS... no closing, N. 1

(J30749) 16-MAY-74 07:32; Title: Author(s): Nancy J. Neigus/NJN;
Distribution: /DVN; Sub-Collections: NIC; Clerk: NJN;

message

ISI Confessions have been changed to the 22 May, 1

message

(J30750) 16-MAY-74 07:35; Title: Author(s): Anna A. Cafarelli/AAC; Distribution: /RADC; Sub-Collections: RADC RADC; Clerk: AAC;

AAC 16-MAY-74 07:45 30751

Info

Message for your Information: The annual filing of supplementary DD Forms 1555 must be accomplished by 31 Jul 74, with information current as of 30 Jun 74. Due to the 30th falling on Sunday this year, time is extended to 1 Jul 74. This requirement pertains basically to civilian personnel classified at GS-13 or above. 1

AAC 16-MAY-74 07:45 30751

Info

(J30751) 16-MAY-74 07:45; Title: Author(s): Anna A. Cafarelli/AAC; Distribution: /RADC; Sub-Collections: RADC RADC; Clerk: AAC;

message

Message for your information: On Form 2 - Charge 1 hr Training to Job order No, 9994TRNG to all those who attended the Movies OPSEC/COMSER, 1

message

(J30752) 16-MAY-74 08:09; Title: Author(s): Anna A. Cafarelli/AAC; Distribution: /RADC; Sub-Collections: RADC RADC; Clerk: AAC;

This is a sample message for you to see how they look when you recieve it in your initial. 1

(J30753) 16-MAY-74 11:43; Title: Author(s): Joe P. Cavano/JPC; Distribution: /AAC; Sub-Collections: RADC; Clerk: JPC;

Thoughts on the PSO

These are some quick impressions on what problems we have with the PSO and even some ideas for solving them (How about that?),

Thoughts on the PSO

With the creation of the PSO, we have, in effect, created a secretarial pool. After my first official encounter with this new organization, as well as many unofficial discussions with all the members of the PSO, a few things are shaping up in my mind. The most apparent is that we have a secretary pool with a noticable lack of secretaries. Let's run-down the membership: 1

Bobbie -- the only true secretary by training (and maybe by inclination as well). 1a

Anne -- a temporary hire and I confess that I do not know under which job classification she falls. 1b

Donna -- a UC student hired for database work and now put into the PSO. 1c

Sharon -- another UC student and it is my understanding that she works on WWMCCS alone and if so, is not a true PSO member. 1d

Duayna -- will soon be in a full-time capacity but I'm not sure under what classification. I know she has taken typing in school but I don't know what other secretarial training she has had. 1e

The benefits to be obtained from this Public Support Organization (as found in Kennedy's paper) are increased productivity, improved response time and a higher quality product. Now I realize that this is just getting started as an experiment but reflecting on the skill levels of the people involved, I think we are asking an awful lot from this group. This group is not a trained collection of secretaries, yet we asking them to perform as such. In addition, they are instructed to use an On-Line Computer System (NLS) that requires even more training and more exact procedures and more problems (what do you do when the machine is down or you can'T log on?). 2

Bobbie has shown me the current issue of "Modern Office Procedures" in which the execetive editor, John Dykeman,lists a key element of successful system implementation: 3

"Successful systems owe their success in great part to TRAINED people who know how do their jobs, know how their jobs relate to others, and what their jobs contribute to the success of the system. Many new systems change more than procedures; they often transform the very structure of the office organization. These dramatic changes require more education and retraining than ever to enable workers to understand objectives, procedures, machines and the roles of others". 3a

Unless we supply some comprehensive training on a number of different levels, this experiment will not be much more effective than it is now. Others may not view this situation as much of a problem but I think training is needed in secretarial skills and in NLS. How the function of the PSO relates to ISIM should also be stressed. I do not know what training is available for secretaries here on base but maybe someone like MItzi can be of help. Branch and Section leaders should be able to help explain the role of the PSO in relating to the jobs of other people in the branch (if this has been attempted before, I would guess that it hasn't been too effective). 4

Additional training is needed for this group in NLS. I have three ideas that may be of some use in helping us deal with this aspect of the problem. 5

(1) Formulation of a test to show what areas of NLS are lacking in the PSO. There is always an evil connotation to any kind of test but this one will be attempted in the classical sense of helping the person learn by pinpointing areas of weakness. A test like this would be difficult to construct and it won't be foolproof but coupled with some insight on what types of NLS capabilities apply to a PSO group and we might have a better idea of where to concentrate our training efforts. 5a

(2) One avenue for getting the training accomplished is by some SRI troops, assuming we still have provisions for training from them. I talked with Jim Bair about this, and he said training will have to be requested from him rather than from his own initiative. Most of his training seems to be at a pretty basic level that is more appropriate for newcomers. We need advanced training for people who have been on the system for a while and have gotten into bad habits or narrow grooves of doing things. A possible impact on training will be the conversion to the new command language if this change-over is scheduled soon. If it is, we might be able to disguise our training under the cover of that. If the date for that is too far away, we must take steps sooner. 5b

(3) Of course, another way to provide training for the PSO is for guys in the section to do it under a formal plan. Although questions can always be answered and help provided when needed, more formality is necessary to ensure that everyone is kept informed of all the instructions. Otherwise, we won't be able to keep track of who covered what and who attended, etc. This is only one step up from what Stoney has been trying to do on his Thursday afternoon sessions. 5c

Thoughts on the PSO

(J30754) 16-MAY-74 12:29; Title: Author(s): Joe P, Cavano/JPC;
Distribution: /EJK DLS RFI RBP JLM ELF; Sub-Collections: RADC; Clerk:
JPC;
Origin: <CAVANO>PSO,NLS;1, 16-MAY-74 12:21 JPC ;

RJC 17=MAY=74 06:54 30755

jovial crap

<CARRIER>JOVIALCHAP1EDIT,NLS;1, 17=MAY=74 06:10 RJC ; 1

Chapter 1 1a

INTRODUCTION 1a1

1,1 Purpose of the Manual 1a1a

The purpose of this manual is to describe the 1973 version of the JOVIAL Computer Programming Language, and to establish standard language specifications upon which the acquisition of compilers for the language can be based, The JOVIAL 73 (abbreviated J73) language is to be considered a replacement for the previous standard, JOVIAL (J3), defined by AIR FORCE MANUAL AFM 100=24, dated 1967 June 15, with amendments thereto, 1a1a1

1,2 Scope and Changes 1a1b

This manual contains the complete set of JOVIAL (J73) language features, The scope of these language features is designed to provide both effective support of today's processing requirements and evolutionary growth as future system requirements dictate, Implementation of the full J73 language is not intended at this time, A basic set of J73 language features is being identified for standard implementation by all compiler systems, Methods of extending the basic set of language features have not yet been determined, Existing J3 programs may not be completely converted to the J73 language because of machine dependencies and resultant changes in language features, Conversion requirements and aids should be considered in conjunction with compiler acquisition for each replacement system, Using activities are requested to submit recommended changes, additions, and deletions to the manual in sufficient detail to permit both a technical and economic evaluation, AFR 300=10 prescribes both policy and procedures for using standard computer programming languages (i,e,, COBOL, FORTRAN, JOVIAL) and for specifying computer programming language compilers, 1a1b1

1,3 Overview and Objectives of the Language 1a1c

JOVIAL 73 has developed out of nineteen years of study and experience with regard to appropriate programming languages for command and control applications, JOVIAL has also been found to be well suited to the programming of many other applications including general scientific and engineering problems involving numeric computation and logically complex problems involving symbolic data, Because of its wide applicability and the optional control it provides over the details of storage allocation, JOVIAL is especially suitable for problems requiring an optimum balance between data storage and program execution time, The earliest versions of JOVIAL borrowed heavily from ALGOL 58, This latest version incorporates features permitting the design and utilization of the most sophisticated data structures, and at the same time simplifies the manipulation of elementary forms==the sort of manipulation that typically involves over 95% of computation time (Knuth, D,E,, "Software, Practice and Experience", Vol, 1, pp, 105=133, 1971, John Wiley & Sons, Ltd,), 1a1c1

,1 The prime motivation for the development of JOVIAL is the desire to have a common, powerful, easily understandable, and mechanically translatable programming language, suitable for wide=range applications, Such a language must be relatively machine independent, with the power to express logical operations and symbol manipulation as well as numerical computation, A JOVIAL "program:declaration describes a particular solution to a data processing problem, meant to be incorporated by translation into a machine language program, The two main elements of this description are: 1a1c1a

a, A set of "data:declarations, describing the data to be processed, 1a1c1a1

b, A set of "statements, describing the algorithms or processing rules, These two descriptive sets are, to a great extent, mutually independent, so that changes in one do not necessarily entail changes in the other, Further, the pertinent characteristics of an element of data need be declared only once and do not have to be repeated with each reference to the data, 1a1c1a2

,2 One of the further requisites of a programming language intended for large=scale data processing systems is that it include the capability of designating and manipulating system data, as contained in a communication pool (compool), A compool serves as a central source of data description, communicating changes in data design by supplying the compiler (or assembler) with the current data description parameters, thus allowing automatic modification of references to changed data in the machine language programs. Though highly desirable for any data processing system, a compool is a vital necessity for large-scale systems where problems of data design coordination between programmers are apt to be otherwise unsolvable. 1a1c1b

.3 JOVIAL is a readable and concise programming language, using self-explanatory English words and the familiar notations of algebra and logic. In addition, JOVIAL has no format restrictions and has the ability to intermix "comments among the "symbols of a program and to define notational additions to the language, the only limit to expressiveness is the ingenuity of the programmer. A JOVIAL program may thus serve largely as its own documentation, facilitating easy maintenance and revision by programmers other than the original author. 1a1c1c

.4 The convenient subordination of detail without loss afforded by JOVIAL also contributes to readability and expedites the task of uniting programs. One simple JOVIAL "statement can result in the generation of scores of machine instructions which might normally take hours to code in a machine-oriented language. This reduction in source program size proportionally reduces the opportunity for purely typographical errors. Such errors are much more obvious when they do occur, due to JOVIAL's readability. Since many coding errors based on the idiosyncrasies of computer operations are eliminated, experience has shown that JOVIAL programs may be written and tested, even by neophyte programmers, in less time than previously required with machine-oriented programming languages. 1a1c1d

.5 Computer users are often faced with the necessity of producing large numbers of computer programs in short periods of time. A readable language such as JOVIAL alleviates the heavy burden this places on the existing programming staff, by permitting augmentation with relatively inexperienced programmers. 1a1c1e

.6 JOVIAL simplifies and expedites the related problems of training personnel in the design of data processing systems and the development of computer programs for such systems. Although JOVIAL was designed primarily as a tool for professional programmers, its readability makes it easy for nonprogrammers to learn and use, It also helps to broaden the base of JOVIAL users beyond those engaged in actual programming, 1a1c1f

,7 The objectives of standardizing JOVIAL are as follows: 1a1c1g

a, To attain a greater degree of inter-system compatibility, 1a1c1g1

b, To provide clear guidance to the computer manufacturing community in the production of computer-based systems, 1a1c1g2

c, To use existing programs and ease the transition when upgrading to new computers, 1a1c1g3

d, To improve the productivity of programmers, 1a1c1g4

e, To establish a base for language improvement, 1a1c1g5

f, To reduce the cost of retraining programmers when transferred between facilities, 1a1c1g6

g, To establish a training requirement on which to base a comprehensive skill resource development program, 1a1c1g7

1,4 The Descriptive Metalanguage for JOVIAL 1a1d

One purpose of this manual is to specify a language, The purpose of the language is to specify algorithmic processes for the solution of computational problems, We must carefully distinguish between the elements of the JOVIAL language and other objects, including the objects a JOVIAL *program:declaration discusses, _A, _B, _C, _B+C, and _A=B+C are five structures in the JOVIAL language, There are, however, an infinite number of structures in the JOVIAL language, In order to speak about them all we need to classify them, We give names to the classes of JOVIAL structures and we distinguish them from all other objects by writing them in italics, The classification schema and the names of classes used in this manual are arbitrary, JOVIAL 73 can be validly described using other classification schemata and/or class names, 1a1d1

,1 Every class of structures in the JOVIAL language that we discuss in this document is named by a word in italics or by a phrase in italics with colons (in italics) between the words of the phrase. We do not distinguish between a class and a general element of the class. We use plurals in italics when we mean several elements of the class. Italics are only used in this manual for class names and to number the syntax equations in Appendix A. Thus, "letter is a class (having 26 members) of elements of JOVIAL. A "letter is also a member of that class. "Name is a class (having infinitely many members) of elements of JOVIAL. A "name is also a member of that class. We use the phrase "metalinguistic term" to mean one of these italicized words or phrases. Every metalinguistic term (except "system:dependent:character) is defined in terms of other metalinguistic terms and the 59 elements of the JOVIAL alphabet. By substitution, every metalinguistic term is ultimately defined in terms of the 59 elements of the JOVIAL alphabet (and "system:dependent:character). 1a1d1a

.2 The definition of a metalinguistic term is called a "syntax equation" or a "metalinguistic equation". Several notational devices are needed in constructing syntax equations. The syntax equations occur throughout the document and are all gathered together in Appendix A in alphabetical order. In fact, Appendix A may be considered the syntactic specification of JOVIAL 73. In Appendix A, each heavily black-bordered box (except one) contains the definition of a single metalinguistic term. Each syntax equation is preceded, in its box, with a sequential number in italics, followed by a colon, followed by a list of the numbers of the syntax equations in which this metalinguistic term is part of the definition. 1a1d1b

.3 Following the metalinguistic term being defined is the definitional operator: 1a1d1c

::= 1a1d1d

Following the definitional operator is the definition, consisting of elements of the JOVIAL alphabet (the "signs of JOVIAL), metalinguistic terms, and metalinguistic symbols indicating choice, repetition, and continuation. Many definitions contain optional elements or mandatory choices. Braces ordinarily denote a choice. One line must be selected from among the lines within the braces in order to satisfy the definition. If there is only one line within the braces, it must be chosen--the braces then only indicate the extent of application of a repetition operator. 1a1d1e

Brackets denote an option or an option and a choice. The line within the brackets may be included or omitted. If there is more than one line within brackets, zero or one of the lines within may be used to satisfy the definition. "Brackets are elements of the JOVIAL alphabet, all of the same size. Brackets are distinguished from "brackets by being considerably larger (and of various sizes). Arrows are used to indicate continuation of a line. If a line is too long for the page (or the space available within braces or brackets) an arrow is placed at the right of the first part of the line and is repeated at the left of the continuation line. In one or two places vertical arrows are used for similar purposes where a column (a stack of lines within braces) is too long for the page. There are two repetition symbols. means that the preceding element of the definition may be repeated an arbitrary number of times. means also that the preceding element may be repeated, but that "commas must be inserted between occurrences of the repeated element. If the repetition symbol follows a metalinguistic term, it is that one metalinguistic term that may be repeated. If the repetition symbol follows a right bracket or a right brace, it is the entire structure within the brackets or braces that may be repeated. A bracketed structure followed by a repetition symbol means "use this structure zero or more times, choosing any one of the lines herein, independently, for each occurrence." A braced structure followed by a repetition symbol means the same except that "zero or more times" becomes "one or more times." 1a1d1f

.4 There is no terminator symbol for a syntactic equation. One ends where another begins or where there is nothing left in the box. In a few of the boxes there are some anomalies. Syntactic equation 144 defines "mark. Opposite each "mark is a metalinguistic term. This association serves to define each of these metalinguistic terms, as the "mark to its left. Opposite "space is only space. That's the definition of "space, the "mark indicated by not marking the paper. Syntactic equation 172 defines "pattern:digit. It also gives tabular information involved with the significance of "pattern:digits. Syntactic equation 190 defines "relational;operator and gives a phrase for each "relational:operator indicating its meaning. Box 234 defines "system;dependent;character by means of a prose discussion. Syntactic equations 247 and 248 are in one box. Each is a definition of "variable in terms of different collections of covering sets. And equations 94 and 95, for "format:list, are in one box. 1a1d1g

.5 Leading and trailing spaces in the definition of a metalinguistic term are of no significance. Spaces between the "symbols of a definition may or may not be significant; the body of this manual clarifies the issues. Certainly, if there is no space between elements of the definition, then no "space is permitted in the corresponding positions in a "program:declaration. For example, _BEGIN must not be rendered as _B _E _G _I _N or as _BE _GIN. 1a1d1h

.6 The syntax equations are not completely correct. There are actually limitations on the seeming generality of the syntax equations. The limitations that must be observed to maintain syntactic integrity are stated in the text. In addition, the text tells what the programmer can do with the syntax and explains the meanings of all JOVIAL constructs. 1a1d1i

1.5 JOVIAL "Characters, Examples 1a1e

Anything in a syntax equation that is not in italics is composed of JOVIAL "signs, the actual alphabet used to write a "program:declaration. These "signs (and "system:dependent:characters) are used also in examples illustrating what may be written in substitution for a metalinguistic term. Examples and metalinguistic terms are never hyphenated for the sake of composing the type in this document. A metalinguistic term never continues from one line to the next in a syntax equation. In text, however, a multiword metalinguistic term may start on one line and continue on the next. In this situation, the italicized colon at the end of one line is repeated at the beginning of the next line. "Colon happens to be one of the JOVIAL "signs. The JOVIAL "colon is not in italics and is always separated by at least one space from any italicized word. The metalinguistic colon is closely pressed on both sides by words in italics. 1a1e1

.1 Metalinguistic terms (the words and phrases in italics) represent structures that can be understood and translated by a JOVIAL compiler, or at least they represent elements of such structures. A *program:declaration can be understood by a compiler and translated into computer instructions. *Simple:statements and *table:declarations are elements of *program:declarations. The translated version of a *program:declaration and the structures it manipulates, however, are an entirely different class of objects. The collection of computer instructions is known as a "program." The word is not in italics because the thing it represents does not exist in JOVIAL. JOVIAL can contain *program:declarations; it cannot contain programs. In a similar manner, a *table:declaration, upon being processed by a compiler, gives rise to a structure, known as a "table", to be manipulated by a program. 1a1e1a

.2 *Program:declaration and *table:declaration are distinguished from program and table both by the use of different type fonts and the use of the word "*declaration." With many terms, the distinction is only made by means of type fonts because the use of extra words would make the explanations awkward. For example, a *variable is part of a *program:declaration, whereas a variable is a value that can be set, used, and changed by a program at different times. 1a1e1b

1.6 Notational Symbols, System-Dependent Values 1a1f

In various parts of this manual, various numeric values that may change from time to time, or that are system dependent are represented by letters or character combinations after the manner of algebraic notation. The meanings of these notational symbols are given where they are used. They have no pervasive meaning and are to be considered valid only in the local context where they are used. 1a1f1

.1 Knowledge of many of the system-dependent values is vital to a sufficient understanding of the environment to enable the programmer to construct valid and useful *program:declarations. Such information is not available at this writing and is not appropriate to this manual. This information must be made available in other documentation. 1a1f1a

1.7 One-Dimensional Nature of a Program 1aig

Regardless of the forms used for coding, the input medium, or the arrangement of the coding on that medium, the language definition considers a JOVIAL "program:declaration to be a continuous stream of JOVIAL "signs. 1aig1

1.8 Syntax and Semantics--Illegal, Undefined, Ungrammatical 1aih

This manual gives complete specifications for writing legitimate JOVIAL "program:declarations, except for the necessary system-dependent values and compiler capacities, explains in detail how the particular compiler deviates from these specifications, and lists and explains all error messages that the compiler may generate. 1aih1

.1 For a "program:declaration to be legitimate, it must be meaningfully structured in accordance with the specifications in this manual. If the "program:declaration or any part of it fails to meet these requirements, it is of small concern whether it is called illegal, undefined, or ungrammatical. 1aih1a

.2 It often happens that compilers do not reject certain illegal or undefined structures, but compile them instead, giving results that the programmer considers appropriate. It is recommended that programmers avoid exploiting these quirks, since there is no guarantee that a new version of the compiler will exhibit the same eccentricities. Using such discovered idiosyncrasies leads to extra work in reprogramming when transferring the work to another computer or when an updated compiler replaces the old one. 1aih1b

.3 As part of the structure of a JOVIAL "program:declaration, nothing is permitted by unstated implication. If it is not prescribed by this manual (or other documentation in the case of system-dependent features), it is not legitimate JOVIAL code. In the matter of exceptions to prescribed forms, nothing is prohibited by innuendo. All exceptions are explicitly stated. 1aih1c

.4 The document is to be taken as a unit. All sections, all figures, the list of syntax equations, and the index-glossary are interrelated. 1aih1d

jovial crap

(J30755) 17-MAY-74 06:54; Title: Author(s): Roberta J, Carrier/RJC; Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;

jovial crap

Contains edited corrections, All yours! Let me know when I can delete, For time being I will keep,

<CARRIER>JOVIALCHAP2EDIT.NLS;1, 17-MAY-74 06:14 RJC ; 1

Chapter 2 1a

ELEMENTS 1a1

2.1 Introduction 1a1a

A "program:declaration written in JOVIAL consists, basically, of "statements and "declarations. The "statements specify the computations to be performed with arbitrarily named data. "Simple:statements can be grouped together into "compound:statements in order to help in specifying the order of computations. Among the "declarations are "data:declarations and "processing:declarations. The "data:declarations name and describe the data on which the program is to operate, including inputs, intermediate results, and final results. The "processing:declarations generally contain "statements and other "declarations. They specify computations, but they differ from "statements in that the computations must be performed only when the particular "processing:declaration is specifically invoked by "name. In addition to "statements and "declarations, there are "directives which serve various purposes. They designate externally defined "names the compiler is expected to recognize, they control selective compilation of various "statements and "declarations, and they provide information the compiler needs in order to optimize the object code. The "statements, "declarations, and "directives are composed of "symbols, which are the words of the JOVIAL language. These "symbols are, in turn, composed of the "signs that constitute the JOVIAL alphabet. 1a1a1

.1 The general order in which the elements of a "program:declaration are introduced in the preceding paragraph represents the general order in which one looks up definitions when trying to clear up a question. The definitions in this manual are introduced, however, in the opposite order. Such arrangements lead to complaints that one must "read the book backwards." This comment arises from the process of looking up a form in the table of contents, then turning to the last chapter where it is defined in terms of earlier defined forms. These more elementary forms are then found via the table of contents in an earlier chapter. And so forth. The document is primarily arranged for the use of a reader rather than for reference, Difficult as this may be for reference use, the opposite arrangement is much more difficult for a reader, 1a1a1a

,2 An index=glossary is included which facilitates reference, The index=glossary answers many questions directly, In other cases, it references syntax equations and sections by number, 1a1a1b

2,2 Spaces and "Spaces 1a1b

It is important to distinguish between a "space, an element of JOVIAL, and a space, an element of our descriptive language, JOVIAL is written using "symbols, the words of the language, The "symbols are composed of "signs, the elements of the JOVIAL alphabet, In general, "symbols do not contain "spaces, The exceptions are pointed out in Section 2,5,2, with respect to "comment, and in Section 2,8,2, with respect to "character:constants, In general, "symbols are separated by "spaces, Exceptions to this are noted in Section 2,10 but these exceptions are permissive; i,e,, it is always correct to put "spaces between "symbols, 1a1b1

,1 The following example is wrong: 1a1b1a

```
_PLXMPY ( 1, 375, =, 75, 5 ,, 7,3 : REAL,
IMAG ) ;                                            1a1b1a1
```

,2 The following examples are right: 1a1b1b

```
a, _BEGIN 1, 3, +5, = 7 END                         1a1b1b1

b, _SL:PLXMPY(1,375,=,75,5,,7,3:REAL,IMAG);          1a1b1b2

c, _SL : PLXMPY ( 1,375 , = ,75 , 5, ,
7,3 : REAL , IMAG ) ;                               1a1b1b3
```

,3 In defining and explaining "signs and "symbols, any spaces included in the metalanguage formulas are ,B=1;not,B=0; meant to be included in the definition, The phrase "string of" implies that there are to be ,B=1;no,B=0; "spaces between the elements strung together, Similarly, phrases such as "followed by", "enclosed in", and "separated by", imply that there are to be no "spaces between the elements concerned, This is the situation (except where explicitly stated to be different) in this chapter, Chapter 2, In

Chapter 3 and beyond, the opposite view is maintained with respect to these phrases, 1a1b1c

2,3 "Signs, Elements of the JOVIAL Alphabet 1a1c

(equ) 1a1c1

,1 "Sign means a "letter, a "numeral or a "mark, "Letter means one of the 26 letters of the English alphabet, written in the form of a roman capital, "Numeral means one of the ten arabic numerals: _0,_1,_2,_3,_4,_5,_6,_7,_8 or _9, (The slash through the zero is only for the purpose of distinguishing it from the "letter _O in definitions and examples of JOVIAL,) "Sign, "letter, and "numeral are defined more formally by means of the syntax equations in the boxes at the head of this section, "Mark is most easily defined by the formal means of the syntax equation in the box above, The box above also contains a metalinguistic term associated with each "mark; this serves to define these terms, 1a1c1a

2,4 "Symbols, The Words of JOVIAL 1a1d

(equ) 1a1d1

,1 The "symbols or words of the JOVIAL language are composed of strings of "signs, in some cases a single "sign, Most "symbols do not contain "spaces, In fact, "spaces serve to separate "symbols from one another, 1a1d1a

2,5 "Primitive, "Ideogram, "Directive:Key, "Comment 1a1e

(equ) 1a1e1

,1 "Primitives may be considered the key words of the JOVIAL language, They are generally used to give the primary meaning of a "statement or "declaration, although some are used for secondary purposes, "Ideograms are generally used as "arithmetic:operators, as "relational;operators, and for purposes such as grouping, separating, and terminating, "Directive:keys are used to state the primary meanings of "directives, "Comments can be used to annotate a "program:declaration--explaining to readers (and often the original programmer) what is going on, 1a1e1a

,2 Notice that a "comment is delimited by "quotation:marks, Therefore, "spaces are permitted within a "comment, but a "quotation:mark is not permitted within a "comment, Also, a "semicolon is not permitted within a "comment, The reason for this is to permit some recovery in case a delimiting "quotation:mark is left off a "comment, If the "comment were not then terminated by the next "semicolon, the entire remainder of the "program:declaration would be turned inside out--the "comments being interchanged with the "statements and "declarations, Even with this rule, failure to terminate a "comment can lead to disaster, If an _END is swallowed up, the entire program structure can be disarrayed, 1a1e1b

,3 The "system:dependent:characters that can be included in "comments (and other structures) are simply those "characters, other than JOVIAL "signs, that the particular system and compiler can read and write, 1a1e1c

,4 Notice that "primitives, "ideograms, and "directive:keys do not contain "spaces, "Spaces are significant in a "program:declaration; usually in that they separate "symbols, "Comments, on the other hand, may contain "spaces, This permits easier reading and writing of the commentary, The "quotation:marks delimiting the "comment provide the necessary grouping so that the "spaces do not cause trouble, 1a1e1d

2,6 "Abbreviation, "Letter:Control:Variable, "Name 1a1f

(equ) 1a1f1

,1 "Abbreviations are specific "letters having specific meanings in specific contexts, usually "data:declarations, The specific uses are documented later on, usually without calling the "letter an "abbreviation, 1a1f1a

,2 The "letter:control:variable is a special "variable having meaning only within a "loop:statement and passing out of existence when the "loop:statement is not being executed, It is explained more fully in connection with the explanation of the "loop:statement, 1a1f1b

,3 Regardless of the syntax in the box above, a "name must not be the same as any "primitive. Notice that a "name must include at least two "signs. The use of the "dollar:sign is system dependent. That is, it provides a means whereby a "name can be designated to have some special meaning in relation to the system in which the compiler is embedded. Such special meanings are outside the scope of this manual, however, and "names containing "dollar:signs are considered the same as other "names herein. "Names do not contain "spaces. An embedded "space would change a "name into two "names or other "symbols. 1a1f1c

2.7 "Number, "Constant, "Status 1a1g

(equ) 1a1g1

.1 The above definitions are obviously not complete, in that several kinds of "constants mentioned in the box are not yet defined. This discussion is mainly concerned with the use of "spaces together with "numbers, "constants, and "statuses as "symbols. 1a1g1a

.2 A "number is a string of "numerals, without "spaces. In some places, a "number can stand alone as a "constant. In other places, particularly "data:declarations, it stands alone as a "symbol but is not considered a "constant. In other places, a "number is part of another "symbol. A case in point is the "character:constant, defined above. The optional "count in a "character:constant is a "number. (In several places, "numbers or other constructs are given new names reminiscent of their uses in those places.) 1a1g1b

.3 A "character:constant is a "symbol. If it begins with a "count, there must be no "spaces between the "count and the first "prime. Between the "primes, the string of "characters may include "spaces, but these "spaces are significant. They represent part of the value represented by the "character:constant. (There are restrictions on the "characters permitted in a "character:constant, discussed in Section 2.8.2). In a "status:constant and a "qualified:status:constant, the "left:parenthesis, the "name, the "colon, the "status, and the "right:parenthesis are all "symbols. "Spaces are permitted between these elements, but not within the "name or the "status. "Space is not pemitted between _V and the "left:parenthesis. All other "constants are "symbols, not containing "spaces. 1a1g1c

2,8 "Constants and Values 1a1h

(equ) 1a1h1

,1 "Character:constants are the means of representing character values to be manipulated by a program, ("Character:variables and "character:formulas are indirect means,) The "characters acceptable as character values are whatever the system will accept from among those given in the body of Figure 2-1, At least the 59 JOVIAL "signs must be accepted, Comparison of Figure 2-1 with Section 2 of USAS X3,4-1968, "USA Standard Code for Information Interchange", shows the graphic characters in identical positions in the two tables, Figure 2-1 includes eight additional columns presently under consideration by standardization bodies, The positions of the "characters in the table are the only correspondence, This manual does not require that internal representation be in accordance with USAS X3,4-1968, If, however, JOVIAL "program:declarations generate messages for transmission to other systems or process messages received from other systems, these messages are required by other directives to conform to USAS X3,4-1968 in their external representation, 1a1h1a

,2 All of the character values indicated in the body of Figure 2-1 can be represented in "character:constants (except for system-dependent limitations), Artifices are required, however, to represent some of the values, Any "spaces within the delimiting "primes, except within a three-"character code, represent characters of value "space", "Primes, "semicolons, and "dollar:signs have special meanings, Therefore, in order to represent a single occurrence of one of these "signs, two of them are used in succession, If a succession of these "signs is desired as part of the value represented by a "character:constant, the entire string is doubled, In summary: 1a1h1b

_2n "primes are used to represent _n "primes, 1a1h1b1

_2n "semicolons are used to represent _n "semicolons, 1a1h1b2

_2n "dollar:signs are used to represent _n "dollar:signs, 1a1h1b3

.3 The reason for doubling the "primes inside a "character:constant is that a single "prime terminates the "constant. The reason for doubling "semicolons inside a "character:constant is the same. Although it is illegal, a single "semicolon terminates a "character:constant, and for the same reason, it terminates a "comment to avoid turning the whole "program:declaration inside out if the correct terminator is omitted. The reason for doubling "dollar:signs is that a single "dollar:sign introduces the codes described in the next two paragraphs. 1a1h1c

.4 Any "character represented in the body of Figure 2-1, if it is acceptable at all by the system as a character value, may be represented by a three "character code beginning with a "dollar:sign. The second "character is a column code from the figure; i.e., any "numeral or one of the "letters from _A through _F. The third "character is any "character from the body of the figure that can be recognized by the compiler. The character specified by such a code is the one at the intersection of the column designated by the column code and the row in which the third "character is found. For example, the percent mark can be represented by any of several three "character codes, including these two: 1a1h1d

_$25 1a1h1d1

_$2U 1a1h1d2

.5 Within a "character:constant, there is a recognition mode for "letters. Initially, the mode is "general", in which all "characters, including uppercase and lowercase "letters, and the three-"character codes are recognized as described above. The mode can be changed to "lowercase", however, by including the two-"character mode code consisting of "dollar:sign followed by uppercase or lowercase _L. All "letters following such a mode code in a "character:constant, regardless of the case used, are considered to be in lowercase. The two-"character mode consisting of "dollar:sign followed by uppercase or lowercase _U sets the "uppercase" mode, in which all "letters are considered uppercase. The three-"character codes pevail, without changing the mode, regardless of the mode. Hence, the appropriate case can be specified for one "letter in a stream of "letters. For example, here are four "character:constants with the value "De Gaulle": 1aihie

_'De Gaulle' 1aihie1

_'D$6E G$6A$7U$6L$6L$6E' 1aihie2

_'D$LE $4GAULLE' 1aihie3

_'$ud$le$u g$laulle' (none of these are ones) 1aihie4

.6 If the "count is present in a "character:constant, there must be no "spaces between the "count and the first "prime, and the "count gives the number of concatenated repetitions of the character values represented within the "primes. Examples: 1aihif

_2'TOM' is equivalent to _'TOMTOM' 1aihif1

_10'*' is equivalent to _'**********' 1aihif2

_3' ' is equivalent to _'   ' 1aihif3

.7 Notice that it is indeed the values that are repeated, not the "characters making up the "constant before evaluation. Thus, _2'T$LOM' is equivalent to _'TomTom'; it is not equivalent to _'Tomtom'. 1aihig

.8 The system may impose a limit on the number of characters in strings representable by "character:constants, "character:variables, or "character:formulas. The size of a "character:constant is the number of characters represented in the value, not the number of "characters between the "primes. 1aihih

.9 "Pattern:constants directly represent values consisting of strings of bits. (Various "variables and "formulas also represent bit values.) The "numeral to the left of the _B in the "pattern:constant is the "order" of the "constant and controls the possible "pattern:digits and affects their meanings. These relationships are displayed in the box above wherein "pattern:digit is defined. The right column contains the possible orders. The "pattern:digits are displayed in the center in braces. The permissible "pattern:digits are only those on the line with or above the selected order. For example, if the pattern is of order _4, only _F and the 15 "pattern:digits above _F are permitted as part of this particular "pattern:constant. The meaning of each "pattern:digit is given in the column on the left, but these are also affected by the order. If the order is _n, then the _n rightmost bits of each pattern represent the meanings of the corresponding "pattern:digits. The optional "count gives the number of concatenated repetitions of the "pattern:digits enclosed in "primes. No "spaces are permitted anywhere within this structure. 1a1h1i

.10 The meaning of a "pattern:constant is the string of bits resulting from the concatenation of the strings of bits (as modified by the order) represented by each "pattern:digit. The size of the "pattern:constant is the number of bits in the string and may be obtained by multiplying the order times the "count (assumed to be _1 if not specified) times the number of "characters inside the "primes. In the following examples, a "pattern:constant on the left is shown with the bit string it represents on the right: 1a1h1j

_4B'7CF03' 01111100111100000011 1a1h1j1

_3B'3120' 011001010000 1a1h1j2

_1B6'10' 101010101010 1a1h1j3

_5B2'R' 1101111011 1a1h1j4

.11 "Numeric:constants represent numeric values. (There are also "numeric:variables and "numeric:formulas.) "Numeric:constants, as well as "numeric:variables and "numeric:formulas, are described in terms of their three possible modes of representation; as integer values, fixed values, and floating values. The compiler may represent "constants in modes other than those indicated by the "program:declaration; as long as the overall effect of the "program:declaration is not compromised. (This principle applies in general; i.e., the compiler can do things differently as long as the result is the same.) Suppose, for example, an "integer:constant is used in a context that requires it to be converted to a floating value. It is far more efficient for that conversion to be done once, at compile time, instead of each time the code is executed. 1a1h1k

.12 An integer value is a numeric value represented as a whole number without a fractional part, but treated as if it had a fractional part with value zero to infinite precision. In this manual, precision means the number of bits to the right of the point in binary representations of numeric values. A "number used as an "integer:constant represents an unsigned integer value. The size of an "integer:constant is the number of bits needed to represent the value; from the leading one bit to the units position, inclusive (value zero has size 1). No "spaces are permitted in an "integer:constant. The system may impose a limit on sizes of integer values. 1a1h1l

.13 Floating values (_v) are represented within the computer by three parts, the significand (_s), the radix (_r), and the exrad (_e), having the following relationships (with regard to the absolute value): 1a1h1m

_v = s x r 1a1h1m1

_s = 0 or _m < s < m x r 1a1h1m2

.14 The radix (_r) and the minimum value (_m) are fixed in any system. Therefore, only the significand and the exrad are saved as representations of a floating value. For a negative value (not a "constant), a minus sign is also saved with the significand. Regardless of the system values of _r and _m, we assume that _r = 2 and _m is one-half. The language permits inquiry into the values of significands and exrads based on the radix and minimum of these values. Therefore, with respect to value, internal representation of floating values exhibits (so far as the programmer can see from results) the relationships: 1a1h1n

_v = s x 2 1a1h1n1

_s = 0 or _1/2 < s < 1 1a1h1n2

.15 "Floating:constants are written with the assumption that, externally, _r = 10, and there is no _m. Thus, the value of a "floating:constant is given as: 1a1h1o

_v = s x 10 1a1h1o1

.16 A "floating:constant must not contain any "spaces. In the syntactic equation for a "floating:constant, the "number (or "numbers) and the "decimal:point (if present) give the value of the external significand, The "scale (with or without its "plus:sign or "minus:sign) following _E gives an exrad (exponent of the radix) to be used as a power of ten multiplier, If the exrad is zero, it and the _E can be omitted, To be a "floating:constant, the "symbol must contain a "decimal:point, or a "scale as exrad, or both, It must not contain an _A; that would make it a "fixed:constant, 1a1h1p

,17 A "floating:constant can contain information relating to the precision of its internal representation, The "scale following _M gives the minimum number of magnitude bits in the significand of the internal representation, In most systems, there are one or two modes of representation of floating values, If the "scale following _M is greater than the maximum number of magnitude bits in any of the system=dependent modes of representing floating values, the "floating:constant is in error, Otherwise, the compiler chooses the mode with the smallest number of magnitude bits in the significand at least as large as the "scale following _M, If there is a choice of exrad size also, the compiler chooses one that can encompass the value of the "floating:constant, These sizes are based on the numbers of bits in the actual representations, not on what may be a fictional assumption that the radix is 2, If the _M and its following "scale are omitted, the compiler chooses its normal mode of floating representation or one that can contain the value, 1a1h1q

,18 A fixed value is an approximate numeric value, Within the computer, it is represented as a string of bits with an assummed binary point within or to the left or right of the string, The number of bits in the string, not counting a sign bit if there is one, is the size of the fixed value, The number of bits after the point (positive or negative, larger or smaller than the size) is the precision of the fixed value, 1a1h1r

,19 A "fixed:constant is seen, in the syntactic equation above, to be an "integer:constant or a "floating:constant (without an _M and its "scale) followed by the "letter _A and a "scale, The _A and its "scale are essential to make the form a "fixed:constant, "Spaces are not allowed anywhere within a "fixed:constant. All that precedes the _A determines the value of the "fixed:constant (which may then be truncated on the right). The "scale after the _A tells how many bits there are after the point. (If the "scale is negative, the bits don't even come as far to the right as the point). The size of the "constant is the number of bits from the leftmost one-bit to the number after the point as specified by the "scale after _A, inclusive. Here are some "fixed:constants, their values, their sizes, and their precisions: 1a1h1s

.20 No "spaces are permitted within a "fixed:constant. The system may impose a size limitation on fixed values. 1a1h1t

.21 "Integer:constants, "floating:constants, and "fixed:constants cannot have embedded "spaces and cannot have negative values. Both of these characteristics are changed for "status:constants and "qualified:status:constants. In "status:constants and "qualified:status:constants, there must be no "spaces within the "status, within the qualifying "name, or between the _V and the "left:parenthesis. There may be "spaces elsewhere within such "constants. 1a1h1u

.22 "Status:constants and "qualified:status:constants represent constant integer values. How they become associated with these values and how they may be used are explained elsewhere. In distinction to "integer:constants, which can only stand for zero and positive integer values, "status:constants and "qualified:status:constants can also stand for unvarying negative integer values. 1a1h1v

2.9 Computer Representation of "Constants and "Variables 1a1i

JOVIAL is designed to be compatible with binary computers, machines in which numeric and other values are represented as strings of binary digits (ones and zeros). The bits (binary digits) of a computer are organized in a hierarchical structure. A compiler may impose a different structure on the computer, but for reasons of efficiency it usually adopts a structure identical to or at least compatible with the structure of the machine. The structure discussed in this section is the system structure; i.e., the structure presented to the programmer by the combination of a particular computer and a particular JOVIAL compiler that produces object code for that computer. 1a1i1

.1 JOVIAL "program:declarations are not completely independent of the system. The extent of dependence, however, is related to the use of certain language features. Dependence is increased by the use of features, such as "pattern:constants and _BIT, that relate to bit representation or those, such as _LOC, that relate to system structure. The value of a "pattern:constant is completely independent of the system, but its use implies knowledge of the representation of other data. It is that knowledge, built into the "program:declaration, that is system dependent. 1a1i1a

.2 Even if such deliberate system dependence is avoided, the programmer must still have knowledge of structure and representation in his system so that he may know the limitations on precision, how his tables must be structured, and how to avoid gross inefficiencies. For example, in processing long strings of character data, it is often much faster to examine and manipulate them in word-size, instead of byte-size, sets. 1a1i1b

.3 A "byte" is a group of bits often used to represent one character of data. The number of bits in a byte is system dependent. Although JOVIAL permits some leeway in positioning bytes, there are usually preferred positions. When referring to these preferred positions, we generally use the term "byte boundary". 1a1i1c

.4 A "word" is a system-dependent grouping of bits convenient for describing data allocation. Entries and tables are allocated in terms of words. Data are overlaid in terms of words. The maximum sizes of numeric values may, but need not, be related to words. Word boundaries usually correspond to some of the byte boundaries. 1a1i1d

.5 The "basic addressable unit" is the group of bits corresponding to each machine location. In many machines, the basic addressable unit is the word. In others, it is the byte. If it is the word, each value of the location counter refers to a unique word. If the basic addressable unit is the byte, each location value refers to a unique byte. In these latter circumstances, it often happens that adresses are somewhat restricted. For instance, it may be permitted to refer to a string of characters starting in any byte, or to double-precision floating values starting only in bytes with locations divisible by 8. 1a1i1e

.6 Integer and fixed values are represented in binary as strings of bits. The number of bits used to represent the magnitude of a value is known as its size and is (in most cases) under the control of the programmer. The position of the binary point is understood and takes up no space. For signed values, the sign bit is an additional bit not counted in the size of the value. For purposes of the use of _BIT, the sign bit is considered to lie just to be left of the most significant bit accounted for by the size of the value. The maximum permissible size of an integer or fixed value is system dependent. The maximum size of a signed integer or fixed value is one less than this system-dependent size and the places where unsigned values of maximum size may be used are restricted; i.e., they must not be used in conjunction with any "arithmetic:operators, nor with the four nonsymmetric "relational:operators (_<, _>, _<=, _>=), and when used with the symmetric "relational:operators (_= and _<>) the other operand must not be signed. 1a1i1f

.7 The compiler determines the sizes of "constants. The programmer usually supplies the sizes of "variables. The size does not include the sign bit for signed data. For unpacked or medium packed data, there may be more bits in the space allocated for an item than are specified by the programmer. Whether or how these extra bits are used is system dependent, but in any case they are known as "filler bits". The sign bit, if there is one, and any filler bits are to the left of the magnitude bits. It depends on the system whether the sign bit is to the left or right of the filler bits. 1a1i1g

.8 The meanings of bit values _0 and _1 are not stipulated, but in most implementations _0 stands for _0 and _1 for _1 in positive values. For negative values, there is considerable variation. All the following are known and acceptable representations of _-12 in an unpacked, signed, integer item declared to be four bits long: 1a1i1h

_111111111111111111111111111111111111111111110011 1a1i1h1

- 1000000000000000000000000000001100 1aiiih2

- 10100 1aiiih3

,9 Floating values are represented by two numbers, both signed. The significand contains the significant digits of the value and the exrad is the exponent of the understood radix, Each system has a standard mode of representing floating values, known as "single precision", with a specified number of bits in the significand and a specified number in the exrad, Many systems have one or a few additional modes in which there are more bits in the significand, the exrad, or both, If there is more than one mode, the programmer can usually choose the mode for each floating value, In the absence of an indication of such choice, the compiler will usually choose single precision, The radix is an implicit constant having a system-dependent value, 1aiiii

,10 Character values are represented by strings of bytes, each byte consisting of a string of bits, The number of bits in a byte is system dependent, The number of bytes used to represent a character value is under control of the programmer, but there is a system-dependent maximum, 1aiiij

,11 A character item that fits in one word is always stored in one word, by the compiler, By use of a "specified;table;declaration, the programmer may override this rule, If it is not densely packed, a character item always starts at a byte boundary, If it crosses a word boundary, a character item always starts at a byte boundary, The programmer must not attempt to override this rule, 1aiiik

,12 An entry variable whose relevent "table;declaration does not describe it as being of some other type is a bit variable, It is merely the string of bits, of a size corresponding to the number of words in an entry, representing the entry, 1aiiil

2,10 "Spaces, "Comments 1aij

The syntactic structures of all "symbols have now been explained, as well as the places where "spaces are permitted or prohibited within them, All further structures that go to make up a "program;declaration are composed of strings of "symbols, It is always permitted to place one or more "spaces between "symbols, It is sometimes required to put at least one "space between "symbols, The criterion is to avoid ambiguity, "Comments can often replace required "spaces, 1a1j1

,1 "Spaces are required in many situations to enable the compiler to detect the end of one "symbol and the beginning of the next, Generally, at least one space is required between two "symbols of any class except "ideograms, but including the "quotation:mark, The rule is exhibited in detail in the following table, The rows are labelled with the ending "signs of the left "symbol of a pair of "symbols, The columns are labelled with the beginning "signs of the right "symbol of a pair, "SR" at the intersection of row and column indicates that at least one "space is required between the pair of "symbols: 1a1j1a

,2 A "comment may occur between "symbols, However, it must not occur within a "definition nor within any "constant, such as a "status:constant or a "character:constant, A "comment may be used instead of the required "space between "symbols unless use of the "comment would cause the occurrence of two "quotation:marks in succession, In fact, only the use of a "comment can bring about the situation indicated by the lower right corner of the table above, Introduction of a "comment between "symbols where a "space is permitted but not required may then require a "space to prevent the "comment from interfering with another "symbol, 1a1j1b

,3 A "comment must not be used where the next structure required or permitted by the syntax is a "definition, That is, a "comment must not follow the "define:name or a "right:parenthesis in a "define:declaration, And a "comment must not follow a "left:parenthesis or a "comma in a "definition:invocation, A "comment, as defined above, must not occur in a "definition delimited by "quotation:marks, 1a1j1c

jovial crap

(J30756) 17-MAY-74 07:09; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;

RJC 17-MAY-74 07:13 30757

jovial crap

oContains edited corrections. All yours! Let me know when I can delete. For time being I will keep.

<CARRIER>JOVIALCHAP3EDIT.NLS;1, 17-MAY-74 06:00 RJC ; 1

Chapter 3 1a

"VARIABLES 1a1

3.1 Concept of "Variables 1a1a

A JOVIAL "program:declaration consists of a string of "statements and "declarations that specify rules for performing computations with sets of data. The basic elements of data are items. Items are named to distinguish one from another. Sometimes, a "name applies to a group of items, requiring indexing to tell one member of the group from another. Several named groups may be subsumed under another group, which is known as a table and which is itself named. Tables and items may in turn be collected in another group called a data block which, again, is named. Space may be allocated these data structures either statically at compile time or dynamically at execution time. 1a1a1

.1 The value of items and other data can be changed in various ways. A data element whose value can be changed by means of an "assignment:statement is known as a variable. Items, then, are variables. Table entries can function as variables, as can parts of items under the influence of the "primitives _BIT and _BYTE. 1a1a1a

.2 A "variable is the designation, within a "program:declaration, of a variable to be manipulated within the computer. The two syntax equations for "variable (above) indicate, first, the type of data involved, and second, the grammatical form of the "variable related to the kind of data structure in which the variable exists. 1a1a1b

3.2 "Named:Variable 1a1b

A "named:variable is a reference to a variable by means of a "name associated with the variable through a "data:declaration. A "simple:variable is a reference (for the purpose of using or changing its value) to a variable declared to be a simple variable; one not declared as a constituent of a table. No "index is involved in a "simple:variable because the reference is to a variable that is one of a kind, not part of a matched set. Use of the "pointer:formula is explained in Section 7,8 1a1b1

,1 A "table:variable is a reference to a variable declared to be part of a table, A table consists of a collection of entries and there is an occurrence of each table item in each entry, An "entry:variable is a reference to the entire entry as a single variable, An "indexed:variable (a "table:variable or "entry:variable) generally includes an "index to select the particular occurrence of the variable being referenced, 1a1b1a

,2 An "index is correlated with a "dimension:list, Every "table:declaration contains a "dimension:list which prescribes the number of dimensions of the table and the extent of the table in each of these dimensions in terms of its "lower:bound and its "upper:bound, (Some of the detailed specifications can be omitted; the defaults are explained elsewhere,) Each "index:component must evaluate to an integer value ("numeric:formulas are explained in Sec 4,5) not less than the "lower:bound and not greater than the "upper:bound in the corresponding position of the relevant "dimension:list, The relevant "dimension:list is, of course, the one in the "table:declaration bearing the "table:name beginning the "entry:variable or in the "table:declaration containing the "item:declaration bearing the "item:name starting the "table:variable, The rightmost "index:component selects the element, of the row selected by the "index:component second from the right, from the plane selected by the "index:component third from the right, etc, 1a1b1b

,3 If the "index is omitted from an "indexed:variable, whether or not the empty "brackets remain, the meaning is the same as if the complete "index were present and each "index:component were equal to its corresponding "lower:bound, In fact, a legitimate form of "indexed:variable is to omit one or more "index:components, marking their positions if necessary with "commas, The meaning of such a form is the same as if each missing "index:component were present with a value equal to its corresponding "lower:bound, The following example shows an "ordinary:table:declaration and three "entry:variables, all with exactly the same meaning: 1a1b1c

_TABLE ALPHA [3:7, 9, 100:157, 0:50]; NULL; 1a1b1c1

_ALPHA [3, 3, 100,0] 1a1b1c2

_ALPHA [, 3,, 0] 1a1b1c3

_ALPHA [,3] 1a1b1c4

3.3 "Letter:Control:Variable, "Functional:Variable 1a1c

A "letter:control:variable is a reference to a variable designated within a "loop:statement to aid in control of execution of the "controlled:statement and to have meaning only within the "loop:statement. It is explained in Section 5.8 in conjunction with "loop;statements. 1a1c1

.1 "Format:variable is a special form that enables a list of values to be converted to character type and assembled into a character value. The details are given in Section 6.1.7 1a1c1a

.2 The above construct selects a string, of the characters denoted by the "named:character:variable, to be considered as the variable to be given a new value. The "named:character:variable can be any "simple:variable or "indexed:variable of character type. The bytes of the "named:character:variable are considered to be numbered, starting with zero at the left. The "numeric:formula following the first "comma is evaluated as an integer and used to select the byte of the "named:character:variable to be considered the leftmost byte of the "functional:variable. If there is no second "comma and no second "numeric:formula, the leftmost byte of the "functional:variable is its only byte. Otherwise, the second "numeric:formula is evaluated and tells how many bytes there are, including the leftmost byte, in the "functional:variable. 1a1c1b

.3 The "named:variable in the above metalinguistic formula can be of any type. The construct selects a string of bits, from the bits denoted by the "named:variable, and treats that string of bits as a bit variable. The bits of the "named;variable are considered to be numbered, starting with zero at the left. The "numeric:formula following the first "comma selects the bit to be considered the first bit of the derived variable. The "numeric:formula following the second "comma (if there is one) determines the number of bits in the derived string (one bit if there is no such "numeric:formula), In signed variables, the sign bit is bit zero and the leftmost magnitude bit is bit one, In unsigned numeric variables, the leftmost magnitude bit is bit zero, In entries, the leftmost bit of the first word is bit zero, In character variables, the number of bits per byte is system dependent, In floating variables, the sign bits of the significand and exrad are included in the bit count, but the arrangement of bits is system dependent, 1a1c1c

3,4 "Format:Variable, "Bit:Variable, "Character:Variable 1a1d

"Format variable is explained in Section 6,1,7, 1a1d1

,1 The construct using _BIT is explained in Section 3,3,3, A "bit:variable denotes a string of bits without consideration of any numeric or other meaning associated with those bits, Almost all "named:variables carry an implication of some data type other than "bit", However, an "entry:variable denotes only the string of bits constituting the entry if the "table:name is not declared to imply some specific data type, 1a1d1a

,2 The construct using _BYTE is explained in Section 3,3,2, The "named:character:variable is a "named:variable using a "name declared to denote a variable (an item or an entry) of character type, 1a1d1b

3,5 Numeric:Variable 1a1e

Any "numeric:variable can be used as a "pointer:variable, The details of the use of "pointer:variables are given in Chapter 7 in conjunction with the discussion of controlled allocation, "Letter:control:variable is explained fully in connection with "loop:statements, Without being explicitly declared, it becomes an "integer:variable through its usage, All "names that can be used as "named:variables are declared as explained in Chapter 7, Some "entry:variables may use "names not associated with any data type, All other "named:variables use "names that are associated with "item:descriptions, These "item:descriptions give the data type among other things (see Section 7,16 for details), One data type is "character" as mentioned above in Section 3,4,2, Another data type is "floating", "Floating:variables use "names declared to be of floating type. The other descriptive terms in "item:descriptions denote "signed" and "unsigned", but we are interested here in other attributes. Signed and unsigned data are also associated with one or two "numbers. The first "number declares the size of the datum, the number of bits in its magnitude. If this is the only "number in its "item:description, the datum is an integer value and the "named:variable denoting it is an "integer:variable. The second "number in the "item:description for a signed or unsigned value declares the precision of the value, the number of bits in its magnitude after the point. If this second "number is present, even if its value is zero, the datum is a fixed value and the "named:variable denoting it is a "fixed:variable. 1a1e1

jovial crap

(J30757) 17-MAY-74 07:13; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;

Continued NIC asccount query

Jim, is there any chance that OFFICE-1 will stay on the network? If so, how much will be charged for an account with 100 pages of disk space. I might be able to continue support separately. Respnse to CERF at ISI. Thanmks. Vint 1

Continued NIC asccount query

(J30763) 17-MAY-74 02:15; Title: Author(s): Vinton G. Cerf/VGC; Distribution: / JHB MDK; Sub-Collections: NIC; Clerk: VGC;

Message to activate the 'System'.

This message is being sent in order to try to have the system recognize that there is a Directory named Nelson RN2, and that Journal mail is supposed to get there. So Far we have been unsuccessful. I hope that this one gets there without the intervention of FEEDBACK. BUT, if necessary... 1

This is a new Directory which has never been exercised enough to get out the bugs. Quite a bit of Journal Mail has been sent to RN2 but NONE has been received. There are two journaled items in the Journal Branch but they were copied into this position by Duane Stone. Messages have been received. When logging in, and after going into NLS, you are told that there are messages and Journal Mail. BUT, nothing ever gets into the File from the journal system. 2

Any attempt to teach the system to Dick Nelson is being held up until the directory is functioning properly. This makes us, and the system, look unwell, to say the least. 3

Message to activate the 'System'.

(J30764) 17-MAY-74 03:40; Title: Author(s): Edmund J. Kennedy/EJK; Distribution: /RN2 MLK FEED DLS EJK; Sub-Collections: RADC; Clerk: EJK;

JLM 20-MAY-74 06:17 30765

16,2 Briefing

oThis is a out line of the pitch I gave to Dr Helimeyer on the 15th of May,

16,2 Briefing

The rugged programming environment is directed at by having the programmer insert assertions into his program that the computer can aid in checking out hie voerall logic ,It is planned to be interactive so the programmer can do his thing where appropiated and the compter its

NEW CAPABILITIES & OPTIONS

COMPILER GENERATION TECHNOLOGY

JOCIT

CHART & WORDS

Next chart depicts how a language could be effectively supported utilizing META-Compiler Techniques.

- Capt Ives -

ON LINE PGRAMMIN AIDS AND DOCUMENTATION PRODUCTION

- ARPA - sponsored on Line Textual Manipulation

By-Product - very sophisticated

Out put processing commands as well as a powerful on line deguggin system and

- As a result High quality documentation can be produced by going from a NLS FILE to a Computer Output Microfilm Capacity

In addition, this software was developed to support on-line programming & it is anticipated that its inherent structure, , With its sophisticated documentation capabilities lend it to Structured Programming as well.

We will get some words from stoney /sri and dick

- The defense mapping Agency

This is a question we have to consider how to handle they are talikng like we will handle efforts in structured programming ,on line text/grahichs,and large file handling and will have 75 money

LANGUAGE SPECIFICATION LANGUAG

LS Get chart from Robbie - Take to arts & drafting.

In addition, as indicated, the use of English to specify a computer language, itself creates numerous ambiguity

OIR,s

Language Control

Compiler optimizarion studies	15k
AVS	82k

Software/Modeling

GCOS investigation	55k
GCOS Simscript	80k
DMS Modeling	60k

Software Design and TEST

Ext of Harvards ECL	50k
Rugged porgramming enviornment	72k
Software Modeling Studies	136k

New starts

The only thing of any matter I guess is interfacing jovial to NLS,It can be said I guess that by so doing we aslthe like of afsds can then begin to see what it is like to be able to bulid source file in a system which allows for extensive editing,documentation and interqction with the program though this will be limited since the jovial compiler is etill on some other machine,

The software medel bit is a reasearch effort aimed at attemptinmg to bulid software models for predicting software reliability figures,

OBJECTIVE

As a result of studies like the ccip-85,WWMCCS acquistion and a general maturing of the data processing field ,it is becoming evident that the major problem facing all large users of computers and cerataintly the Air Force is software production,Programmer productivity and more importanly the reliability of their reslultant product the program are cricial problems to af current and future systemsThough it was known intuitively by both the R&D

community and the operational commands,the CCIP study surface in a very real sense how unfeliable large software systems really are,In a anlaysis of four large software systems done for the study team of which two were large command and control systems it was found that on the average there exisited one error for every 100 lines of code,In addition this is coupled with the downward trend of hardware o at least processor cost with the resultant use use of computers for more and more task prevviously handled by staffs of people and the magintude of the problgins to emerge,

In terms of schedulingg we are all too famialr with the inability of anyone to predict or mange the entire process of a large software productin,

As a result ,5581 is being redirected to focus specifically on the software production program ,Current programs in data management and management inforation systems are being phased down or discontinued,

Within the past year though many of the problems are just beginning to be addressed we do fell in the area of compiler writing we have made siginficant improvements,

ACCOMPLISHMENTS/MILESTONES

WWMCCS JOVIAL COMPILER

Using the jocit commpiler tool we have dev and produced a jovial compiler which has been apted as a operational compiler for the WWMCS comminity

It is consider by many to be the best jovial compiler ever produced by any technique

Using the jocit tool a new version a or a new compiler for a different machine can be produced at 30% of the cost

- Language Spec

As a result of a three year effort a new specification for jovial j73 has beedn comleted

Though not operational there is a d&f item in on it now the DAIS program has adopted it and SAC has agrees to conduct a full scale operatinal test when the complier is imlpemented

- In House using NLS and the computer output microfilm

A high camera ready copy for lrge scale production of the j73 specification was prepared using rhe out put processing commands on NLS,The current estimate of cost saving is 2s per page versous 40s per page,

Even more importantly possibly is that the entire spec is now in such a form that it will be a very simple process to create subsets of the language as well as maintain the changes to the basic document

The SEMANO toll has been used to speify the j3 language and will be used to specify the j73 spec

The navy are using it to specify their CMS 2 language spec

WWMCC,S Support

Working agreement with AFSDC who has been tasked

- Based on our past in-house experience

- WWDMS testing

Designed, coded and will run the test at RESTON next week

- 2108 PLAN

Have prepared

- JOVIAL Language Specification - words from Dick

The next chart depicts major problem areas we belivie exist

Software Validation reliability

Lack of Software discipline probably, the basic problem in Data Processing is that it is an art, As has been often said, if you cannot measure & predict its performance, you do not know what it is - could certainly be applied to software, It is an art, not a science,

Basic lacks such as a Metric, Error Data,

The lack of error data hampers any sientific approach to software design etc,ata software confernece held in Septemeber at the Naval post graduate school it was cited as the single most glaring lack within the field,

Testing

We test for the absence of errors not the presence

- Test until one runs out of money

, Reliability

It is a ironic quirk that software wne fixed does notnecessaryliy get better but often gets worse the next chart show the difference between software reliability and hardware,

Air Force Standard Lang Dev

One basic problem has been the inability to specify, implement and keep current higher order languages which facilitate software production,

Compiler Implementations are different for each computer, sometimes even same model

, still an art, done manually, expensive

,no way to introduce corrections/changes orderly

, language specification does not keep pace with Hardware Developments

, Is written in English, initself a very ambiguous language,

Software Configuration Mgt

- The entire process of scheduling, costing & tracking software development

, Mgt understanding (Use chart on visible code)

, Programmer views himself as an artist

, Documentation & Library functions very demanding

, Technology Transfer

, Net

, Guides to R & D Progress

- Put more structure into Software Design

(Pitch by RCA report - a clean room kind of environment)

Though our current work in this area is covered in the 6,3 program it is belived that considerable more research will be required,The structed programming approach can be likened to a clean room environment where many of the programming nuances are being remobed so a better understanding of what programming really is can take place

System Dev/Modeling

- As a result of our WWMCCS's role, it is quite clear, 3rd generation systems do not support on-line, real time processing

The hardware\software configuration is aimed at the resource allocation problem not at programmer productivity or reliability

In addition-

Though there is no exploratory work in being ,the conviction is growing that hardware architecture studie are required to attempt to indentify hardware configurations which better support programming and reliability of the reulting programs

Program

Since their is very little money available for new starts this year the money is being used to clean up a few loose ends and put us in a postion where we can explore in house the appicability of the SRI software to software production we intend to interface the jovial compiler to NLS so we can bulid source code in NLSand ship it to a JOVIAL compiler and based on a extensive review of the J73 specification by professor Hore there are some revisions mainly in the logic control part of the languages,

We then anticipate that this will then put us in the postion that we can experiment more extensively with NLS as both a documentation tool as well as a programming tool aiming at the establishment of a Software analysis faccility where all of the tools can be brought to bear on the overall problem of software productivity and reliability,The chart also show our tie in the ARPA net where again we can explore or epose AF operatonal users to various programmer aids and documentation toll,

OIR,s

Language Control

Compiler optimizarion studies	15k
AVS	82k

The compiler opt effort is directed at evaulating the 30 odd optimization routine to produce a set of guide lines as to what aspects of the compler can best use it and for waht kinds of effiency,

The AVS effort is a attemept to make testing more visible

one approach is to segment thecode and then the avs can let you know automatically which segments have been exercised as a result of any test programs and data as well as suggest what kind of data is required to exercise the segements her to for not exercised,

Software/Modeling

GCOS investigation	55k
GCOS Simscript	80k
DMS Modeling	60k

The gcost investigation is a attempt to make the tpe a general set of software so any one desiring message handling for things like query etc would be able to achieve a interface in a straight forward manner ,in addition it alos will enable jobs in time sharing to talk to to the batch world of gcos in a much more staight forward manner,

The Simsript effort is a attempt to take advantage of work funded by the intelinge people to bulid a model of some parts of gcos using simsript to so that i working in support of the data system design center weand they could use the model to observe the effect of proposed mods of GCOS to various parts of the WWMCCS software,

We are contuing at a low level to investigate the area of modeling as it could apple to GDMS data base design ,since as it now is all most all of the design decisisons are based om a seat of the pants kind of decision,

Software Design and TEST

Ext of Harvards ECL	50k
Rugged porgramming enviornment	72k

16,2 Briefing

Software Modeling Studies 136k

The ECL work is aimed

COMMENTS

Lete me see,firs FT suggest we have a opening vue graph which sums up the fact that I am briefing a program for only two small efforand their context will follow as well as some inidication of accopmlishments during the past year,

is

They suggest a bullet supporting slide on the jovial accomplishment,including the 600k,plus the others,

The y want a supporting chart on wwmccs accomplshiments ,maybe one showing the extensive moneus involved or a chart showing how one of them itens to modify gcos to fit their problem,

They implied they would like to see a chart on NLS,FT says no I tend to think yes as I give the talk I feek I do not say enough about it when in fact it is one of the two new items we are asking money for,

They want me to explanin away the facility money in vue of the heat I will construst a sentence which says this covers the computer facility which bthe this project as wel as the rest of the center

They r3eally would like if we briefed a couple of over cieling items which are real strong ha ha

cainidates are large file modeling

secure gcos in a multics kind of enviornment

maybe the effort to tie NLS to structed programming but I doubt it as we are not reaaly ready

the mini effort but i dom,t really kmow

the high class terminal effort but againg it might be more dangerous than it is worth though with DMA the requirement is neat

Line graphics in to text pay somebody isn,t a bad idea

Nelson any from him?

16,2 Briefing

Shift the uc contract up to oirs

I think your opening read staement felt too lone see if you canit cut it off earyier

The closing could stand some help ft made a comment but right now I can,t remener

Might grab cooridination chart from sam or robbie

16.2 Briefing

(J30765) 20-MAY-74 06:17; Title: Author(s): John L. McNamara/JLM;
Distribution: /ST; Sub-Collections: RADC; Clerk: JLM;
Origin: <MCNAMARA>PITCH.NLS;2, 15-MAY-74 06:57 JLM ;

Trip Report 1

We held discussions with the DMA people in three main areas; Automated Aids for Documentation, Computer Architecture, Data Management and Computer Graphics. 1a

The personnel we talked with are really the old ACIC, which is responsible for all Air Force charts and maps. DOD created a new agency called DMA, which consists of the Army organization, which produces all terrain kind of maps, ACIC Air Force maps and charts, and the Navy which does all undersea charts. There are two labs who they look for to do their R&D, et al, which is an Army lab and RADC. Up to now, their involvement with us has been through the IR division but as they move into more and more digital data, they are interested in working with either us or the Army. I think preferably both in this area. They work almost exclusively with Joe Diello as of now. They do seem to have a sizable amount of R&D dollars which they must fund lab kind of research as they can only have a R&D staff by reg. 1b

The main focal point for the meeting was A. Kriegel who visited here about a month ago. She works on the R&D staff for ACIC, is relatively new to the job and very interested in funding research in many areas. She believes it will be impacted in the next few years. We were briefed on some of their current R&D programs with Joe Diello. In essence, they are building all kinds of capabilities for digitizing charts, maps, etc. Her questions and concern is how will they handle them both from a software standpoint and hardware. She stressed that for the most part, we were too late for the FY-75 program as most of their budget was planned for and that our meets would hopefully result in FY-76 kind of funding. There are a couple of exceptions; one is an effort which Joe Diello has in-house now where they want him to fund a look at their cardiographic database and the sota and make some kind of plans as to how it can be handled and what kinds of R&D are needed to insure it is being handled. This effort has 75 money on it and IR wants us to handle it. It sounds like a nice opportunity for us to begin to get into the large file problem. I mentioned that I suspected it was more of a software problem than a hardware problem and she quickly agreed. 1c

Data Management 2

I was actually surprised that they were interested in this area so much. I gave our standard GDMS pitch and they apparently related closely to the kinds of problems we talk about. In more detailed discussions, it became clearer why. When they updated from the 7094 to an 1108, they were supposed to go over all of the files, which were under FFS on the 7094. They debated about using the

GDMS on the 1108, but the facility people objected as they were afraid of the core hog appearance of the system. As a result, they are still on the 7094 and are now trying to bring the file across. Worse yet, they are not using any kind of GDMS but are writing special programs for each file. The man in charge, B. Brown, seemed quite aware of the drawbacks but said as of now, he was stuck with a bad decision. He said the worst part was that the users were used to playing with their own database and now they would not be able to since they would be so programmer oriented. 2a

They were very interested in DM-1, and I agreed to send them documentation, and if they desired, some follow-up discussion with them. Quite honestly though, I do not see it as being too promising as they have not even used what they have in GDMS, so it is doubtful if they will see the potential power of DM-1. It is certainly worth a try as they do have money. 2b

Automated Aids to Documenation 3

They were very interested in the potential of using NLS as a tool for speeding up their process for producing things like a manual, which contains all information on all of the Air Fields in the U.S. This is actually updated once a month. The paper shortage is very real to them as well. They are currently examining the whole process for potential help, and we discussed various kinds of text processing systems which might be of potential use to them. I feel follow-up here is a must because I believe that they will be willing to fund an application of this system to their operation as well as any longer range kind of stuff we can identify, like a mini for instance. A. Kriegel indicated she would like to know when Stoney was talking at Wright-Patterson AFB and also at some point in time to arrange a visit by her to SRI. 3a

Action 4

Push Yale Smith on the effort which is in-house right now. I think we could handle it with some admin type help and it would get our foot in the door early. 4a

Find out when you could send her a rough draft of the sota done for SADPR as it might be of use. 4b

I brought back a set of their documentation. Would be neat if we could prepare a little using the system. 4c

(J30766) 20-MAY-74 06:50; Title: Author(s): John L. McNamara/JLM;
Sub-Collections: RADC; Clerk: JLM;
Origin: <MCNAMARA>DMA.;1, 25-APR-74 07:33 JLM ;

FEED 20-MAY-74 16:55 30772

Missing news

there's no money, Please ask ARPAfor further details, 1

Missing news

(J30772) 20-MAY-74 16:55; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /ADO; Sub-Collections: SRI-ARC; Clerk: FEED;

Interaction of substitute and viewspecs.

Ed, The substitute command is designed to be controlled by the current viewspecs. This gives the user a powerful way to control its effect. If you want to have it work on all of the entity specified merely ensure that viewspec w is in force. See Section 5, page 15 of the TNLS User's Guide, statement 10c. Copy to all RADC because this is probably notcommon knowledge. 1

Interaction of substitute and viewspecs.

(J30773) 20-MAY-74 17:10; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /EJK RADC(for your info); Sub-Collections: SRI-ARC RADC; Clerk: FEED;

FEED 20-MAY-74 17:34 30774

Items for NIC attention

(NIC) Items directly concerning the NIC function. 1

ADO 30-APR-74 18:18 30593
missing news
Message: There's no news online.
1a

6-MAY-74 2107-EDT Vint Cerf at AMES-TIP: Net mail from site
BBN-TENEX
Distribution: FEEDBACK
Received at: 6-MAY-74 19:06:23 1b

Jim:
haven't tried guest because office-1 was not responding on
Monday May 6 at 1800 hours. Will try again later this week and
will gripe if I have trouble. Thanks for checking again.
Vint 1b1

15-MAY-74 1610-PDT FEEDBACK: Library services at the NIC
Distribution: FEEDBACK, hughes at MIT-MULTICS, norton,
hhughes,mac at MIT-MULTICS
Received at: 15-MAY-74 16:10:02 1c

Herb,
There have been major changes in the NIC recently by ARPA.
These changes
will virtually eliminate service as it is now known. This will
be formally
announced this week through the System, with copies sent
through the mail to
key people at each site.
Thus, the answers to your questions are very different than
they would have
been a few weeks ago. Yes, we would like to have any documents
for our
hardcopy collection that you would like to share with us. They
would of
course be indexed in our Journal indexes, but that will be
available only to
people that are on line.
Since there will be no service via the NIC after 1 July, it
would not be
useful to establish any kind of abstract library (there is none
now).
However, if you wish to purchase part of the Utility service
from us, we
would be glad to discuss any possibiliites. The service
currently sells for
a minimum of $10 to 40 thousand per year which includes user

support as well
as computer service/time, If you are interested or have any
further
questions, please feel free to contact Jim Norton or myself at
SRI (415)
326-6200 ext, 3614, Thanks for your inquiry, Jim Bair (Head,
User
Development) 1c1

05/15/74 0949-edt HHughes,MAC at MIT-Multics: Net mail from site
MIT-MULTICS
Distribution: FEEDBACK
Received at: 15-MAY-74 06:48:07 1d

i have several questions,
we have here at project mac several documents that we think of
interest, Do you have a hardcopy library and if so could we
send
you copies to be numbered and placed in the library, We noticed
for example that the tenex documentation has nic numbers 1d1

second related question
we have prepared abstracts of MAC technical memos for
journal entry into the NIC
is there already a library of abstracts, If so what kind of
stuff
is in it and would it be appropriate for us to put ours into
it?
or would it be better to simply create a mit-multics file of
documents and abstracts available to everyone 1d2

thanks Herb Hughes 1d3

Items for NIC attention

(J30774) 20-MAY-74 17:34; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /JAKE; Sub-Collections: SRI-ARC; Clerk: FEED;

RJC 21-MAY-74 11:29 30775

J73 - Chapter 4 - Edit Version

oStoney, Chapter 4 is finishe(as far as I'm concerned),,,Remember, tis contains the bold face directives and the monospace directives (monospace - circled items),,,except for the 1 character crap,,,you know what I mean,,,good luck,,,Bobbie

<CARRIER>JOVIALCHAP4EDIT,NLS;1, 21-MAY-74 11:06 RJC ; 1

Chapter 4 1a

"FORMULAS 1a1

4,1 Concept of "Formulas 1a1a

Chapter 3 discusses "variables, the constructs standing for elements of data whose values may be changed, "Formulas are the means for specifying the new values for "variables, "Formulas also generally supply values for any purpose--such as comparisons and other selections of courses of action, Since "constants and "variables denote values they are also "formulas, 1a1a1

,1 Any "numeric:formula can be used as a "pointer:formula, The details of the use of "pointer:formulas are given in Section 7,8, "Value:formulas and "numeric:value:formulas can occur only in "loop:controls, The details of their use are explained in section 5,8, 1a1a1a

4,2 "Constant:Formula 1a1b

A "constant:formula is a "formula whose value can be determined at compile time, once and for all, That particular criterion is somewhat system dependent, In places in this language specification where a "formula is called for, it is only a matter of efficiency whether a "constant:formula is evaluated at compile time or execution time, A "constant:formula, however, can be used in places where this manual calls explicitly for a "constant, The "constant:formula must then be evaluated at the time it is encountered in order properly to compile the "program:declaration, The same consideration applies to a place where a "number is required, but not as part of another "symbol such as a "floating:constant, When a "constant:formula is used to represent a number, it must evaluate to an appropriate integer value, In general, parts of this document which require "constants or "numbers do not reiterate this permission to use "constant:formulas, A "constant:formula is not permitted as part of a "format:list, which is, after all, a second level syntax equation applied to that which is first the value of a "character:formula, 1a1b1

4,3 "Conditional:Formula 1a1c

A "conditional:formula is the "formula following any of the three "primitives _IF, _WHILE, _UNTIL (see sections 5,7 and 5,8 on "conditional:statements and "loop:statements) or the "directive:key _!TRACE, A "formula of any type can be used in these positions, After all operations are performed as called forth in the "formula --bit or byte extraction, shifting, concatenation, function evaluation, comparisons, arithmetic, logical combination, attribute guidance, etc,--the rightmost bit of the result is examined without further conversion, If that rightmost bit is _0 the "conditional:formula represents the logical predicate "false", If the rightmost bit is _1 the "conditional:formula represents the logical predicate "true", This can, of course, lead to machine dependencies if "conditional:formulas contain any operands other than unsigned integers except in "comparisons, For example, a negative integer as a "conditional:formula will lead to a result on a one's complement machine opposite to the result on a two's complement or sign-magnitude machine, The following table indicates the action to take, depending on the value of the "conditional:formula 1aic1

4,4 "Character:Formula 1aid

"Character:constant is explained in Section 2,8,1, "Character:variable is explained in Section 3,4,2, "Character:form is one of the two types of form, explained in Section 4,17,2, A "function:call is the invocation of a certain kind of "procedure:declaration as explained in Section 4,18, A "character:function:call is the invocation of one of these special "procedure:declarations having its effective output parameter of character type, One of the "intrinsic:function:calls (see Section 4,19), the "byte:string:function:call, is a "character:function:call, 1aid1

,1 Any "character:formula represents a value having a size measured in bytes, For its use in the "byte:string:function:call, the bytes of the "character:formula (any "character:formula can be used where indicated as the first "actual:input:parameter in the metalinguistic equation) are numbered starting with zero on the left, With respect to this numbering, the first "numeric:formula (the second "actual:input:parameter) tells which byte of the stated "character:formula is to become the first (leftmost) byte of the derived "character:formula, The second "numeric:formula, if present, tells how many bytes (following consecutively to the right) are to be included in the derived "character:formula, If the second "numeric:formula is missing, just one byte is used, The "numeric:formulas must yield non-negative values, Only the integer parts of these values are used--the fractions are truncated, The sum of the two values must not exceed the size of the first "actual:input:parameter, If the second "numeric:formula (the third "actual:input:parameter) has a value of zero, then the "byte:string:function:call represents a character value of zero size, Such a value as an operand in concatenation leaves the other operand unchanged, It can be appropriately padded in any context in which it might occur, For instance, as a "conditional:formula it would be padded on the left with a single bit of value zero, which would thus become the rightmost bit of the "conditional:formula, leading to the logical predicate "false", As an operand of _AND, OR, etc,, it would become a string of bits of value zero to be combined with the bits of the other operand, Example: 1a1d1a

```
        ALPHA = 'OA2C4E6G8I';                              1a1d1b

        BETA = BYTE (ALPHA,3,5);                           1a1d1c

        GAMMA = BETA <> 'C4E6G';                           1a1d1d
```

,2 In the above sequence of code, _GAMMA becomes zero because _BETA does indeed contain the value _C4E6G, 1a1d1e

,3 The "ampersand is the only operator that can apply to "character:formulas, It means concatenation, 1a1d1f

"character:formula _& "character:formula 1a1d1g

is a "character:formula, Its value is the concatenation of the bytes (all the bytes) of its left operand on the left with the bytes of its right operand on the right, Its size is the sum of the sizes of its operands, Example: 1a1d1h

,4 A "character:formula can consist of concatenations, The ordinary left-to-right rule applies--the two leftmost operands are concatenated first. Then the result is concatenated with the next "character:formula to the right,etc. Ordinarily it really makes no difference if concatenation is done left-to-right or right-to-left, but in cases where the resultant size might exceed system-dependent limits some system-dependent differences might arise. Example: 1a1d1i

(ALPHA & BETA) & (GAMMA & DELTA) 1a1d1j

.5 Notice the "parentheses in the above example. A parenthesized "character:formula is also a "character:formula. The utility of the "parentheses is to change the order of concatenation--operations within "parentheses are performed before the value of the parenthesized "formula is used in further operations. In the above example _ALPHA is concatenated with _BETA, _GAMMA is concatenated with _DELTA and then these two results are concatenated together. A "formula of any type can be used as a "formula of any other type--its value is appropriately transformed. "Parentheses may, at times, be significant in determining the type of a "formula. 1a1d1k

.6 A "bit;formula may be used in a context requiring a "character;formula. The most obvious such context is as the first "actual:input:parameter to the "byte:string:function:call. Assignment to a "character:variable does not make a "bit:formula into a "character:formula. For the use of a "bit:formula in assigning a value to a "character:variable see Section 5.5.1. In concatenation of a "bit:formula and a "character:formula the "bit:formula is stronger--the "character;formula is treated as a "bit;formula. In the "byte:string:function:call, a "bit:formula as the first "actual:input:parameter is padded on the left with as many bits of zero value as are needed to yield an integral number of bytes in the value. The resulting bit string is then considered a byte string and the "numeric:formulas are used to select the desired byte string. For example, suppose that in a system in which bytes consist of eight bits each, there is a "byte:string:function:call requiring _3 bytes starting with byte _1 (the 2nd byte) of a "bit:formula of _35 bits. The following table illustrates the example and shows the resultant value of the "byte:string:function:call 1aid11

4,5 "Numeric:Formula 1aie

"Numeric:constant is explained in section 2,8,11, "Numeric:variable is explained in section 3,5, A "numeric:function:call is the invocation of a "procedure:declaration (see Section 8,4,2) having an implicit output parameter of numeric type, Several of the "intrinsic:function:calls are "numeric:formulas (see Section 4,19), 1aie1

,1 A "bit:formula in a context requiring a "numeric:formula is treated as an unsigned integer value, The string of bits comprising the value of the "bit:formula is considered, without any change, conversion or alteration, as the magnitude of a non-negative integer value, If its size is too great for the use to which it is being put, leading bits are truncated to reduce its size to the maximum that can be used for the arithmetic, conversion, indexing, pointing or formatting, If its size is unknown at compile time it is given a system-dependent default size (if there is any possibility it could be larger) in which the rightmost bits are right justified and any extra leading bits at execution time are zeros, This default size is most likely to be the largest size of unsigned integer with which integer arithmetic may be done conveniently, If its default size is unknown, but its maximum possible size is known to be less than the default size, the maximum possible size is taken as the size of the unsigned integer in the numeric context, 1aie1a

,2 Being in a position to be assigned to a "numeric:variable, being an "actual:input:parameter corresponding to a numeric "formal:input:parameter, or being compared with a "numeric:formula, does ,B=1;not,B=0; impose numeric assumptions on a "bit:formula, The contexts requiring any "formula to be treated as a "numeric:formula are as follows: 1aie1b

a, As an operand to participate in arithmetic, 1aie1b1 b, As an operand to be converted to a numeric in accordance with attribute guidance, 1a1e1b2

c, As an "index;component, 1a1e1b3

d, As a "pointer:formula, 1a1e1b4

e, As an operand to be encoded for "output" in accordance with a "numeric:format, 1a1e1b5

4,6 Arithmetic 1a1f

"Arithmetic:operators are used to specify arithmetic calculation in determining numeric values, The meanings of the "arithmetic:operators are as follows: 1a1f1

_+ Add, 1a1f1a

_- Subtract, (or negate) 1a1f1b

_* Multiply, 1a1f1c

_/ Divide, 1a1f1d

_\ Determine the residue (modulo), 1a1f1e

_** Raise to the power of (exponentiation), 1a1f1f

,1 The syntax equations permit long sequences of "plus:signs and "minus:signs before an operand, The effect of such a sequence can easily be determined by counting the "minus:signs and ignoring the "plus:signs, If there is an even number of "minus:signs, the entire sequence is equivalent to one "plus:sign, If there is an odd number of "minus:signs, the entire sequence is equivalent to one "minus:sign, 1a1f1g

J73 - Chapter 4 - Edit Version

(J30775) 21-MAY-74 11:29; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;

sample program,,,,,,it reformats statements created by the INMES program, making them look like journal documents (ie, senders IDENT in paren's, then the date, then the title of the message, all followed by the message)

thoght you might light to take a look at this; there's nothing really new in it, but it's apparently useful, sample program,.....,it reformats statements created by the INMES program, making them look like journal documents (ie, senders IDENT in paren's, then the date, then the title of the message, all followed by the message)

```
PROGRAM reform
   DECLARE TEXT POINTER sf, pt1, pt2, pt3, pt4;
   (reform) PROCEDURE;
      IF FIND "sf $NP $D '- $L '- $D $NP $PT "pt3 CH $NP "pt1 [SP/':]
      "pt4 < CH "pt2 > THEN
           ST sf _ '(, pt1 pt2, '), "  ", SF(sf) pt3, " ", pt4
SE(sf);
       RETURN(FALSE) END.
    FINISH                                                                 1
```

sample program......it reformats statements created by the INMES program, making them look like journal documents (ie, senders IDENT in paren's, then the date, then the title of the message, all followed by the message)

(J30776) 21-MAY-74 11:35; Title: Author(s): Michael T. Bedford/MIKE; Distribution: /PAN; Sub-Collections: NIC; Clerk: MIKE;

EJK 21-MAY-74 13:25 30777

Job Assignments, short term,

Correlate these with (mcnamara,staffmeet,4)

EJK 21-MAY-74 13:25 30777

Job Assignments, short term.

PREPARE A PACKAGE FOR SOME TERMINALS TO BE BOUGHT BY SRI FOR US. CHECK INTO THEIR CHEEPIE DNLS. (ELF with assist from DLS.) 1

THE MINI CONFESS WILL BE NEXT TUESDAY AT 1030. THE TOPIC WILL BE THE BRIEFING TO HEILMEIER GIVEN BY JLM. THIS WILL BE PRECEDED BY A RUNDOWN ON THE NLS TASKS. (ALL) 2

START THE EFFORT WRITE UPS ON EACH OF THE TASKS. WE HAVE A LIST OF TASKS THAT WE WORKED UP. (MCNAMARA,ISIM,1) ALL THE 1634'S ARE ALREADY IN THE SYSTEM. (ALL) 3

HELP IN FIGURING OUT HOW TO PREPARE A FORMAT WHICH IS USEFUL FOR US. THE MANNING FOR INSTANCE, WHERE IT IS IMPORTANT TO KNOW THE RATE OF MANPOWER EXPENDITURE AND THE PROJECTION. (RBP with assist from DLS, EJK) 4

THE TPO JAZZ IS ABOUT TO START. WE NEED ED LAFORGE TO MANAGE THE TPO DOWN TO THE LAST DETAIL, SO THAT WE DO NOT HAVE THE SAME FORMAT PROBLEM WE HAD LAST YEAR. LAST YEAR'S, IN ITS VARIOUS MANIFESTATIONS, IS ALREADY IN THE SYSTEM. GET THE INSTRUCTIONS ETC.? GET SOME OF LAST YEARS COPIES DISTRIBUTED TO THE PEOPLE WHO WILL BE REDOING IT. GET A HEAD START. (ELF with assist from RBP DLS EJK RJC) 5

WE KEEP GETTING INFO OR REQUESTS ON THE ESP FOR DSC/O PROJECT, WE SHOULD CANCEL THIS THING OUT. (DLS) 6

EXPLORE THE USE OF THE PSO AS A LIBRARY KIND OF OPERATION USING THE JOURNAL SYSTEM. A FAIRLY EXTENSIVE INDEX CROSS-REFERENCES ETC. (RBP with assist from DLS RJC) 7

WE HAVE A LETTER IN HERE FROM DR MAYER ON THE BBN WORK. WE NEED THINK ABOUT OUR POSITION ON THIS WORK SINCE IF WE DON'T PUSH IT IN 5581 IT WON'T BE FUNDED. (EJK with assist from DLS) 8

PREPARE TO BRIEF THE SRI CONTRACT, THE ARPA PART, NOT OURS, THOUGH I GUESS WE WILL HAVE TO MENTION WE ARE USING OFFICE 1. THE OLD PAR BRIEFING RESURRECTED AND NOW CALLED MARS. (EJK with assist from DLS) 9

PREPARE A BRIEF PLAN FOR TRAINING STINSON, SEMARARO, BECKY, MARCELLE, NELSON AND CARM. (EJK with assist from DLS) 10

Job Assignments, short term,

(J30777) 21-MAY-74 13:25; Title: Author(s): Edmund J, Kennedy/EJK; Distribution: /JLM RBP DLS JPC EJK RJC ELF EJK RFI; Sub-Collections: RADC; Clerk: EJK;

Whats up revisited

Hi Bob, everything here is going as usual, A few of us are looking around for new positions, I wont name names but Mel,Larry,George,Roger, and Mike would agrree with the above statement, As for the cutback in NIC services, I agree that it could have been worse but not by much, As I see it, the Nic will no longer serve as a method of interaction for the ARPANETWORKING community (e,g,, working groups) and this was one of the networks strong points, It looks very much like ARPA is out of the networking biz, Oh well,,, I'll try to link to you the next time that your on, See you soon, Frank, 1

Whats up revisited

(J30779) 22-MAY-74 05:17; Title: Author(s): I. Larry Avrunin/ILA; Distribution: /RLL; Sub-Collections: NIC; Clerk: ILA;

comment on the sample program INMES, and on the L-10 manual generally,

The example program that you sent me was too complicated for me to make much sense of, Take the first occurrence of the the CASE construction, for example:

```
CASE lookc( ) OF
   = CA: input( );
   .....etc,
```

Those paren's are throwing me, Also, generally speaking, the program is too long for me to get a grasp of its logical structure, even though I know what it's intended to do,

As a further general comment on the users's manual, both Penny and I have found that it is short of examples of the type that would be useful to someone learning L-10 for the first time, We haven't looked at this in detail, but I think that most of the examples are geared towards someone who knows L-10 and wants to take a look at a partiuclar operating construct, In other words, the examples are geared toward a user who wants to see how to fit an operation into his existing bag of tricks, rather than towards someone who has no bag of tricks and would like to develop some,

I think that a lot more very simple programs like the delete space one, would be appreciated by types like us,

Other than these few difficulties, I think we're getting along quite well, We're looking forward to a chance to work with you in person on some of these problems (either here, or wherever you happen to working out of), 1 comment on the sample program INMES, and on the L-10 manual generally.

(J30780) 22-MAY-74 05:34; Title: Author(s): Michael T. Bedford/MIKE; Distribution: /NDM PAN MIKE IMM; Sub-Collections: NIC; Clerk: MIKE;

Tickler

(maym5) 27 May - Monday - HOLIDAY 1

(mayt5) 28 May - Tuesday 2

Due Date - ISIM/ISIS - Nominations for Specialized Short-Term Courses for 1st half FY-75, 2a

1300 hrs, Branch Chief's Meeting 2b

28 May - Due Date - ISIM - Unsol, Prop, 164-74 "Modeling of Data Management System" COMPLETED 2c

28 May - Unsol Prop 165-74 "A Security Model of Operating Systems" COMPLETED 2d

(mayw5) 29 May - Wednesday 3

Due Date - ISIS/ISM - memo - Authorization to Sign for and Receive Classified Material - Request each Division submit a listing of authorized personnel who will have occasion to sign for reproduction work to 416 CSG/DAPR through RADC/DAP by 3 June 74, The following format is requested: Name Grade Social Security No, Office Symbol Extension Sample Signature, 3a

(mayth5) 30 May - Thursday 4

Commander's Supervisors' call - 30 May - 1000 hrs, - 106 - Auditorium, 4a

Due Date - ISI/FJT, ISIM/R, Iuorno, ISC/M, Kesselman - Subject: WWMCCS - Ltr from AFSC/DLC - Support for System Dmt Notification (SDN) - AF-025-Under Project 2108; Prepare document or cancel mini computer ffort; and finally - 1634 for the Study - 1 Atch RDP ltr, dtd, 3 apr 74 w/atch, 4b

0830 hrs, Branch Chief's Meeting 4c

ISIM/ISIS - University of Michigan brochures - Nominations (AF Form 1152) for FY 75 courses should be submitted to this office (ISI) by 30 May 74, School application forms should be submitted also, Catalog on file in ISI, 4d

Laboratory Activity Reports due today: Bucciero must have them by 1000, ISM must have them by 1100, and DOT must have them by 1600,(JJOURNAL,30511,1:w) 4e

(mayf5) 31 May - Friday 5

CONGESSIONAL VISIT - Donald J, Mitchell, 31st District, N,Y, -

Tickler

Mitchell will visit Bldg 3, with concentration on the "proposed" new laboratory area, The Commander and Joseph G, Vincent, Special Asst to the Commander, RADC will accompany him on RADC portion of the visit, Approximate time of visit: 1300 - 1330 - 31 May 1974 - RB/IS advised to clean area and make it as presentable as possible, 5a

Due Date - ISIS - Final Invention - f30602-73-c-0161, 5b

Due Date - ISIM(R, Iuorno) Final Report, Contract F30602-73-C-0223 for technical review, DMS Test Methods for Security & Restart/Recovery, 5c

Form 2's (employee time expenditures) are due today, 5d

Form 6's (projected manpower) are due today, 5e

Bobbie; Travel figures due by noon, 5f

Tickler

(J30781) 22-MAY-74 07:16; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /RADC; Sub-Collections: NIC RADC; Clerk: RJC;

FAREWELL PARTY - 5 JUNE 74

SAY FAREWELL - 1800 - 2000 hrs, GAFB Officers Club $3,00 per person Stand-up Buffet Pay as you go Happy hour prices first hour Tickets may be obtained by contacting M, Xobos by 31 May 74, 1

FAREWELL PARTY - 5 JUNE 74

(J30782) 22-MAY-74 07:24; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /RADC; Sub-Collections: NIC RADC; Clerk: RJC;

MINIMIZE

Due to MINIMIZE, before making Autovon or long distance calls, permission must be obtained from Branch Chief! Report to Becky for logging the call, after permission is granted, PLEASE COMPLY 1

MINIMIZE

(J30783) 22-MAY-74 08:46; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /RADC; Sub-Collections: NIC RADC; Clerk: RJC;

Chapter 5 1

STATEMENTS 1a

5,1 Concept of "Statements 1a1

"Statements are the operational units of JOVIAL, They describe self-contained rules of computation specifying manipulations of data, and they describe the sequencing, conditional or unconditional, of the execution of "statements, 1a1a

,1 Wherever the syntax says that a "statement can be used, any of the above kinds of "statement can be used, The term "controlled:statement is sometimes used in describing other "statements, A "controlled:statement is not any special kind of "statement but is a required part of a "conditional:statement or "loop:statement, 1a1a1

,2 Any "statement may be used where a "controlled:statement is specified--except for the particular forms prohibited in the description of the "conditional:statement, 1a1a2

,3 The kinds of "simple:statements listed below are discussed in Sections 5,5 through 5,14: 1a1a3

,4 Some "simple:statements such as the "loop:statement, "switch:statement, and "conditional:statement may contain "compound:statements, 1a1a4

5,2 "Null:Statement 1a2

In some language forms defined in this manual, some preliminary structure is followed by a single "statement, A "simple:statement or "compound:statement usually completes the form, If there is no significant "statement desired in such a case, a "null:statement can be used to complete the form, 1a2a

,1 The second form given above for "null:statement is a single "null:statement regardless of whether the optional "semicolon is included, A single optional "semicolon is permitted as a terminator for a "direct:statement, a "switch:statement, a "compound:declaration, or a "compound:statement, If used, this "semicolon merely terminates the "statement or "declaration and is not a "null:statement, Another use for the "null:statement is given in the discussion of "switch:statement, 1a2a1

5,3 "Compound;Statement 1a3

A "compound:statement is a "statement whose essential character is bound up in containing one or more other "statements as a part of itself, A "compound:statement (like a "loop:, "conditional:, or "switch:statement) is a grouping of other "statements, 1a3a

,1 In the syntax equation above, there must be at least one "statement other than a "null:statement between _BEGIN and _END for it to be considered a "compound:statement, "Directives may be included in a "compound:statement (see Chapter 11), The optional "semicolon, if given, does not constitute a "null:statement; it merely terminates the "compound:statement, An example of a "compound:statement that contains a "compound:statement is: 1a3a1

```
BEGIN     IF J > 9 ;                          1a3a1a
  BEGIN   J = J + 1 ;   GOTO OUT ;            1a3a1b
_ END                                         1a3a1c
_END                                          1a3a1d
```

5,4 "Named;Statement 1a4

Any "statement can be named, 1a4a

,1 A "statement;name is defined by attaching the "name followed by a "colon to any "statement, The "statement to which the "name is attached becomes thereby a "named;statement, The "named:statement retains its character as a "null:statement, "simple:statement, or "compound:statement when the "name is attached, and so another "name can be attached, Example: 1a4a1

```
CEASE: DESIST: HALT: STOP;                    1a4a1a
```

,2 The above example is a "stop:statement that has three "names, _CEASE, _DESIST, and _HALT, The "stop:statement is also a "named;statement and a "simple;statement, More examples of naming various kinds of "statements follow: 1a4a2

```
EPSILON:  ZETA = ETA;                         1a4a2a
THETA:    IF  . . . (uncompleted)             1a4a2b
```

```
IOTA:     FOR . . . (uncompleted                       1a4a2c

KAPPA:    BEGIN . . . (uncompleted)                    1a4a2d

LAMBDA:   TEST;                                        1a4a2e

MU:       EXIT KAPPA;                                  1a4a2f
```

.3 A "statement;name may be attached to two structures that are not "statements. Specifically, a "statement;name may precede _ELSE, and a "statement;name may precede the first _BEGIN following _SWITCH, as in these examples: 1a4a3

```
NU:       ELSE     ...    (uncompleted)                1a4a3a

SWITCH XI=3;  OMICRON:   BEGIN  ... (uncompleted)      1a4a3b
```

.4 The effect of references to _NU is exactly the same as if _NU were attached to the "statement following _ELSE. The effect of a "go:to:statement referencing _OMICRON is explained in section 5.12.1. The effect of an "exit:statement referencing _OMICRON is the same as if _OMICRON were attached to the "switch:statement. 1a4a4

5.5 "Assignment:Statements, "Exchange:Statement 1a5

A "simple;assignment;statement specifies that the "formula to the right of the "equals:sign be evaluated and that this value become the new value of the "variable to the left of the "equals:sign. It is permissible for the "variable on the left to occur also in the "formula on the right. In this case, the old value of the "variable is used in the calculations needed to evaluate the "formula. The "formula is evaluated as described in Chapter 4 (see particularly sections 4.15 and 4.16), then any "index or "pointer:formula associated with the "variable on the left is evaluated, and the value of the "formula is assigned to the "variable. Any reordering of the computations for optimization is prevented if an "order;directive precedes the "assignment;statement or the "declaration of a function for which a "function:call occurs in the "formula. In the forms: 1a5a

_ BIT _("formula_, "numeric;formula _, "numeric:formula _) 1a5a1

_ BYTE 1a5a2

the leftmost "formula, including any "indices or pointers, is evaluated first, then the second and then the third, if it exists, to determine the parts of the first "formula to use. 1a5b

,1 Assignment of a "formula of any type is permitted to a "variable of any type. Conversions and scaling are performed as needed when the operands are of different numeric types. If the types seem incompatible, the "formula on the right becomes a "bit:formula; then the bits of this "bit:formula replace the bits of the "variable on the left. If the "bit:formula has more bits than the "variable to which it is being assigned, leading bits are truncated; if there are too few bits, leading zeros are supplied before assignment. In assigning to the "functional:variable beginning with _BIT, the bits of the "formula on the right, whatever its type, are assigned to the specified bits of the "variable, right justified. 1a5b1

,2 After a value has been obtained from the "formula on the right, any "index and "pointer:formula for the "variable to be assigned are evaluated. If the form on the left is: 1a5b2

_BIT _("named:variable_, "numeric:formula _,"numeric:formula _) 1a5b2a

_BYTE 1a5b2b

any "index and "pointer:formula the "named:variable bears are evaluated first, then the second and third (if any) formulas, before assignment takes place. 1a5b3

,3 When assigning a "character:formula to any "variable ,B=1;not,B=0; a "character:variable, it first becomes a "bit:formula and is assigned as a bit string, right justified, and truncated on the left or padded on the left with zeros if necessary. In assigning a "character:formula to a "character:variable, if the "formula is too long, excess bytes on the ,B=1;right,B=0; are truncated before the assignment. If the "formula is too short, blank characters are added at the ,B=1;right,B=0; to match the size of the "variable before the assignment. 1a5b4

,4 The more general "assignment:statement permits multiple "variables to be assigned. These "variables may be individually listed or a sequence of occurrences of a "variable may be indicated by using an "indexed:variable:range (see Section 10.4.4), or some combination of these "variable forms may be used. The forms of the "assignment:statement that use "format:variable and "format:function:call are discussed in Chapter 6 on formatting. Omitting the forms related to formatting gives an "assignment:statement 1a5b5

```
     variable                       _     _=     formula
  _    _;                                                   1a5b5a

     indexed:variable:range
   indexed:variable:range                                   1a5b5b
```

.5 When the "indexed:variable:ranges are expanded to the sequences they represent, the above form is equivalent to: 1a5b6

```
   variable     _      _=   formula    _      _;            1a5b6a
```

which closely resembles the "simple:assignment:statement. In this "assignment:statement all of the "formulas are evaluated before any assignments are made. Then the value of the leftmost "formula is assigned to the leftmost "variable and the value of each "formula is assigned to its corresponding "variable--corresponding by position in the list. There must be at least as many "variables to the left of the "assignment:operator as there are "formulas to the right. If there are fewer "formulas than "variables, the value of the rightmost "formula is assigned to all the extra "variables at the right of the list of "variables. The leftmost "formula is evaluated first, then the next "formula, and so forth. After all "formulas are evaluated, any "index or "pointer:formula for the leftmost "variable is evaluated and the "variable is assigned. Then any "index or "pointer:formula for the next "variable is evaluated and that "variable is assigned a value, and so forth. 1a5b7

.6 The handling of _BIT or _BYTE, conversions, and type considerations are the same as for a "simple:assignment:statement. For a "formula to be assigned to several "variables, these considerations apply independently to each assignment. The following example illustrates an "assignment:statement 1a5b8

```
   AA[8:10], BB = AA[2:4];                                  1a5b8a
```

.7 This is equivalent to four "simple:assignment:statements: 1a5b9

AA[8] = AA[2]; AA[9] = AA[3]; 1a5b9a

AA[10] = AA[4]; BB = AA[4]; 1a5b9b

.8 The "assignment:statement below uses a special form for an "indexed:variable:range 1a5b10

ALL(TAB) = 0; 1a5b10a

.9 This "statement causes every entry of table _TAB to be cleared to zero. It should not be used unless the programmer knows that the entire table is core resident at this time even though it may be allocated in increments. If some occurrences of the "pointer:variable point to the same submanifold, it wastes time. If they point to other data, some unwanted clearing may occur. 1a5b11

.10 The "exchange:statement specifies that the old value of each of the two "variables is to become the new value of the other "variable. Any index or pointer for the "variable on the left is evaluated, then any "index or pointer for the "variable on the right is evaluated, and finally the values of the "variables are interchanged. The remarks made for "simple:assignment:statement concerning conversion, "variable type, and handling of _BIT or _BYTE apply also to the "exchange:statement. The following example of an "exchange:statement: 1a5b12

AA[I] == AA[I+J]; 1a5b12a

is equivalent to the three "simple:assignment:statements 1a5b13

TEMP = AA[I]; 1a5b13a

AA[I] = AA[I+J]; 1a5b13b

AA[I+J] = TEMP; 1a5b13c

5.6 "Zap:Statement 1a6

Execution of a "zap:statement causes all items of all entries of the designated table, or (in the second form) all items of the designated "entry:variable, to be set to null values. Character items are set to blanks; numeric items are set to zeros of the appropriate type. If items are overlaid (or share bits because of "specified:table:item:declarations) and such sharing results in conflicting values for some bits, the effect of _ZAP on such bits is undefined. If there are bits in an entry not affected by any "item:description, the effect of _ZAP on such bits is undefined. Like _ALL, _ZAP must never be used on a "table:name unless the entire table is core resident at the time even though it may be allocated in increments, 1a6a

5,7 "Conditional:Statement 1a7

The "conditional:statement provides for the conditional operation of a "statement or "statements based on the value of a "conditional:formula, 1a7a

,1 In either position in the definition of "conditional:statement, "controlled:statement is any one "statement, including a "conditional:statement, "switch:statement, "compound:statement, or "loop:statement, _BEGIN and _END brackets are not required unless it is desired that a group of two or more "statements constitute a "controlled:statement (or in the situation described in Section 5,7,8), The optional "statement:names preceding _ELSE have the same effect as if they followed _ELSE, 1a7a1

,2 The value of a "conditional:formula is the value of its rightmost bit after all operations called for in the "formula have been performed, The effect of a "conditional:statement depends upon whether there are one or two "controlled:statements, If there is but one, that "statement is executed if the value of the "conditional:formula is _1 and is otherwise skipped, If there are two "controlled:statements, the first one is skipped when the "conditional:formula has the value _0 and the second one is skipped when the value is _1, Whenever there are two "controlled:statements in a "conditional:statement, only one is executed, Following the execution of the appropriate "controlled:statement, program execution normally continues from the point immediately following the "conditional:statement, There are exceptions discussed in Section 5,7,7 and shown in the examples of Section 5,7,8, 1a7a2

,3 Following are two examples of "conditional:statements 1a7a3

a, IF ALPHA - BETA < 2 ; 1a7a3a

GOTO NEAR ; 1a7a3b

b, IF BOOL ; 1a7a3c

LBL: BEGIN 1a7a3d

```
          _RANDOM(;BASIC);                                        1a7a3e

          BASIC = BASIC**2;                                       1a7a3f

          _END                                                    1a7a3g

    _ ELSE                                                        1a7a3h

    L2:  BASIC = 0 ;                                              1a7a3i
```

.4 A list of alternatives may be encoded by nesting "conditional:statements. This can best be shown by example: 1a7a4

```
    IF ALPHA < BETA ;                                             1a7a4a

       ALPHA = BETA ;                                             1a7a4b

    _ELSE                                                         1a7a4c

    L1:    IF ALPHA + BETA > 10 ;                                 1a7a4d

    _              BEGIN                                          1a7a4e

                  GAMMA = (ALPHA + BETA)/2 ;                      1a7a4f

    L2:           ALPHA = GAMMA+1 ;                               1a7a4g

                  BETA = GAMMA-1 ;                                1a7a4h

    _              END                                            1a7a4i

         ELSE GOTO KEEP ;                                         1a7a4j

    _L3:                                                          1a7a4k
```

.5 The above example provides for the execution of one "assignment:statement if the first "conditional:formula is satisfied. It makes no difference then if the other "conditional:formula is satisfied. After execution of the single "assignment:statement the execution sequence continues with the statement at _L3. If the first "conditional:formula is not satisfied, the second "conditional:formula is examined, (and so forth). A jump to _L1 from elsewhere in the program causes the search for alternatives to begin at the point; it is as if execution of the "conditional:statement began at the top, where all the "conditional:formulas before the referenced "name were false. A jump to _L2 causes execution of the "statement at that point regardless of the satisfaction of the earlier "conditional:formulas. In this case only two of the three "simple;statements that constitute the "controlled:statement are executed. Following execution of BETA = GAMMA - 1 ; control passes to the statement at _L3. 1a7a5

.6 In constructing a nested "conditional:statement, it must be remembered that each _ELSE is matched with the nearest preceding unmatched _IF at the same level of BEGIN END nesting. For example, in 1a7a6

IF ...; ...; IF ...; ...; ELSE ...; 1a7a6a

the _ELSE matches with the second _IF, and in 1a7a7

IF ...; BEGIN IF ...; ...; END ELSE ...; 1a7a7a

the _ELSE matches the first _IF because the second _IF is not at the same level of _BEGIN END nesting as the _ELSE. 1a7a8

.7 In analyzing the flow of control through a complicated "conditional;statement, start with the innermost "conditional:statement matching _ELSE's with _IF's. Remember that an _ELSE is always immediately preceded and immediately followed by a "controlled;statement, one and only one of which will be executed depending upon the value of the "conditional;formula of the "statement to which the _ELSE belongs. Jumps must be generated around the "controlled:statement following a "conditional:formula if the value of the "formula is zero, and around the "controlled:statement following any _ELSE if the "controlled:statement preceding the _ELSE is executed even in part. This may require a series of jumps and it is hoped that a compiler will optimize a chain of unconditional jumps by using the final destination with the original jump. 1a7a9

.8 Some nestings of "conditional:statements are not permitted. An embedded "conditional;statement without _ELSE cannot be the "controlled:statement preceding _ELSE within an encompassing "conditional:statement since this would permit an _ELSE to match with an _IF for which it is not intended. In nested "conditional;statements, a "conditional:statement without _ELSE cannot be the "controlled:statement between two occurrences of the word _ELSE. Of course, the effect can be achieved by appending ELSE NULL; (or ELSE ;) or by enclosing the short statement in _BEGIN and _END brackets. The following two "conditional:statements are legal and have the same effect: 1a7a10

```
   IF A; IF B; IF C; E=1; ELSE ; ELSE IF D; E=2; ELSE ;
   ELSE E=3;                                                1a7a10a

                     C1        C2           D1        D2    1a7a10b

                         _B1                      _B2       1a7a10c

                                     _A1            _A2     1a7a10d

   IF A; IF B; BEGIN IF C; E=1; END ELSE BEGIN IF D; E=2;
   END ELSE E=3;                                            1a7a10e

                            _C1
   _D1                                                      1a7a10f

         _B1                                        _B2     1a7a10g

                                      _A1
   _A2                                                      1a7a10h
```

.9 In the examples above _A1 and _A2 show the first and second "controlled:statements of the "conditional:statement starting _IF A, _B1 and _B2 show the first and second "controlled:statements associated with _IF B, etc. The flow diagram below applies to either of the above "conditional:statements and shows the required skipping of "controlled:statements. 1a7a11

5.8 "Loop:Statement, "Test:Statement 1a8

The "loop:statement provides for the iteration of a "controlled:statement. 1a8a

.1 "For:clause is defined below. "Controlled:statement means the full generality of "statement under the control of iteration management mechanisms. Any "statement then can be iteratively operated in a "loop:statement. If the "controlled:statement starts with _FOR or BEGIN FOR, this is a nested "loop:statement. Multiple (parallel) control is provided with only one "for:clause. 1a8a1

.2 The iterations or repetitions of the "controlled:statement are managed by means of one or more "control:variables which are established and maintained within the "loop:statement. The "loop:statement consists, then, of a means of specifying and controlling "control:variables and a "controlled:statement that is to be iteratively operated, 1a8a2

.3 A "letter:control:variable, represented by a single "letter, is introduced within the "loop:statement for the purposes of iteration control, and is defined or active only within the immediate "loop:statement, See Section 5,10 for a more complete discussion of the scope of a "letter:control:variable, 1a8a3

.4 The definitions above show that a "for:clause may have a list of "loop:controls (for parallel control) where each "loop:control consists of a "control:variable followed by a list of "control:clauses enclosed in "parentheses, The "control:clauses associated with each "control:variable provide the successive values to be assigned to that associated "control:variable for successive executions of the "controlled:statement, Each "control:variable is given a successor value for each execution of the "controlled:statement, Execution of the "loop:statement is terminated when the "controlled:statement causes a non-return jump out of the "loop:statement or when, for any one of the "loop:controls, there is no successor available, The leftmost "control:clause in each parenthesized list is used first to assign values to the "control:variable, When it has been fully utilized, the next "control:clause in the list provides values, Utilization of the "control:clauses proceeds in this manner until the list is exhausted, 1a8a4

.5 Each "control:clause as defined above consists of up to three parts or phrases to specify the range for iterative operation of the "controlled:statement, 1a8a5

a, The "initial:phrase, if present, must come first in the "control:clause and serves to provide an initial value for the "control:variable, 1a8a5a

b, There may be either a "replacement:phrase (introduced by _THEN to specify the next value for the "control:variable or an "increment:phrase (introduced by _BY to specify the amount by which the "control:variable is to be modified on each iteration, 1a8a5b

c, A "terminator:phrase (introduced by _UNTIL or _WHILE may contain the test by which the end of the iteration process is determined, 1a8a5c

.6 During repeated cycling within a "control:clause, "formulas are normally reevaluated during each cycle. However, if the syntax permits and the "formula is enclosed in "brackets (indicating a "value:formula), the "value:formula is evaluated once upon entering the "control:clause and that value is used repeatedly without change until the "control:clause is exited. "Value:formulas in the "replacement:phrase, "increment:phrase and "terminator:phrase are evaluated after the "control:variable has been once given the value from the "initial:phrase, if there is one. The value (or even existence) of a "letter:control:variable and of "value:formulas are undefined if the "controlled:statement is entered via a jump to an internal "statement:name. 1a8a6

.7 In the general scheme, iteration begins at some initial value and continues in increments of a certain amount until (or while) a specified condition is detected. However, all of the parts of a "control:clause are optional except that a "letter:loop:control must have an "initial:phrase in its first "control:clause unless this same "letter is used as the "control:variable of a "loop:statement containing this one (nested). The effects of omitting various parts of the "control:clause are detailed in the table below. 1a8a7

.8 In utilizing the "control:clauses in accordance with the above table, every "formula is normally evaluated each time it is used. However, a "value:formula is evaluated only the first time it is encountered in executing the "loop:statement and the value is saved for subsequent use. When a value is added as stated in the table, its algebraic sign is, of course, taken into consideration. The presence of a "terminator:phrase causes testing after the "control:variable gets its new value (if it does get a new one) and before the "controlled:statement is executed--indeed, before the next "loop:control is attended to. The termination mentioned in the table applies really to utilization of the "control:clauses. If the "primitive is _WHILE and the "conditional:formula is _0 (false) or if the "primitive is _UNTIL and the "conditional:formula is _1 (true), instead of going on to the next "loop:control or executing the "controlled:statement immediately, the next "control:clause is utilized--or the "loop:statement is terminated if there are no more "control:clauses to be utilized with this "loop:control. In cases 1A, 1B, 3A, 3B, 4A, and 4B above, the previous value is whatever is left by previous operations including, if it happens, the value left by the previous "control:clause but not used in an execution of the "controlled:statement because of the condition of the terminator, 1a8a8

,9 It bears repeating--for every execution of the "controlled:statement a value is specified for each "control:variable by its respective associated list of "control:clauses, Whenever any one of the several lists of "control:clauses has been fully utilized and there is no value available for its respective "control:variable, execution of the "loop:statement is terminated, 1a8a9

,10 Consider an "indexed:variable as a "control:variable, In this case, a particular instance of the referenced "indexed:variable must be considered as well as the iteration of the "controlled:statement, It is possible for one iteration of a "controlled:statement to be performed based on a certain instance of a "named:variable as the "control:variable while the next iteration employs a different instance of that "indexed:variable, Each time an "indexed:variable as a "control:variable is to receive a new value from its associated list of "control:clauses, its "index is evaluated--to provide the particular instance of the "control:variable to be used, If incrementation is indicated, it is the presently indicated instance of the "control:variable that is incremented, The incrementation is performed as if by an "assignment:statement in which the old value is incremented and returned to the "indexed:variable, The rules affecting the order of evaluation of elements are in effect and the programmer is responsible for avoiding undesirable side effects, If the new value is merely the old value, without replacement or incrementation (first use in cases 1,3,and 4), all values are left undisturbed, In this last situation, there may not even be a need to evaluate the "index at this point, 1a8a10

,11 The "test:statement is permitted only within a "loop:statement, If a "control:variable is referenced, it must be a "control:variable for a "loop:statement (possibly nested) within which the "test:statement appears, Since all incrementing, replacing, and testing occurs before execution of the "controlled:statement, the "test:statement will invoke the "loop:controls for the designated "control:variable and all those which ,B=1;follow,B=0; it rather than those which precede it in the "for:clause, 1a8a11

.12 Execution of the "test:statement transfers execution to the "loop:controls in the "for:clause at the top of the "loop:statement--whichever "control:clauses are active at the moment. If no "control:variable is referenced, transfer is to the first "loop:control; and (in case the loops are nested) "loop:Controls for the innermost loop of the nest in which the "test:statement appears are invoked in the order in which they occur in the "for:clause. 1a8a12

.13 If a "control:variable is referenced in a "test:statement, transfer is to the "loop:control associated with the referenced "control:variable--skipping earlier parallel "loop:controls in the same "for:clause, and (in case the same "variable is a "control:variable in more than one "for:clause for nested "loop:statements) to the innermost active "loop:control associated with referenced "control:variable. 1a8a13

5.9 "Loop:Statement Execution 1a9

The "control:clauses associated with each "control:variable may be considered to form a list with a pointer indicating which clause to utilize. The pointer may even have two heads, the first pointing to the initial value (or the place reserved for the initial value) and then to the replacement or the increment and the second pointing to the terminator. The descriptions of what happens with regard to the various kinds of "control:clauses indicate what happens to the pointers. They remain with the clause just utilized or they go on to the next--the first pointer head moves from the initial value to the replacement or increment, or it remains with the replacement value or the increment. 1a9a

.1 The position of each pointer is undefined until the "loop:statement begins execution the first time. At that moment all pointers for the "loop:controls in that "loop:statement are set to point to their respective first "control:clauses. This happens every time upon normal entry to the "loop:statement through the top. 1a9a1

.2 From that time on, as long as the scope containing the "loop:statement remains active, the positions of the pointers are defined. If control leaves the loop through execution of a "procedure:call:statement or a "go:to:statement, the positions of the "control:clause pointers are undisturbed. If execution of the loop is resumed (such as by means of a "go:to:statement referencing a "statement;name within the "controlled:statement), utilization of the "control:clauses proceeds as if the "loop:statement had not been interrupted. 1a9a2

.3 Each "loop:control may be considered to have an additional clause that says, in effect, "terminate now". If the pointer is moved to this implicit clause and the clause is then examined, execution of the "loop:statement terminates immediately; no action is taken with regard to subsequent "loop:controls in case there are more than one. The pointer will continue to point to the "terminate now" clause and will not be moved again until it is initialized through a subsequent normal execution of the "loop:statement. If now, the loop is entered "through a side door" it will surely terminate when the iteration mechanism is invoked, regardless of what values the "control:variables may have (having been set perhaps outside the loop) because one pointer is pointing to "terminate now". Before termination, however, "control:variables which occur earlier in the "loop:statement than the one which caused the previous termination will be incremented. In fact, one of these earlier "control:variables may now cause the termination and there will then be two pointers pointing to their respective "terminate now" clauses. Of course, if the iteration mechanism is invoked by means of a "test:statement that bypasses all "loop:controls pointing to "terminate now", the "controlled:statement may be executed again. 1a9a3

.4 An example of a parallel "loop:statement is: 1a9a4

```
   A1 = 0;                                          1a9a4a

   FOR A1(1 THEN 1-A1 UNTIL 0)                      1a9a4b

   B2(1 BY 1 UNTIL C3 = 8)                          1a9a4c

   _C3(1,1,2,3,5);                                  1a9a4d

   S4; D5[B2] = A1 * C3;                            1a9a4e

   C3 = C3 + 3;                                     1a9a4f

   IF C3 < 12; S4; STOP;                            1a9a4g
```

.5 In order to illustrate the flow of execution for the example above, the diagram below illustrates the changing of values for the "variables in the example, If there is a number at the intersection of an arrow and a column, the "variable at the left of the row receives that value, The setting of values proceeds down the leftmost column, then down the column to its right, etc, 1a9a5

	A	B	C	D	E	F	G	H	J	K	L	M	1a9a5a
A1	0	1	0	1	0	1	0		1		0		1a9a5b
B2		1	2	3	4	5	6		7				1a9a5c
C3		1	1	2	3	5		8		11		14	1a9a5d
D5[1]		1											1a9a5e
D5[2]			0										1a9a5f
D5[3]				2									1a9a5g
D5[4]					0								1a9a5h
D5[5]						5							1a9a5i
D5[6]								0					1a9a5j
D5[7]										11			1a9a5k

,6 Column _A above represents the setting of _A1 by the first "statement in the example, Column _B represents the initialization of the "control:variables and the first execution of the "controlled:statement, Columns _C, _D, _E, and _F represent iterations of the loop, In column _G, we have another in the series of replacements for _A1, another incrementation of _B2, and termination because the next clause for _C3 is "terminate now", Column _H shows the setting of _C3 outside the loop and the setting of _D5[6] because of the jump into the loop, Column _J shows another replacement for _A1, incrementation of _B2, and termination--this time because the test on _C3 (in the _B2 control) turns out "true", Column _K shows a new setting for _C3 outside the loop and the setting of _D5[7] because of the jump back in at _S4, Column _L shows the replacement for _A1 and termination--now because the _B2 control points to "terminate now", Column _M shows the setting of _C3 outside the loop, The program now stops because the test on _C3 fails, 1a9a6

5.10 Scope of "Letter:Control:Variables 1a10

The scope of a "letter:control:variable is just the "loop:statement in which it is defined and activated. A "letter:loop:control defines the "letter:control:variable and it is assigned the value of the "initial:phrase of its first "control:clause. The "letter:control:variable is then active and may be used as an "integer:variable until the end of the "controlled:statement of the "loop:statement. The "letter:control:variable may also be used in the "increment:phrase, "replacement:phrase, or "terminator:phrase of the first "control:clause of the "letter:loop:control that activates it and in any "formula of other "control:clauses of the "letter:loop:control. 1a10a

.1 In a nested "loop:statement, if an inner "letter:loop:control uses as a "control:variable a "letter that is already active as a "letter:control:variable, it is treated as a "named:variable in the inner "loop:statement. Its first "control:clause need not have an "initial:phrase. The "letter:control:variable is the same "variable as the one defined in the outer "loop:statement, and its final value in the inner "loop:statement is carried into the outer "loop:statement. 1a10a1

.2 The same "letter may be used as the "letter:control:variable for non-nested "loop:statements, but these "control:variables are then considered as different "variables. 1a10a2

.3 A "letter:control:variable remains active only so long as execution remains within the "loop:statement. In general, "letter:control:variables are deactivated whenever control is transferred outside the "loop:statement by means of a "go:to:statement or by coming out the bottom because of completion of the "loop:statement. "Letter:control:variables are not deactivated when a procedure or function is called and that procedure or function returns control to this "loop:statement. 1a10a3

.4 "Procedure:declarations and "switch:statements may occur inside a "loop:statement with "letter:loop:control and they may reference the "letter:control:variable. However, the value or even the existence of a "letter:control:variable is undefined if the "controlled:statement is entered via a jump from an outer scope to a "statement:name in the "loop:statement, or if the internal procedure or function is called from outside the "loop:statement. 1a10a4

5.11 "Procedure:Call:Statement 1a11

A "procedure:call:statement is used to invoke a procedure that is not a function. 1a11a

.1 For a discussion of "remquo:procedure:call:statement, see Section 5.11.17. 1a11a1

.2 "Actual:input:parameters must match the "formal:input:parameters associated with the named procedure (or alternate entrance) in number, kind, and position in the list, and "actual:output:parameters must match the "formal:output:parameters in number and position in the list. Matching to kind proceeds thus: if the "formal:input:parameter is a "statement:name, the corresponding "actual:input:parameter must be a "statement:name or one of the forms beginning with _STOP, _RETURN, _TEST, or _EXIT. If the "formal:input:parameter is an "item:name, the corresponding "actual:input:parameter must be a "formula. If the "formal:input:parameter is a "table:name or a "data:block:name, the corresponding "actual:input:parameter must be a "data:block:name, a "table:name (with or without an "index), a "variable, or _@ followed by a "pointer:formula. If the "formal:input:parameter is a "procedure:name, the "actual:input:parameter must be the "name of a procedure with the same number, kind, and position of "formal:input:parameters and "formal:output:parameters as the procedure used as a "formal:input:parameter. 1a11a2

.3 When a procedure makes a call on a procedure which is one of its "formal:input:parameters, the conversions between "actual:input:parameters and "formal:input:parameters are made in accordance with the descriptions of the "formal:input:parameters of the procedure as a "formal:input:parameter. No cognizance is taken of the descriptions of the "formal:input:parameters of the procedure which is given as an "actual:input:parameter. Consider this example: 1a11a3

PROC AA (BB,CC); 1a11a3a

BEGIN PROC BB (DD); 1a11a3b

procedure _AA BEGIN ITEM DD F; END procedure
_BB (formal "parameter) 1a11a3c

ITEM CC U 26; 1a11a3d

BB (CC); 1a11a3e

_END 1a11a3f

ITEM COUNT S 15; 1a11a3g

PROC EE (FF); 1a11a3h

BEGIN ITEM FF U 30; 1a11a3i

COUNT = COUNT + FF; procedure _EE (actual
"parameter) 1a11a3j

_END 1a11a3k

AA (EE,5); the executed "statement 1a11a3l

,4 In the example above, the value _5 is assigned to _CC, using whatever conversion is called for by the assignment rules, during invocation of _AA, Then the formal invocation of _BB causes _CC to be converted as if for assignment to _DD (there need be no actual space allocated for _BB or _DD, There is ,B=1;no,B=0; further conversion from _DD to _FF--it is the programmer's responsibility to see that _FF matches _DD (or to accept the consequences if it does not), 1a11a4

,5 The order of evaluation of parameter data is left to right, The values of "actual:input:parameters are assigned to "formal:input:parameter:variables from left to right as if by an "assignment:statement (Section 5,5), Upon exit, "formal:output:parameters are assigned to "actual:output:parameters from left to right as if by an "assignment:statement, This order of assignment is certainly of significance if some "formal:input:parameters are pointers to other "formal:input:parameters or if some "actual:output:parameters are pointers to other "actual:output:parameters, 1a11a5

,6 If a procedure is to refer to an external table or data block, the location of the table or data block may be passed to the "formal:input:parameter which will be used by the procedure as the pointer in its references to its local table or data block. The location of the external structure can be indicated by using a "location:function:call or a "pointer:formula evaluating to the correct location. The procedure will then utilize the pointed-to space in accordance with the declared structure of its local pointed-to table or data block. Alternatively, a local "table:name or "data:block:name can be a "formal:input:parameter. In that case, the "actual:input:parameter represents a location that will become the value of the pointer (either programmer designated or compiler supplied) to the local structure. It doesn't matter if the "formal:input:parameter is a table or a data block. If the corresponding "actual:input:parameter is a "variable, a "table:name, or a "data:block:name, the location of the variable, table , or datablock (as if called forth by the use of _LOC) becomes the value of the internal pointer. This location is the address of the word in which the variable or other data structure begins. No bit or byte locating information within the word is involved. If it is desired to provide a "formula (perhaps just a "constant or "variable) as a location value to be provided to the internal pointer to table or data block used as a "formal:input:parameter, the "formula must be preceded with an _@. 1a11a6

.7 Note that "table:name or "variable has a different meaning as an "actual:input:parameter depending on the corresponding "formal:input:parameter. If the "formal:input:parameter is a "table:name or "data:block:name, a "table:name or "variable as an "actual:input:parameter means the location (of the variable or table). If the "formal:input:parameter is a "variable, a "table:name as an "actual:input:parameter means the value of the first "entry:variable, and any other "variable appearing as an "actual:input:parameter is simply interpreted as its value. 1a11a7

.8 Consider, for example, a procedure that processes messages in the form of long "character:variables. Within the "procedure:declaration a table (_T1) is declared with one entry consisting of a very long "character:variable (_C1). _T1 is also a "formal:input:parameter. Therefore, no space is allocated for the table or its item. Now the "procedure:call:statement uses the "name of a "character:variable (_MSG) as the corresponding "actual:input:parameter. The location of _MSG is passed in to the procedure and becomes the value of the pointer (named or unamed) to _T1, If the procedure looks at one message and creates another, perhaps two such "formal:input:parameters and "actual:input:parameters are needed, 1a11a8

,9 If the procedure exits by means of a "go:to:statement referencing a "formal:input:parameter, the first effect is to set the "actual:output:parameters from the values of the "formal:output:parameters, The next effect depends on the nature of the "actual:input:parameter corresponding to that "formal:input:parameter, If the "actual:input:parameter is a "statement:name, it is as if a "go:to:statement referencing that "statement:name were executed at a point immediately following the "procedure:call:statement, If the "actual:input:parameter is one of the forms beginning with _STOP, _RETURN, _TEST, or _EXIT, it is as if a corresponding "stop:statement, "return:statement, "test:statement, or "exit:statement were executed at a point immediately following the "procedure:call:statement, 1a11a9

,10 A "go:to:statement referencing an outer "statement:name causes a jump to the point named and it also deactivates all procedures called from the scope of that outer "statement:name, and procedures called by those procedures, etc, It bypasses the setting of "actual:output:parameters of the procedure in which the "go:to:statement is executed and all other procedures deactivated by the jump, If, however, the outer "statement:name is a "formal:input:parameter in its own scope, the "actual:output:parameters of that procedure are set before control is transferred in accordance with the "actual:input:parameter corresponding to that "statement:name used as a "formal:input:parameter, 1a11a10

,11 The deactivation through execution of a "go:to:statement can occur recursively, Suppose that _ALPHA is a recursive procedure in which the procedure _BETA is nested and the procedure _GAMMA is nested in _BETA, Suppose also that _ALPHA had been invoked, which then called _BETA, which called _GAMMA which called _ALPHA (creating a second copy of _ALPHA, including _BETA and _GAMMA, Now, once again _ALPHA calls _BETA and _BETA calls _GAMMA, If the second copy of _GAMMA jumps directly to a "named:statement in _ALPHA, it is a jump to that "statement in the second copy (or perhaps second use in case _ALPHA is reentrant), It deactivates the second activations of _BETA and _GAMMA but not the first, 1a11a11

.12 There are several attributes of procedures that can lead to complications in implementation and execution. The least traumatic attribute is a pointed-to data space. This can be implemented by the calling program placing the value of the pointer in a base register which is then used by the procedure in all its references to its private data. 1a11a12

.13 Complications due to the use of external procedures depend on the amount of work done by the loader. If the loader resolves the locations of all "formal:input:parameters and "formal:output:parameters as well as of the procedure, no further complication arises. If the loader does not resolve these locations, then it is necessary for the calling program to convert "actual:input:parameters, if necessary, and store them or their locations in a standard communications area from whence they are retrieved by the called program. The process is reversed for "actual:output:parameters. 1a11a13

.14 For recursive procedures, it becomes necessary to save some extra pointer values in the data space assigned for each "procedure:call:statement in order to be able to unwind. The "recursive:directive (Section 11.7.5) may be important in this regard. If procedure _P1 calls procedure _P1, the "recursive:directive is unnecessary--it is obvious to the compiler that _P1 is a recursive procedure. On the other hand, if _P2 calls _P3 and _P3 calls _P2, recursiveness depends on information not available to the compiler. It may happen that execution is such that there is never more than one active call each of _P2 and _P3. But if _P3 does actually call _P2 after it has been called by _P2, then the programmer uses the "recursive:directive to indicate that _P2 is recursive. 1a11a14

.15 Whether and where the "recursive:directive is needed is system dependent. Recursive procedures will probably be able to unwind properly if all procedures with pointed-to data space implement calls on other such procedures in a manner similar to the following: 1a11a15

a. Call on procedure with pointed-to data space from within procedure with pointed-to data space (_r1 and _r2 are registers set aside for such calls):
Conditions: _r1 points to current data space,
_r2 points to data space of
procedure that called this one. 1a11a15a

b. This procedure calls a procedure: 1a1ia15b

_r2 _SAVE @ r1 1a1ia15b1

_r1 _r2 1a1ia15b2

"pointer:formual (elements @ r2 evaluated _r1 1a1ia15b3

"actual:input:parameters @ r2
"formal:input:parameters @ r1 1a1ia15b4

return jump (called procedure saves return @ r1 and jumps back) 1a1ia15b5

"formal:output:parameters @ r1
""actual:output:parameters @ r2 1a1ia15b6

_r2 _r1 1a1ia15b7

SAVE @ r1 _r2 1a1ia15b8

.16 The following example is a simple recursive procedure that calculates the factorial of a number. It is certainly not the recommended way to calculate a factorial, but it illustrates recursion: 1a1ia16

PROC FCTRL @ (G1 : G2) ; 1a1ia16a

BEGIN ITEM G1, G2, Q1 U ; 1a1ia16b

G2 = 1 ; 1a1ia16c

IF G1 <= 1 ; RETURN ; 1a1ia16d

Q1 = SPACE (DSIZE (FCTRL)) ; 1a1ia16e

FCTRL @ Q1 (G1 - 1 : G2) ; 1a1ia16f

GARBAGE (Q1) ; 1a1ia16g

G2 = G2 * G1 ; 1a1ia16h

_END 1a1ia16i

.17 The "remquo:procedure:call:statement is used to obtain the quotient and remainder that result from dividing the first "actual:input:parameter by the second "actual:input:parameter. 1a1ia17

.18 The compiler may implement the "remquo:procedure:call:statement by efficient inline code (preferably) rather than a call to a procedure. The effect of this code is as if the following procedure were called: 1a11a18

```
   PROC REMQUO (NUM, DEN : QUO, REM) ;                1a11a18a

   BEGIN ITEM NUM, DEN, REM, QUO S n;                 1a11a18b

      QUO = NUM/DEN ;                                 1a11a18b1

      REM = NUM - QUO * DEN ;                         1a11a18b2

   _END                                               1a11a18c
```

.19 The "item:declaration above contains the letter _n, which stands for a system-dependent size for these integer items. It will undoubtedly be the maximum size for convenient arithmetic with integers rather than the size used for pointers and addresses. The letter size is likely to be the system default size. 1a11a19

5.12 "Go:To:Statement, "Stop:Statement, "Return:Statement, "Exit:Statement 1a12

Normally, the "statements of a "processing:declaration are executed in the order in which they are written. The four "simple:statements below are used in modifying this normal execution order of the "statements. The "test:statement (Section 5.8.12) discussed in connection with the "loop:statement of which it may be a part, the "procedure:call:statement (Section 5.11), and the "switch:statement (Section 5.13) are also used to control the execution order of "statements. 1a12a

.1 The "go:to:statement effects a transfer of control to the "statement bearing the referenced "statement:name. A single-component "index is permitted only in transferring control to a named "switch:statement. In such a reference, the value of "index is used instead of the value of the "numeric:formula beginning the "switch:statement in selecting the constituent "statement to be executed. If a "go:to:statement referencing a "switch:statement has empty "index "brackets, it is as if there were an "index with the lowest meaningful value with respect to the "switch:statement. A "go:to:statement with an "index (or empty "index "brackets) may reference a "statement:name that precedes the _BEGIN of a "switch;statement. In fact, a "go:to:statement, without "index or "brackets, referencing such a "statement:name has the effect of a "go:to:statement with empty "brackets. The value of the "index (or its lowest meaningful value in case it is omitted) is used to select the constituent "statement to be executed. 1a12a1

.2 The "stop:statement is the logical termination of execution of a program. Depending on the system, _STOP may cause a machine halt or a normal return to the executive. 1a12a2

.3 The "return:statement is permitted only within a "procedure;body. Its effect is to terminate execution of the procedure, set the "actual:output:parameters from the "formal:output:parameters, and return control to the "statement following the call in whatever program invoked the procedure. The call might have been in any scope such as another procedure, the main program, or even the system executive. 1a12a3

.4 It is immaterial whether the "return:statement references a "procedure:name or an "alternate:entrance:name; the return will be that associated with the active entrance in any case. If no "name is referenced, the "statement means to return from the most local procedure. After the last "statement in a "procedure;body is executed, if it does not transfer control, return from the procedure is effected as if the "statement _RETURN; had been executed. 1a12a4

.5 Within nested procedures, the referenced "name in a "return:statement means to return from the procedure having the referenced "name as its normal or alternate entrance. (If nested procedures use the same "name, it means return from the most local procedure, having the referenced "name, within which the "statement appears. If return is made to an outer procedure from within an inner procedure, the "actual:output:parameters are not set for the inner procedure. 1a12a5

.6 Return to a procedure that is not active is undefined. "Active" means the procedure has been called but an explicit or implicit return from the procedure has not yet been executed. Such a return could only be attempted from a procedure declared with an external definition within another procedure. 1a12a6

.7 There exists a school of programming philosophy that holds that the use of "go:to:statements leads to poor coding practices and renders certain optimization techniques impossible. Programming done in accordance with the precepts of this school is known as structured programming. Inclusion of the "exit:statement (and the "switch:statement) makes structured programming possible in JOVIAL. It is not suggested that any compiler ought to, but it would certainly be possible to build a compiler to enforce structured programming or to issue a warning when the precepts of structured programming are violated. 1a12a7

.8 Heretofore, a "statement:name was considered only to designate a point, an entrance point, in a program. To give meaning to the "exit:statement, a "statement:name must be understood to have reference to an entire program structure--the "statement to which it is applied, its entrance point, and its exit point. An "exit:statement may only appear between the entrance point and the exit point of the program structure whose "name it references. The effect of executing the "exit:statement is to transfer control to the exit point of the program structure. The interstices between program structures sometimes have a fine structure so that, for example, the exit point of a "loop:statement is not the same as the exit point of its "controlled:statement. 1a12a8

.9 The effect of an "exit:statement referencing a "name attached to a "procedure:body is the same as a "return:statement for that procedure. The effect of an "exit:statement referencing a "name attached to the "controlled:statement of a "loop:statement is the same as a "test:statement, without reference to a "control:variable, in the "named:statement but not in any loop nested within that "named:statement (even if the "exit:statement is in the nested loop). The effect of an "exit:statement referencing a "name attached to a "loop:statement terminates execution of the loop. The effect of an "exit:statement referencing a "statement:name that precedes the _BEGIN of a "switch:statement is the same as if that "statement:name were attached to the "switch:statement. 1a12a9

.10 The following example illustrates the details of the effect of executing various "exit:statements. The relevant program details and the "exit:statements are on the left. The notation . . . indicates a sequence of "statements, To the right of each "exit;statement is an equivalent "statement and an explanation, 1a12a10

```
    L0:  BEGIN                                                  1a12a10a

         . . .                                                  1a12a10b

         EXIT L0                 GOTO L1;             (Exit
    from "compound:statement)                                   1a12a10c

         . . .                                                  1a12a10d

         END                                                    1a12a10e

    L1:  FOR I (1 BY 1 UNTIL I = 9);                            1a12a10f

    L2:  BEGIN                                                  1a12a10g

         . . .                                                  1a12a10h

        EXIT L1;                 GOTO L3;             (Exit
    from the "loop:statement)                                   1a12a10i

         . . .                                                  1a12a10j

        EXIT L2:                 TEST;                (Exit
    from the "controlled:statement)                             1a12a10k

         . . .                                                  1a12a10l

         END                                                    1a12a10m

    L3:  IF  ALPHA > 0;                                         1a12a10n

    L4:  BEGIN                                                  1a12a10o

         . . .                                                  1a12a10p

        EXIT L3;                 GOTO L6;             (Exit
    from the "conditional:statement)                            1a12a10q

        EXIT L4;                 GOTO L6;             (Exit
    from "controlled:statement                                  1a12a10r

         . . .                                       is
    effectively the same)                                       1a12a10s

         END                                                    1a12a10t
```

```
L5:  ELSE                                                  1a12a10u

     BEGIN                                                 1a12a10v

    . . .                                                  1a12a10w

    EXIT L5;                GOTO L6;             (Exit
from "controlled:statement)                                1a12a10x

    . . .                                                  1a12a10y

     END                                                   1a12a10z

L6:  SWITCH ALPHA + 1;                                     1a12a10a@

L7:  BEGIN                                                 1a12a10aa

L8:  BEGIN                                                 1a12a10ab

    . . .                                                  1a12a10ac

    EXIT L6;                GOTO L11;            (Exit
from "switch:statement)                                    1a12a10ad

    EXIT L7;                GOTO L11;            (Exit
from "switch:statement)                                    1a12a10ae

    . . .                                                  1a12a10af

    EXIT L8;                GOTO L11;            (Not _L9,
because of the                                             1a12a10ag

                                                  "comma
    . . .
before _L9                                                 1a12a10ah

     END                                                   1a12a10ai

L9:  BEGIN                                                 1a12a10aj

    . . .                                                  1a12a10ak

    EXIT L9;                GOTO L10;            (Exit
from "compound:statement--                                 1a12a10al

                                                  no
    . . .
"comma follows)                                            1a12a10am

     END                                                   1a12a10an
```

```
        L10: . . .                                               1a12a10ao
             END                                                 1a12a10ap
        L11: . . .                                               1a12a10aq
            PROC P1 ;                                            1a12a10ar
        L12:  BEGIN                                              1a12a10as
            . . .                                                1a12a10at
            PROC P2 ;                                            1a12a10au
        L13:  BEGIN                                              1a12a10av
            . . .                                                1a12a10aw
            EXIT L12;               RETURN P1;          (Exit
        from "procedure:body)                                    1a12a10ax
            . . .                                                1a12a10ay
             END                                                 1a12a10az
            . . .                                                1a12a10b@
             END                                                 1a12a10ba
```

5.13 "Switch:Statement 1a13

A "switch:statement provides a multipath branch to other "statements contained within it. 1a13a

.1 Each "statement between _BEGIN and _END in the above form is associated with an integer. In the absence of explicit bracketed "numbers ahead of or between "statements, the first "statement is associated with zero, and successive "statements (including "null:statements) are associated with successive integers. (A "compound:statement, "switch:statement, "loop:statement, or "conditional:statement counts as a single "statement.) Each bracketed "number, where present, interrupts the succession of associated integers and states a positive or negative integer value to be associated with the next "statement. The succession then resumes following the stated value. There must be no repetition in values--each "statement must be associated with a unique integer value. The "statements and their associated integer values are then effectively reordered so that the integer values are in monotonically increasing order, with no duplications but possible gaps, Some of the ^statements may be followed by ^commas, 1a13a1

,2 In executing the ^switch:statement, the ^numeric:formula following _SWITCH is evaluated as an integer (truncated if necessary), Then the ^statement enclosed in the BEGIN END brackets and corresponding (as described above) with the values of the ^formula is executed, If the ^numeric:formula does not yield a value corresponding to a ^statement in the list (including ^null:statements), the result is undefined, Values skipped due to explicit ^numbers in the list do not correspond to ^statements, A ^statement in the list can be executed, if it bears a ^statement:name, by execution of a ^go:to:statement somewhere that references that ^name, 1a13a2

,3 After execution of any ^statement in the list, if it does not permanently transfer execution elsewhere, the next ^statement, effectively, in the list is executed, This occurs unless the statements are separated by a ^comma or a gap in the sequence of integer values, in which case the execution sequence is transferred to the ^statement following the _END, 1a13a3

,4 Execution control can arrive at a ^switch:statement in three ways: by "falling through" from the ^statement preceding the ^switch:statement, through execution of a ^go:to:statement referencing the ^name of the ^switch:statement and without an ^index or ^index ^brackets, or through execution of a ^go:to:statement that does have a one-component ^index and references the ^name of the ^switch:statement, The first two of these ways result in the execution of the ^switch:statement as described in Section 5,13,2, 1a13a4

,5 The use of the ^index in a ^go:to:statement referencing a ^name before the ^switch:statement (or a ^name before the _BEGIN of the ^switch:statement even if there are no ^index and ^brackets) means that the ^numeric:formula of the ^switch:statement is not evaluated, Instead the value of the ^index with the ^go:to:statement is used to select the ^statement in the list to be executed, Evaluation of the ^numeric:formula is omitted even if an ^order:directive precedes the ^switch:statement, If the ^go:to:statement has empty ^brackets for the ^index, it is as if there were an "index present having the smallest value associated with a "statement in the list, 1a13a5

,6 The following example and flow diagram illustrate the use of a "switch;statement: 1a13a6

```
   SW:  SWITCH ALPHA;                                      1a13a6a

        BEGIN [1] BETA = 3; ;                              1a13a6b

                  GAMMA = BETA;,                           1a13a6c

                  IF GAMMA = 2 ; BETA = 2;                 1a13a6d

              [6] BETA = GAMMA;                            1a13a6e

        END       ALPHA = 7;                               1a13a6f
```

,7 In the example above, values of _2 and _3 for _ALPHA cause the same path to be taken because the "statement corresponding to the value _2 is a "null:statement, The "statement 1a13a7

```
   GAMMA = BETA;                                           1a13a7a
```

exits to the end of the "switch:statement because it is followed by a "comma, The "conditional:statement 1a13a8

```
   IF GAMMA = 2;  BETA = 2;                                1a13a8a
```

exits to the end of the switch because it is "statement number _4 and there is no "statement number _5 1a13a9

,8 Using the same example in Section 5,13,6, the "go:to:statement 1a13a10

```
   GOTO  SW  [BETA];                                       1a13a10a
```

would cause the value of _BETA to be used rather than the value of _ALPHA in executing the "switch;statement, The "go:to:statement 1a13a11

```
   SW  [  ];                                               1a13a11a
```

would cause the path for _ALPHA equal to _1 to be taken, 1a13a12

,9 If the optional "statement:name after the "numeric:formula of a "switch:statement does indeed occur and is referenced in an "exit:statement, it is as if the "statement;name were attached to the "switch;statement and it causes an exit from the "switch:statement, 1a13a13

5,14 "Direct;Statement 1a14

The "direct:statement is a "simple;statement, It is used as a means for breaking out of the JOVIAL language within a program and writing some instructions in another language more directly related to the organization of the computer for which the program is being compiled, 1a14a

,1 What is legal and meaningful in a "direct:statement depends on the system, A "direct:statement may reference a JOVIAL "name in the same scope, The "name translates to a location, but the exact meaning of "location" is system dependent, A "sets:directive and a "uses:directive may appear immediately after the "primitive _DIRECT to inform the compiler of data elements referenced in the "direct;statement, 1a14a1

,2 If the optional "semicolon occurs after the "primitive _JOVIAL, it serves only as a terminator for the "direct:statement and is not a "null;statement, 1a14a2

,3 While machine-language code might in some cases be desirable for object program efficiency, there are obvious disadvantages to using "direct:statements, Errors that might be detected by the compiler if JOVIAL were used are more likely to go undetected, "Program:declarations containing "direct;statements are more difficult to transfer to another machine, 1a14a3

J73Chap5EDIT

(J30784) 22-MAY-74 11:16; Title: Author(s): Roberta J. Carrier/RJC; Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;
Origin: <CARRIER>JOVIALCHAP5EDIT.NLS;1, 22-MAY-74 11:06 RJC ;

Marcia:
Could you please send me one copy of each of the following four RFCs, we have lost all trace of these, and cant find copies of transmittal letters 128 or 130, though we do have 131, The missing RFCs are:

624	22054
623	22004
620	21993
613	21525

1

(J30785)   22-MAY-74 11:35;   Title: Author(s): Jonathan B, Postel/JBP;
Distribution: /MLK; Sub-Collections: NIC; Clerk: JBP;