Chapter 6 1

FORMATTING 1a

6.1 Introduction 1a1

JOVIAL formatting provides the capability of translating between a character buffer and data elements. The character buffer is a "character;formula or "character;variable and contains a string of any legal ASCII characters including the control characters. A data element is a "variable or "formula depending upon the direction of translation or it may be an "indexed;variable;range (see Section 10.4.4). 1a1a

.1 Formatting can be either list-directed or format-directed. When list-directed, the data elements determine the manner in which the character string is generated or scanned. When format-directed, an explicit "format;list controls scanning or generation of the character string. List-directed formatting provides a free-field "input" capability. 1a1a1

.2 Translation is accomplished by an "assignment;statement and formatting routines. The direction of translation is determined by the right and left side of the "assignment;statement. 1a1a2

.3 Translation from the character buffer into the data elements is accomplished by an "assignment;statement (Section 5.5) of the form: 1a1a3

"variable 1a1a3a

"indexed;variable;range _=
"format;function;call _; 1a1a3b

.4 _FORMAT signals an "intrinsic;function;call to a formatting routine designed to translate between character buffers and data elements. Its first "parameter, "character;formula, provides the character buffer from which values are obtained, translated as necessary, and assigned to the data elements on the left of the "assignment;statement. 1a1a4

.5 If the optional second "parameter to _FORMAT, "format;list, is present, it implies "format-directed formatting; but its place in the "parameter list is marked by a "comma for list-directed formatting if the third "parameter is given. "Format;list, defined in

Section 6.14, controls the scanning of the character buffer. 1a1a5

.6 The optional third _FORMAT "parameter, "procedure:name, names a contingency processing procedure to be called by the formatting routine. The details of the interface between the contingency procedure specified and the formatting routines are left to the implementer. The intent of the procedure is to process errors, end-of-line conditions, etc., without the necessity of returning from the formatting routine. For contingency action, the formatting routine will call the named procedure which might, for example, cause a line to be written or read and then return to the formatting routine to continue translation. 1a1a6

.7 Translation from data elements into a character buffer is accomplished by means of an "assignment:statement of the form: 1a1a7

"format:variable _= "formula _; 1a1a7a

"indexed:variable:range 1a1a7b

.8 Here _FORMAT signals a call to an intrinsic formatting routine designed to translate between data elements and character buffers. Its first "parameter, "character:variable, names the character buffer into which the translated values of the data elements given on the right of the "assignment:statement are put. 1a1a8

.9 The second _FORMAT "parameter is as described in Section 6.1.5 above except that the "format:list, if present, controls the generation of the character string in the buffer. The third _FORMAT "parameter is as described in Section 6.1.6 1a1a9

6.2 List-Directed Formatting 1a2

List-directed formatting provides for translation from a input string to data elements or from data elements to an output string according to the syntax rules described below. Fields in the character buffer are separated by spaces. ("Input" and "output" refer to the usual uses of these character strings, but this is not a requirement, since a core-to-core transfer may be all that is actually involved.) 1a2a

.1 When the translation is from a character buffer (input), the formatting routine scans the character

buffer, separating the fields and translating each field according to the type of the corresponding "variable to be assigned a value from the field. The formatting routine must recognize that there may be more data elements than there are fields, and possibly provide for new input using the contingency procedure. 1a2a1

.2 For a "character:variable, the field is delimited by spaces just as for other "variables if there are no spaces which are a part of the field. If the character field contains spaces, it must be .B=1;additionally,B=0; delimited by a prime immediately preceding the first and a prime immediately following the last of the characters which constitute the field. The primes merely delimit the size of the field and are not considered a part of the field to be passed on to the "character:variable. Two consecutive primes would denote a field of zero characters. Obviously, it is impossible to include a prime in an input field delimited by primes. A leading prime with no trailing prime is illegal. For a field where the first nonblank character is not a prime, the field starts with that character and includes all characters following it up to the next space or blank, and so may include primes. Primes may delimit fields that do not contain either spaces or primes. A list-directed input field cannot contain both spaces and primes. 1a2a2

.3 The characters from the input field are assigned to the "character:variable using the rules of assignment; i.e., they are left justified. If the input field is too long, excess bytes on the right are truncated before the assignment. If there are too few bytes in the input field, blanks are added at the right to match the size of the "variable before assignment. 1a2a3

.4 For "numeric:variables, the characters in the input field are syntactically analyzed to determine their form (integer, fixed, or floating type), converted appropriately as if they were JOVIAL "constants to the type of the "numeric:variable, and then transferred according to the rules of assignment. The forms that may occur in the input string do not permit all forms for JOVIAL "numeric:constants. In particular, they must fit the following syntactic form and, as said before, no spaces are permitted: 1a2a4

_+ digit _. digit " _E _+ digit " 1a2a4a

_. digit " 1a2a4b

.5 List-directed input to a "bit:variable is undefined, 1a2a5

.6 For list-directed output to a character buffer, the "formulas that comprise the data elements are evaluated and then translated to the appropriate character representation and placed in the character buffer, Any pair of fields is separated by a space, If the value of a "formula is zero, one zero is output, The formatting routine must recognize when the character buffer will not hold another field, and possibly provide for output and clearing of the buffer using the contingency procedure, 1a2a6

.7 Converson for list-directed output takes place according to the "formula types as follows: 1a2a7

a, "Character:formula, Spaces to the right of the last nonblank character are not included in the output, but would be restored if subsequently input to the same "variable, If the "formula contains spaces other than on the right, it is surrounded by primes as well as followed by a space on transfer to the character buffer, If the "formula contains primes, and the value is to be surrounded by primes, it should be noted that the result will not be legal input to the list-directed formatting routines, 1a2a7a

b, "Numeric:formula of floating type, The character representation of the floating number is a normalized, signed, fractional significand with a base 10 exrad, Trailing zeros are suppressed and the number of fractional positions is equal to (significant fraction bits/3,32)+1, The base 10 exrad is preceded by its sign and the letter _E, The syntactic form for list-directed floating output is: 1a2a7b

_+ _. digit " _E _+ digit " 1a2a7b1

c, "Numeric:formula of fixed or integer type, Fixed values with negative fraction bits or fraction bits greater than the number of significant bits are formatted as if they were "numeric:formulas of floating type (see b, above), For other fixed values and integer values, signed values are preceded with a plus or minus sign, The number of characters in the integer portion of the representation is equal to (integer bits /3,32)+1, The number of fractional digits is similarly computed, Integer and fractional

positions are separated by a decimal point. The syntactic form of the list-directed output is: 1a2a7c

```
_+     digit "  _.     digit "                    1a2a7c1

                _.     digit "                    1a2a7c2
```

d. "Bit:formula. List-directed output from "bit:formulas is undefined. 1a2a7d

.8 List-directed output is acceptable as list-directed input unless a "character:formula containing both spaces and primes is output. In general, it is not feasible to read list-directed output using format-directed input even though, for any given "constant found in a list-directed output field, there is a "format which can describe it. 1a2a8

.9 Given below are examples of a "character:Variable of five bytes output by list-directed formatting and then input from the output buffer. The input and output values are shown as JOVIAL "character:constants and are consequently delimited by "primes. The internal "primes are indicated by the three-"character code _$27. Note that the last input value is undefined because the output value contained a leading (embedded) blank and a prime. _b represents a blank or space. The leading field-separating blank is not shown; the trailing one is: 1a2a9

Output Value Input Value	Output Buffer	
'ABCbb' 'ABCbb'	ABCb	1a2a9a 1a2a9b
'bABCD' 'bABCD'	'bABCD'b	1a2a9c
'A$27Bbb' 'A$27Bbb'	A'Bb	1a2a9d
'AbBbb' 'AbBbb'	'AbB'b	1a2a9e
'bA$27Bb' undefined	'bA'B'b	1a2a9f

6.3 Format-Directed Formatting 1a3

With format-directed formatting, the picture of each character in the buffer is determined by its own "format, Except when "null:format is specified, each space and character is mentioned in the "format:list. 1a3a

.1 Basically, a "format tells how to interpret or generate a string of characters. A "format describes the data in a buffer, which may come from or go to an external medium; it is much like an "item:description as if applied to a "constant. The character strings are rather evanescent from the standpoint of the program, however; so the "formats must be applied to them dynamically--during actual execution of the program. 1a3a1

.2 Usually a string of characters is matched, through a "format, to a data element. However, there are some special "formats; the "insert:format describes character strings without reference to a data element, and the "skip:format causes a data element to be passed without matching it to a charter string. 1a3a2

6.4 "Insert:Format 1a4

"Insert:formats do not correspond to any data element. They may at times stand alone, particularly the form starting with _/, and so be separated from other "formats by "commas. However, they are most likely to be used as a part of another "format and in these cases are not separated by "commas. ("Insert:formats are not permitted as part of a "null:format or "skip:format.) 1a4a

.1 In general, a "format specifies a number of character positions filled in ways dependent on the value of the corresponding data element. Let us call these the "effective" character positions. An "insert:format specifies the content of other character positions inserted preceding, among, or following the effective character positions for a data element. The positions, but not the contents, of effective character positions are affected by "insert:formats. The contents of effective character positions are the same regardless of the presence or absence of "insert:formats. Obvious uses of such "insert:formats are for spacing on a printed page between values corresponding to data elements, for inserting commas between certain digits in a long number, or for titular information. There is no output of spaces not actually coded using an "insert:format, or as part of a character output value, or as a separator for "null:formats. 1a4a1

,2 The effect of "count is as if the following _S, _/ code, or "character string in "quotation:marks had occurred that many times. A "count of _1 is assumed if no "count is given. 1a4a2

,3 The use of _S is a shorthand way of indicating spaces, most often between data elements but possibly separating effective character positions. _S is equivalent to " ". The following five examples of "insert:formats are all equivalent and each would cause seven spaces to be placed in the output buffer or seven character positions to be ignored in the input buffer: 1a4a3

_SSSSSSS 1a4a3a

SSS SSSS 1a4a3b

_4S2SS 1a4a3c

5S 2S 1a4a3d

_7S 1a4a3e

,4 With "slash there is a difference. As an "insert:format, "slash followed by a "letter or "numeral is a system-dependent indicator of line, page, or other device control function. These may cause different results upon input and output. For example, the same code that causes the current record to be written upon output probably causes the next record to be read upon input. 1a4a4

,5 The "insert:format of a "character string in "quotation:marks is intended to permit any string of ASCII characters to be inserted as desired among effective character positions. It would seem preferable to use a "character:constant for the purpose, but "quotation:marks are used for delimiters instead of "primes because an "insert:format is a part of a "format:list which is itself a "character:formula and consequently could be a "character:constant delimited by "primes. So the use of the "quotation:marks as delimiters avoids the problem of having a "character:constant occur within a "character:constant. Before "format analysis takes place, any three-"character codes and special two-"character codes occurring in a "character:constant that will become part of the "format:list are replaced with the single "characters they represent. Additional analysis of "insert:formats

using "quotation:marks as delimiters is necessary in the formatting routines to permit the occurrence of a "quotation:mark as an insert "character. A single "quotation:mark as an insert "character is represented by two "quotation:marks in succession. In general, internal strings of "quotation:marks within the delimiting "quotation:marks must be halved to get the effective insert "characters. The "insert:format _"""" would therefore cause one quotation mark to be inserted (or one character position to be ignored on input). 1a4a5

.6 For input, "insert:formats delimited by "quotation:marks indicate positions in the character buffer to ignore. (There are no positions corresponding to the delimiting "quotation:marks or to half of the internal strings of two "quotation:marks.) 1a4a6

.7 For output, "insert:formats delimited by "quotation:marks indicate positions in the character buffer where the delimited character values are to be placed (the delimiting "quotation:marks and half of any internal strings of "quotation:marks are removed). 1a4a7

.8 The following three examples of "insert:formats are equivalent to each other: 1a4a8

_"/*/*/*/*" 1a4a8a

"/*" "/*" "/*" "/*" 1a4a8b

_4"/*" 1a4a8c

.9 The examples below are equivalent to each other but different from those just above: 1a4a9

"/* /* /* /* " 1a4a9a

4"/* " 1a4a9b

.10 Examples of the use of "insert:formats in connection with other "formats are given in the following sections. 1a4a10

6.5 "Skip:Format 1a5

A "skip:format causes the data element corresponding to this "format to be skipped on input or output. There is no corresponding field in the character buffer. 1a5a

6.6 "Character:Format 1a6

INSERT BOX 1a6a

,1 The "character:format is, in essence, a string of _C's. A "count of _1 is assumed if no "count is given. The total "counts of _C's is the number of effective character positions in the buffer. (Effective character positions are those specified by "formats but not including "insert:format characters.) 1a6a1

,2 For output, the character value of the output data element is placed in the effective character positions of the buffer. If the data element has more bytes than there are _C's, excess bytes on the right of the data element are truncated. If the data element has too few bytes, the field is padded with extra blanks on the right. For input, excess bytes from the field are truncated on the right or, if the field is sort, blanks are added on the right to match the size of the data element before assignment. 1a6a2

,3 An example of use of a "character:format is given in Section 6.7.3. 1a6a3

6.7 "Null:Format 1a7

A "null:format is permitted. It is indicated by means of an extra "comma in the "format:list. The "null:format consists of nothing but possibly a string of "spaces. 1a7a

,1 The "null:format indicates that conversion is the same as for list-directed formatting. For conversion purposes the input field starts with the first nonblank character at or after the current character of the buffer and ends with the first space after a nonblank character unless the first nonblank character is a prime; in this case the field starts with the character following the prime and terminates with the character that precedes the next prime. The current character for the next field will be the character following the space following the terminating prime. 1a7a1

,2 Output involving a "null:format is done in accordance with Section 6.2.6. 1a7a2

,3 The following table gives examples of two input buffer strings broken into fields in accordance with two "format:lists, the first consisting of two "null:formats followed by a "character:format and the second consisting of a "character:format followed by two "null:formats.

Since the description of the corresponding data element affects the meaning of the field, the "variable corresponding to each "null:format is assumed to be of character type and six bytes long. The "variable corresponding to the "character:formats is ten bytes long. The table shows the value of each field, as a JOVIAL "constant, determined from the "formats above and the input buffer strings on the left. (Note: _b represents a blank and _$27 is the three-"character code for a "prime within a JOVIAL "constant.) Blanks have been added or characters truncated to give "constants of the length corresponding to the "variable. 1a7a3

6.8 "Pattern:Format 1a8

INSERT BOX 1a8a

.1 The "numeral preceding _B applies to the whole "pattern:format and indicates the "order", the number of bits of the data element to be associated with each effective character position. The optional "count preceding _P simply is shorthand for indicating so many _P's. A "count of _1 is assumed if no "count is given. The total number of _P's gives the number of effective character positions in the buffer. The bit groups are associated with characters in accordance with the table of "pattern:digits (section 2,8,8). For each bit group in the table under "pattern" (as modified by "order") the letter or numeral under "pattern:digit is indicated for the corresponding effective character position in the buffer. 1a8a1

.2 Effective character positions are matched with bit groups starting at the right. On output, excessive bits at the left of the data element are discarded. If there are too few bits in the data element, zeros are added at the left before conversion to "pattern:digits. On input, the effective character positions of the field are treated as "pattern:digits, translated to bits, then assigned to the data element. Leading and trailing blanks are all treated as leading zeros and consequently do not affect the assigned value. Embedded blanks in the input field corresponding to the "pattern:format and any other characters not indicated as "pattern:digits for the indicated order are undefined. Blanks and other characters corresponding to "insert:formats are, of course, ignored on input. 1a8a2

.3 In the following table, the heads of the rows are

examples of character data element values whose bit patterns are shown in Section 6,8,4 below and the heads of the columns are "pattern:formats, _S's occurring in the "formats are "insert:formats. It is assumed that character values are stored in eight-bit bytes in which the first four bits correspond to the head of the column and the last four bits correspond to the head of the row in the table of characters (Figure 2-1). The body of the table below shows the contents of the output field corresponding to the character values and the "pattern:formats. 1a8a3

	4B3PS3PS3PS3P	5B5PS5P	1a8a3a
Bah!	000 042 616 821	00011 62Q11	1a8a3b
Humbug	487 56D 627 567	28ELM M4TB7	1a8a3c

,4 The above table is based on the following bit configurations, from the table of ASCII characters, ticked off in 4 bit groups by primes and 5 bit groups by commas: 1a8a4

Bah! 01,00'001,0'0110,'0001'0,110'10,00'001,0'0001 1a8a4a

Humbug
010,0'1000,'0111'0,101'01,10'110,1'0110,'0010'0,111'01
,01'011,0'0111 1a8a4b

,5 The table below gives examples of the values of JOVIAL "pattern:constants associated with fields of two input character strings in accordance with the list of "pattern:formats given at the top. These "constants could be assigned to "variables of any type. The _S after the first comma is an "insert:format and causes one character of the input buffer to be skipped. 1a8a5

Input Buffer	Field	_4B5P,S1BP,3BPPP	1a8a5a
bbAAbb1762	1	4B'AA'	
	2	1B'1'	
	3	3B'762'	1a8a5b
4A3CEb217b	1	4B'4A3CE'	
	_2	undefined because	
the "pattern:digit _2 does not occur for order _1			
	3	3B'17'	1a8a5c

6,9 "Numeric:Formats 1a9

"Numeric:formats provide the specifications needed to convert from internal numeric value representations to external character strings and vice versa. 1a9a

.1 It is not required that a data element and its corresponding "format have the same type. 1a9a1

.2 On output, the value of the data element is first converted to a decimal string of the same type (integer to integer, fixed to fixed, floating to floating). A decimal fixed value looks like a floating JOVIAL "constant without scale factors. Consideration of the "format can avoid generating unnecessary digits. If the "format calls for a different numeric type, the decimal string is then converted accordingly. A "generalized:numeric:format does not cause decimal-to-decimal conversion. If the "format calls for rounding, it is decimal rounding performed on the final converted decimal string. 1a9a2

.3 On input, the input field is converted first to its internal representation, then to the type of the data element. If an input string does not match the "format, the translation is undefined. The input string which is left after all character positions corresonding to "insert:formats are deleted must represent a legal JOVIAL "numeric:constant; there can be no embedded blanks in it. If the "format is a "generalized:numeric:format, the input characters may be integer, floating without an _E ("fixed:format), or floating with an _E, and all decimal points must be explicit. In general, values and "formats must be such as to prevent the loss of significant signs and most-significant digits, or results will be erroneous and generally unpredictable. 1a9a3

.4 Following is the list of "format "signs that may appear in "numeric:formats, with an explanation of their meanings: 1a9a4

_N A character position that may contain any character that is part of a legal numeric field. 1a9a4a

D An effective character position that will always contain a digit on output. On input it may contain a digit, a space, or a sign (+ or _-). 1a9a4b

Z An effective character position that may contain a digit or a space on output. On input, it may contain a digit, a space or a sign (+ or _-). 1a9a4c

_E The effective character position for the _E in "floating:format, or a space. 1a9a4d

_R Indicates that the digit string (significand in "floating:format) is to be rounded. On output, a decimal rounding is performed on the final converted decimal string. On input, a binary rounding is performed before the assignment to the variable. If _R is absent the digit string is truncated to the required length without rounding. No space is allocated for _R in the buffer. 1a9a4e

_+ An effective character position for _+ or _-, or space. 1a9a4f

_- An effective character position for _- or space. 1a9a4g

_. An effective character position for the decimal point, or a space. 1a9a4h

_* The position of the understood decimal point. No space is allocated for _* in the buffer. 1a9a4i

,5 No plus or minus sign is printed (unless specified by an "insert:format) in any numeric field containing a zero value. 1a9a5

,6 "Numeric:formats provide for print suppression (replacement with blanks) under certain conditions. Character suppression except for _+ or _- is tied to _Z. Nothing, other than _+ or _-, is suppressed in "formats not containing _Z. Any or all nonblank characters in "formats containing _Z's including insert characters may be suppressed depending on the actual suppression occurring in the effective character positions corresponding to _Z's. 1a9a6

,7 If an "insert:format immediately follows a _Z which is actually suppressed, the inserted characters are also suppressed. If an "insert:format follows a decimal point and immediately precedes a _Z which is actually suppressed, the inserted characters are also suppressed. 1a9a7

,8 The decimal point corresponding to _. in a "format is suppressed if the nearest effective character position on the right (in the same field) corresponds to a _Z and is actually suppressed. 1a9a8

,9 Plus and minus signs are movable in a suppress

context. If leading zeros corresponding to _Z's in the "numeric:format are suppressed on output, then the plus or minus corresponding to _+ or _- to the left of _Z's in the "format, and any "insert:format between the _+ or _- and the _Z's, are moved to the right the number of spaces corresponding to the number of leading zeros suppressed. In the example below we have a table showing an "integer:format (with four "insert:formats as part of it) at the top, and several output values on the left. The output fields corresponding to the "format and the values are given on the right. (Subsequent input of these output fields using the same "format would cause the inserted characters to be ignored and give back the original values on the left.) 1a9a9

```
        "K = " +$3Z","3Z","3DS"MPH"                          1a9a9a
    +523985612     _K = + 523,985,612 MPH                    1a9a9b
    -500005610     _K = - 500,005,610 MPH                    1a9a9c
    +000420000     _K =     + 420,000 MPH                    1a9a9d
     000000000     _K =           000 MPH                    1a9a9e
```

6,10 "Generalized:Numeric:Format 1a10

(box) 1a10a

.1 The optional "count is a way of indicating the number of _N's. A "count of _1 is assumed if no "count is given. On input, if the effective character positions corresponding to all the _N's contain any legal integer or floating JOVIAL "constant (corresponding to "integer:, "floating:, or "fixed:formats) it will be accepted, converted in accordance with its self-evident type, rounded in binary if the _R is present, and assigned to the "variable, 1a10a1

.2 The following table shows the values of JOVIAL "numeric:constants associated with fields of two input character strings in accordance with the list of "generalized:numeric:formats given at the top. The "minus:sign is not a part of the "constant but must be used in asssigning a value to the corresponding "variable, 1a10a2

.3 On output, conversion takes place as integer to

"integer:format, fixed to "fixed:format, and floating to "floating:format. Minus signs are output; plus signs are not. Significands and fraction digits are limited by the field size. If there is not enough room for all integer digits on output, the field is illegal. Excess fraction digits are truncated or rounded depending upon the absence or presence of the _R. Zeros are suplied between the point and the significand digits for fixed values with less than zero fraction bits or more fraction bits than the size. 1a10a3

6.11 "Integer:Format 1a11

INSERT BOX 1a11a

.1 As before, a "count followed by a _D or _Z has the same meaning as that number of _D's or _Z's, respectively. A "count of _1 is assumed if the optional "count is missing. "Insert:formats are permitted as described above. The "integer:format specifies an integer field of specific size in the buffer, for any numeric data element. For output, if the data element is not an integer, it will first be converted to a decimal representation, then to a decimal integer (rounding off any fraction if _R is in the "format, truncating the fraction otherwise). On input, the field is converted to internal integer form and then "assigned" to the data element (there can be no decimal point or fraction digits). If _R is in the "format, binary rounding occurs on assignment to "variables with missing low-order integer bits. 1a11a1

.2 The maximum number of decimal digits in the field is the total of the number of _D's and _Z's in the "format. If they are all _D's, all digits, even leading zeros, are printed on output or expected on input. If there are any _Z's in the "format, leading zeros are suppressed, but no more are blanked than there are _Z's. If the rightmost _D or _Z is a _Z, the non-blanked digits are left justified in the field defined by the _D's and _Z's. If the rightmost _D or _Z is a _D, the non-blanked digits are right justified and the sign, if there is one, is moved to the right by the number suppressed, leading zeros. 1a11a2

.3 _+ in the "format means print _+ if the value is positive, _- if the value is negative, and space if the value is zero. _- in the "format means print _- if the

value is negative and space if the value is zero or positive, 1a11a3

.4 In the table below, the values at the left are used with each of the "integer:formats at the top to give the printed outputs in the body of the table: 1a11a4

.5 The following table illustrates the use of "insert:formats with "integer:formats on input, The character strings in the input buffer are separated into fields in accordance with two "format:lists, The first has an "insert:format followed by an "integer:format and then another "insert:format, The "commas separating the "insert:formats from the "integer:format are not necessary, and only one field corresponding to a data element is described, The second "format:list consists of a "null:format followed by an "integer:format that has an "insert:format as part of it, Assuming that the "variable corresponding to the "null:format is of character type, the table shows the values of the input fields as JOVIAL "constants, 1a11a5

6.12 "Fixed:Format 1a12

INSERT BOX 1a12a

.1 A "count followed by _D, _Z, or _* has the same meaning as that number of _D's, Z's or _*'s respectively, A "count of _1 is assumed if the optional "count is missing, In the "fixed:format, the "decimal:point is the position of the actual printed decimal point and it occupies an effective character position in the field, A single "asterisk, however, is the position of an implied decimal point; the decimal point is not present in the character buffer, but the number is treated as if it were, If there is a sequence of _n "asterisks, where _n is _2 or more, it is understood that _n-1 trailing digits of the integer part of the buffer (beyond those specified by the "integer:part of the "format) are missing--or that _n-1 leading digits of the fraction part of the buffer are missing, Zero suppression is not permitted to the right of _*, the implied decimal point, in a "fixed:format, 1a12a1

.2 One effective character position corresponds to each _Z, each _D, a "plus:sign or "minus:sign, and a "decimal:point, but not to any "asterisks, The position of the decimal point, actual or implied, is fixed in (or outside) a field specified by a "fixed:format, It does

not change position (although it may be suppressed if the value is entirely zero) with changes in value or suppression of leading or trailing zeros. 1a12a2

.3 On output, the suppression of leading zeros and closely associated "insert:formats and the moving of plus and minus signs is the same as for a right-justified "integer:format. Trailing zeros corresponding to _Z's following the explicit decimal point are suppressed. If the zero fraction digit immediately to the right of the explicit decimal point or immediately to the right of an "insert:format is suppressed, then the decimal point or "insert:format is also suppressed. 1a12a3

.4 Considering only the effective character positions, anything can be input using a "fixed:format that could be output using that same "fixed:format or any legal "fixed:format derived from that one by replacing some of the _Z's with _D's. 1a12a4

.5 The presence of _R means the fixed decimal string is to be rounded on output to the decimal precision specified by the "format, or that the binary value is to be rounded on input before assignment to the "variable. _+ or _- means the same as in "integer:format. 1a12a5

.6 Output examples showing the use of each of the "fixed:formats at the top with each of the values on the left are given in the body of the table below: 1a12a6

.7 Subsequent input of the values shown in the above table in accordance with the "formats at the top would give the values shown below (note that precision may be lost due to rounding or truncation on input): 1a12a7

6,13 "Floating:Format 1a13

INSERT BOX 1a13a

.1 The "significand of a "floating:format is much like a "fixed:format, but it is more restricted. The only zeros that may be suppressed are trailing zeros to the right of an explicit decimal point. The _+ or _- have the same significance as in other "formats--they indicate what to do with the sign of the printed (output) "significand. "Count has the same meaning as in "fixed:format. 1a13a1

.2 The "exrad of a "floating:format is like an "integer:format with the added capability of the kind of

"fixed;format that produces integer output. An exrad is always an integer and this "format allows it to be output without zero suppression or with zero suppression and justified either left or right. 1a13a2

,3 If the _R is present, the significand of the value is rounded before input or output. The _E is output or expected in the position indicated in the "format and serves to identify the field as a floating value. The significand of the value precedes the _E as specified by the "significand of the "format. The exrad of the value follows the _E, again, as specified by the "exrad of the "format. 1a13a3

,4 If an output significand is zero, the corresponding exrad is also zero. If a zero significand is completely suppressed, so is an explicit decimal point. If a zero exrad is completely suppressed, so is the _E. If both the significand and the exrad are completely suppressed, so are all the nonblank characters in the field. 1a13a4

,5 Let _v be the value of a number output in accordance with a "floating;format. Then 1a13a5

$$v = m \times 10^{c} \qquad \text{1a13a5a}$$

where _m is the value of the output significand and _c is the value of the output exrad (including the mathematical sign in both cases). If the significand is not completely zero, the leftmost digit in the field and is not zero. With a given "format, every non-zero significand has an absolute value _M such that 1a13a6

$$10^{n} \leq M < 10^{n+1} \qquad \text{1a13a6a}$$

where _n is an integer determined entirely from consideration of the "format. The value of _n is zero or a positive integer equal to the number of _D's and "asterisks to the left of the "decimal;point or the rightmost "asterisk in the expanded "significand of the "floating;format--unless there are no _D's to the left of the "asterisk or "asterisks. If all the _D's are to the right of one or more "asterisks in the expanded "significand, then _n is negative (or zero) and its value is none minus the number of "asterisks. Thus, non-zero _M cannot be less than one tenth unless the "significand has an implied decimal point such that there are one or more missing digits on the left of the output significand. 1a13a7

,6 In the following example, two values are shown as they would be output in accordance with several "floating:formats, 1a13a8

6.14 "Format:List 1a14

INSERT BOX 1a14a

,1 Note that two definitions are given for a "format:list. First, a "format:list is a "character:formula with a value that is a string of "characters. This string of "characters must be analyzed by the formatting routines and recognized as a "format:list fitting the second definition. If the "character:formula is a "character:constant, it would be highly desirable to have the analysis done at compile time; if the "character:formula is not a "character:constant, the analysis must be done at execution time and increases the program's run time, 1a14a1

,2 The "format:list in "parentheses and preceded by a "count has the same significance as the indicated number of iterations of the enclosed list, separated by "commas. The number of "commas is highly significant with regard to indications of "null:formats. The expanded "format:list has "commas inserted between iterations, but not at either end. Example: 1a14a2

2(2(2S3Z "," 3D), 10C) 1a14a2a

,3 The above "format:list means the same as the expanded "format:list: 1a14a3

2(2S3Z "," 3D), 10C, 2(2S3Z "," 3D), 10C 1a14a3a

which means the same as the completely expanded "format:list of six "formats: 1a14a4

2S3Z "," 3D, 2S3Z "," 3D, 10C, 2S3Z "," 3D, 2S3Z "," 3D, 10C 1a14a4a

,4 After expansion of parenthesized "format:lists, a "format:list is a list of "formats separated by "commas. An "insert:format may, without "commas, be included in other "formats except "null:formats and "skip:formats. "Spaces in a "format:list, if not part of an "insert:format, are ignored. Since "null:formats are permitted, a legal "format:list may begin or end with a

"comma or have "commas together with no explicit "format between them. For example: 1a14a5

_,3S,,X15C,4B10P,, 1a14a5a

is a "format:list of eight "formats of the following respective types: 1a14a6

"null:format 1a14a6a

"insert:format of three spaces 1a14a6b

"null:format 1a14a6c

"skip:format 1a14a6d

"character:format of 15 characters 1a14a6e

"pattern:format of ten hexadecimal characters 1a14a6f

"null:format 1a14a6g

"null:format 1a14a6h

.5 The "format:list: 1a14a7

,3S,,X,15C,4BP "," 3P "," 3P "," 3P,, 1a14a7a

is the same as the above with the "insert:format _"," added to the "pattern:format for separating the digits with commas. These eight "formats correspond to seven data elements. The "insert:format _3S serves only to indicate three additional spaces between the first two data elements which had "null:formats. 1a14a8

.6 The rules do not permit omitting "commas between parenthesized parts of a "format:list. 1a14a9

_2(7D)3(6D) is not a "format:list. 1a14a9a

_2(7D) means the same as _7D,7D (two "formats). 1a14a9b

_2(7D,) means the same as _7D,,7D, (four "formats). 1a14a9c

.7 Upon execution of a call to the _FORMAT routine, the individual "formats of the expanded "format:list (including "null:formats and "skip:formats, but not stand

alone "insert:formats) are matched one by one, starting from the left, with the individual data elements. If there are more "formats than are needed, the extra "formats at the right of the expanded "format:list are not used. Even stand alone "insert:formats immediately following the "format matched with the last data element are left unused. If there are not enough "formats to match all the data elements, the entire "format:list is repeated as many times as necessary--it is as if the entire "format:list were enclosed in "parentheses and a sufficiently high "count prefixed to the parenthesized list. 1a14a10

.8 If data elements are output using an appropriate "format:list and then the same data elements are input using the same "format:list, the input values should be identical to the original output values except where precision might be lost due to the truncation or rounding of values. 1a14a11

6.15 Input and Output 1a15

Input and output "primitives are not a part of the JOVIAL language. Aspects of input/output are quite system dependent and, therefore, including input/output in JOVIAL would mean picking some particular input/output scheme or designing one especially for JOVIAL. Instead, JOVIAL is designed so that perhaps with the use of system routines, JOVIAL programs (or systems) can be made to interface with many input/output schemes. JOVIAL even provides the capability to program input/output systems. 1a15a

.1 The following is a list of JOVIAL capabilities as they might be used for input/output purposes: 1a15a1

a. Reference external procedures for _OPEN, _CLOSE, _READ, _INSERT, etc. 1a15a1a

b. Make system calls for performing actual read/write operations. 1a15a1b

c. Pass "parameters by address (tables and data blocks) for transmitting and receiving records. 1a15a1c

d. Core to core conversion (_FORMAT). 1a15a1d

e. Pass "character:formulas; e.g., file name. 1a15a1e

f. Symbol generation (parameterized define) for generating parameter tables, file control blocks, etc. 1a15a1f

.2 Core-to-core conversion using _FORMAT (see Section 6.1.3 and 6.1.7) is defined herein for data types of all kinds to and from character strings. This is useful for input/output involving character files but not binary files. Since core-to-core moving of bit strings is intrinsic to JOVIAL, nothing corresponding to _FORMAT is needed for binary records (use data blocks or tables). 1a15a2

.3 Core-to-core conversion of numeric values between binary and decimal is carried out using standard algorithms. Conversion of integer values is exact if the capacities discussed below are not exceeded. Binary to decimal conversion applies only to formatting. The limit on accuracy is determined solely by the description of the output field. Enough extra digits are produced by the converson method to insure the accuracy implied by the field width and the rounding or truncating expected. A correction increment is added to the appropriate fraction digit before rounding or truncating to compensate for inherent losses. 1a15a3

.4 Decimal to binary conversion applies both to formatting and to the derivation of internal representation for "numeric constants. Enough extra bits are developed to insure the accuracy implied by the size of the receiving registers, with the following limitation. No implementer is required to use more magnitude bits than 150% of the maximum size for the representation of numeric values. The 150% figure is the most required of the total of integer and fraction bits. The binary point may be anywhere within or to the left or right of these bits. 1a15a4

DLS 23-MAY-74 08:12 30787

IBM SP Effort (,30588)

The Journal may not have been the appropriate vehicle for disseminating my thoughts on the IBM effort. However, feel free to use anyway you see fit. I was trying to get the attention of some of the management around here, so some of the comments may seem caustic. They do represent my impressions...when I find out more about the effort, they may change. I would appreciate any thoughts that Dick (or anone else) has on the use of NLS (from tool through philosophy) to support the programming process. Do you have anything handy that talks about superdocuments? The exchange of briefings that I mentioned earlier with IBM, is scheduled for the 5&6th of June. We may visit Carlson while there and possibly Crocker if local management thinks its adviseable. Who again, should I contact about using the line processor? We may want to give a demo if its not too difficult to arrange. If we can make the connectons, I would go down a day early to practice with LP. Will keep you posted on progress. 1

Lynn:
The following is for Chris Beaty,

First let me give you the names of a few people who might be interested in your ideas:
Doug Englebart/ Augmentation Research Center/ Stanford Research Institute/ Menlo Park, California 94025,
Mil Jernigan/ same address,
Dave Crocker/ 3804 Boelter Hall/ Computer Science Department/ UCLA,
Clayton Greer/ Computer Systems Laboratory/ UCSB/ Santa Barbara, California 93106,
Jean Iseli/ The MITRE Corporation/ Westgate Research Park/ McLean Virginia 22101,

There are a few comments on the collected papers that i hope i can make coherently. First there seem to be contradictory uses of the memory -- the idea that only public information would be in the memory (thus avoiding the privacy problem) but that individules could use the memory to produce income. These are random thoughts -- the idea of human gatekeepers gives me two images: first of a priesthood that controls access to the memory (much as priest control access to god), and second as telephone switchboard operators (the telephone company wants to reduce the dependance on human operators because they are expensive, your gatekeepers would be highly trained and have to think about each transaction thus very expensive). Your examples are typicall want ad stuff, what is the advantage of electronic want ads? Newspapers are still cheap. There is a confusion about actors -- messages dont act people and (maybe) programs act. also some of the things you talk about suggest that there is a computer involved in more than inserting and reteriving data. Perhaps you are really suggesting a computer utility service for the public? About how to make this happen -- i would suggest a foundation grant to get it started.

1

JBP 23-MAY-74 11:37 30794

Bill:
I was talking with some people here (Trehan - he visited you guys) about the traffic patterns in the network and mentioned the matrix displays you created using overprinting to indicate the sending receiving pairs. Well they got real interested and we would like to see a few samples, could you send us a few?
--jon.

1

norton's comments on compiling REL files

(NORTON) 10-MAY-74 1103-PDT L10: Output to REL files
Distribution: BEDFORD, norton
Received at: 10-MAY-74 11:03:45 1

Mike: OK I've found out about the output compiler to L10 comand, First, as in the following example program, you need to replace the word "PROGRAM" at the start with the word "FILE", Then you need to be positined at the statement with the word FILE (so the system knows the exact starting place) and you need viewspecs all-all (w) and no statement numbers (n) set, Then you type O [utput] C [ompiler F:] l10 [F:] name where name is the name of the first procedure in the program as i
the following example, the system will do its thing with some stats coming back and you should be left with a rel file, Then you can use it by the go to programs load (relfile) name command, Let me know if you have troubles,,ok? or link and well go thru it, Ill leave most of these to dean in the future, but want to settle this one right now see you Jim
heres the example:

```
FILE name
   DECLARE TEXTPOINTER pt1,pt2;
   DECLARE STRING txt1[20];
   (name) PROCEDURE;
      IF FIND '( "pt1 1sLD "pt2 ') THEN
         BEGIN
         *txt1*  _ pt1 pt2;
         IF FIND [*txt1*] THEN RETURN (TRUE);
         END;
      RETURN (FALSE);
      END.
   FINISH
(this) is this a test ?
   (NO) this is not a test
      (test) It could have been a test,
```
1a

EXAMPLE OF BIBLIOGRAPHIC CITATIONS 1

The format for citation might look as follows: 2

(BELLn) *a1 author name #2 author's organization #3 sub-organization #4 street address #5 city, state zip *b2 publishing organization #3 suborganization *c1 title #1 subtitle #6 pages *d1 date *f1 type *f2 media *p1 project title *s1 contracting org #1 contracting agent #7 project number *w4 source *y1 abstract *y4 key; word; list; key word list; *z3 new * 2a

Rules: 3

The "n" in "(CBIn)" is a sequentail number which will give us a handle on the citation. Increment it by one for each citation. 3a

Fields begin with a space, then a star *. Subfields begin with a space, then a pound-sign #. The citation must end with a space, then "*z3 new *". 3b

Sub-fields may be left out if desired. 3c

"org" should be put in *a1#2 when the author's organization is the same as the publisher (*b2). Use the code "*b4" instead of "*b2" if the publisher is a commercial firm (and leave out the suborganization, of course). 3d

Information which is known and of importance but not actually part of the report should be put in square brackets []. Less important or sure information should not be included. 3e

Names are entered in their proper order (first name first). Titles (Dr, Ph.D.) should be omitted. Follow initials with the usual period and space. 3f

You may list any number of authors, in similar fields with similar subfields, as *a2, *a3, etc. 3g

The date field should never be left blank. If the date isn't known, approximate in square brackets, e.g. "[Fall 1973]". A date should be of the form: 24 April 1974. 3h

Use the *w4 field if the source is different from the author (a1) or from the publisher (b2/b4). 3i

Example (this isn't a real entry, I made it up): 4

(CBI73) *a1 Richard W. Watson #2 Stanford Research Institute #3 Augmentation Research Center #4 333 Ravenswood Ave. #5 Menlo Park,

Example of Bibliographic Citations; for Bell Canada

California 94025 *b2 SRI-ARC *c1 NWG/RFC 592 #1 Some Thoughts on System Design to Facilitate Resource Sharing #6 4p, *d1 20 November 1973 *f1 r *f2 o *p1 Augmenting Human Intellect *s1 ARPA #1 Craig Fields #7 00000-00 *w4 Network Information Center *y1 Discussion of philosophy of system design in the ARPANET,,, *y4 #1 ARPANET; resource sharing; systems design; network design; network resources; unified design; *z3 new * 4a

PROGRAM NAME: CONTR

PROGRAM NAME: CONTR 1

The program enters the IDS Data Base through the Enter-Contract record, It then retrieves all related data from the contract and purchase request funding records and prints all of it in report form, 2

The program can be called from the GECOS Time-Sharing System under userID ABIHIDS1, When the system asks "Old or New" type OLD ABIHIDS1/IDSPROG$IDS/IDSSRC/CONTR, <cr>; when the "ready" is given followed by a system <cr> and asterisk herald then type RUN <cr>, 3

The program listing is as follows: 4

```
0020    $    IDENT   ABIHIDS1,OLD/CONTR,55500415RADC,CONTRACTS                5
0030    $    USERID  ABIHIDS1$ANAA                                            6
0040    $    SELECT  ABIHIDS1/CONTROBJ                                        7
0050    $    EXECUTE DEBUG                                                    8
0060    $    LIMITS  01,17K,,9K                                               9
0070    $    PRMFL   PC,R,R,ABIHIDS1/IDSDATA$IDS/BACKFILE$IDSB/BACK3          10
0080    $    PRMFL   PD,R,R,ABIHIDS1/IDSDATA$IDS/BACKFILE$IDSB/BACK4          11
0090    $    SYSOUT  OU                                                       12
0100    $    ENDJOB                                                           13
```

PROGRAM NAME PR3

PROGRAM NAME PR3 1

The program enters the IDS Data Base through the Enter-PR record, It then retrieves all related data from the purchase request, purchase request step, work unit, applied-man-hours, contract, and purchase request funding records and prints all of it in a report form, The applied man-hours are added together giving total applied man-hours, 2

The program can be called under the GECOS Time-Sharing system under userID ABIHIDS1, When the system asks "old or new" type OLD ABIHIDS1/IDSPROG$IDS/IDSSRC/PR3 <cr>; when the ready is given, followed by a system <cr> and an asterisk herald then type RUN <cr>, 3

The program listing is as follows: 4

```
0020    $   IDENT   ABIHIDS1,OLD/PR3  ,5550041SRADC,PRS TO MANHOURS      5
0030    $   USERID  ABIHIDS1$ANAA                                        6
0040    $   SELECT  ABIHIDS1/IDSPROG$IDS/IDSOBJ/PR3OBJ                   7
0045    $   EXECUTE DEBUG                                               8
0050    $   LIMITS  30,19K,,15K                                         9
0060    $   PRMFL   PC,R,R,ABIHIDS1/IDSDATA$IDS/BACKFILE$IDSB/BACK3     10
0070    $   PRMFL   PD,R,R,ABIHIDS1/IDSDATA$IDS/BACKFILE$IDSB/BACK4     11
0080    $   SYSOUT  OU                                                  12
0090    $   ENDJOB                                                      13
```

NDM 24-MAY-74 17:10 30807

Producing Sample Catalogs for ETS

Check with me if any confusion, Would like to be able to hand to Norton to show O'Sullivan (ARPA) when Jim is here next week. Tnx.

NDM 24-MAY-74 17:10 30807

Producing Sample Catalogs for ETS

Beau: I sent you the first half a dozen entries that ETS has made for thier biblio database. We would like to show these people some examples of the types of catalogs would could produce for them. Can you please make whatever we can of this? We may have to modify the programs a bit. The statement names are different, the *pi field has subfields, and there is a *y4 field with four subfields representing four sets of keywords. For now we can ignore the hard stuff and just kudge around minor differences from standard. If you adjust the standard programs, save them for future use. Let me know when and I can print the files you produce here in DC. Thanks. --Dean 1

Chapter 7 1

"DECLARATIONS 1a

7.1 Introduction 1a1

In this chapter, it will be seen how "names are associated with structures in JOVIAL and how definitions are provided for those structures via the various "declarations. But before getting to the declared "names, it is well to consider some "names which require different treatment. 1a1a

7.2 Undefined and Predefined "Names 1a2

Not all "names depend upon "declarations for their definition. "Names can be defined by their appearance in a "program:declaration and "names can be predefined. 1a2a

.1 A "statement:name is defined by its appearance (followed by a "colon) at the beginning of a "statement (and one or two other places as described in Section 5.4.3). It is, thereby, defined as the "name of the next "statement. In a "procedure:declaration, a "formal:input:parameter can be a "statement:name. How this comes about is explained in Section 8.5. 1a2a1

.2 "Names may be predefined for a "program:declaration. A reference to such a "name causes the compiler to seek its definition from a source external to the "program:declaration. "Item:names, "table:names, other "program:names, "procedure:names, "define:names, in fact, "names of any kind of JOVIAL entity can be predefined by means of a compool, a library, or both. The distinction between a compool and a library is arbitrary and beyond the scope of this language manual. However, while it may be enough for purposes of language specification to state that two arbitrary structures have been named to satisfy the requirements of external predefinition and leave the details of implementation to the implementers, it may enhance understanding to expand the concept by means of a possible implementation. In this traditional scheme, a compool is a table or dictionary of definitions for use by a system of related programs. If a program is to be integrated into the system, the descriptions and locations of common data, procedures, and programs are found in the compool. A library, on the other hand, does not contain descriptions, but rather complete procedures or "procedure:declarations. If a "program:declaration calls one of these procedures or "procedure:declarations,

it is copied from the library and effectively made a part of the program. Definitions to be associated with "names as if through the use of _DEFINE may occur in either the compool, the library, or both. 1a2a2

,3 Together, the compool and library serve as a description of the external environment that may be of interest to a "program:declaration being compiled. This concept does not require that their content be in any particular format--only that the format be "known" to the compiler. There is no requirement that they be unitary or that all information be in the same format. Some parts may be public information, other parts may be private and maintained by individual programmers. There may exist separate files containing "item:declarations, "table:declarations, "define:declarations, symbolic "procedure:declarations to be compiled as part of the program, binary procedures to be included in the program, in-line subroutines in symbolic or binary, link information for system subroutines, etc. This concept neither requires nor precludes a system data base to be present during execution. 1a2a3

,4 If a "program:declaration written in JOVIAL makes reference to a "name defined in the compool or library and if this reference is compatible with the compool or library definition, then the reference is taken to be a reference to the compool or library-defined "name. Any such referenced "name must be listed (either explicitly or by construction) in a "compool:directive at the beginning of the "program:declaration (see Section 11.2). If, however, the "program:declaration properly defines such a "name explicitly, then, if there is a conflict, this definition takes precedence and the compool or library definition is disregarded. "Proper" definition has reference to the necessity of placing "data:declarations ahead of any references to them. In normal usage, procedures and functions can be referenced before they are declared. However, when a local "procedure:declaration is intended to override a compool or library "procedure:declaration, proper definition may require the local "declaration to precede any reference to the procedure or function. 1a2a4

7.3 Scope of Definition of "Names 1a3

Before considering the various "declarations, it is important to understand the concept of scope because it is this concept that determines the portion of the

"program:declaration or system of "program:declarations in which the declared "name is active. The scope of a "name then is defined as that segment of code over which a "name has meaning. 1a3a

.1 The scope of a "name is determined by the area of the "program:declaration or system in which the "name is defined. In JOVIAL there exists a hierarchy of scopes. Starting with the broadest, the scopes are named compool, external, main and procedure. 1a3a1

.2 "Names declared in compools and libraries are of compool scope. (Local "names in library procedures remain local, of course.) They are available to all "program:declarations compiled under the influence of the compool or library. References in these "program:declarations to such "names are taken to be references to the associated structures provided the "names have been properly identified in the "compool:directive and are not masked by "declarations of identically spelled "names within the "program:declaration. All "compool:directives begin the "program:declaration and serve to establish a compool scope outside the "program:declaration in which the "names indicated in the "directive are assumed declared. This provides for overriding any "name declared in a compool at any level of source program scope; a "declaration is in effect for the local scope at which the "declaration occurs as well as any inner scopes not containing a declaration for the same "name. "Table:names, "item:names, "data:block:names, "statement:names, "procedure:names, "status:list:names, "form:names, and "define:names may all be declared in a compool. 1a3a2

.3 External scope covers those entities (items, tables, data blocks, procedures, "statement:names) which while declared in a program are flagged in the "declaration as being common to more than one program by the presence of one of the "primitives _DEF or _REF (see Chapter 9). External scope assumes the presence of some linking loader which will resolve all such external references. 1a3a3

.4 Within a "program:declaration, scopes are defined by the "program:declaration itself and also by all "procedure:declarations within the "program:declaration. Data declared in the "program:declarations but not within any "procedure:declaration is of main scope. Data

declared within a "procedure:declaration is of procedure scope. 1a3a4

.5 Notice that the definition of procedure scope allows unlimited nested levels. "Procedure:declarations can be nested one inside the other and the scopes defined by each are likewise nested. All such nested scopes are called procedure scopes regardless of the level of nesting. 1a3a5

.6 The above scope nomenclature is absolute. There is often a need for relative scope terminology. The relative terms local, outer, and inner allow scope to be discussed in relation to any particular point in a "program:declaration or system. Local scope refers to "names declared in the same scope as the reference point. For example, the "formal:input:parameters of a procedure are local to the "procedure:declaration. Outer scope refers to "names declared in a more extensive scope than the scope of the reference point. For example, the "names of compool data and main program data are in an outer scope with regard to a "procedure:declaration within the "program:declaration. A "procedure:name is considered to be outer with respect to the "procedure:declaration that bears it. Inner scope refers to "names declared within a more restricted scope within the scope of the reference point. For example, data declared in a nested "procedure:declaration is inner with respect to the "program:declaration. 1a3a6

.7 It is a basic rule of JOVIAL that "names may not be multiply declared in the same scope. There may not, for example, be a "table:name and a "statement:name the same in a single scope. Each "name must be unique within its scope. 1a3a7

.8 While uniqueness of "names is required within a scope, "names can be repeated in different scopes. To the compiler, a "name comprises both a spelling and a scope; "names with the same spelling but different scope are easily distinguishable and not the same "name at all. This has certain advantages for JOVIAL programmers. It allows a freer choice of "names within a "program:declaration. The programmer is able to create "names without regard for total uniqueness; his worries are confined to the local scope. Also, an independently created "procedure:declaration can be incorporated into a "program:declaration without fear that its local "names

will be in conflict with any identically spelled existing "names. 1a3a8

.9 The scope of a "name local to a "procedure:declaration is the "procedure:declaration in which it is defined and all contained "procedure:declarations that do not have their own definitions of the "name. A main scope "name is defined at any place in the "program:declaration where inner "procedure:declarations do not have local definitions of the same "name. 1a3a9

.10 "Names explicitly defined within a "procedure:declaration are local to that particular "procedure:declaration. This includes all "formal:input:parameters and "formal:output:parameters (see Chapter 8). Conflicting local definitions within a particular "procedure:declaration are not allowed. A "procedure:name (and an "alternate:entrance:name) is of outer scope to the "procedure:declaration that it names. 1a3a10

.11 In general, "names of local scope are not available in outer scopes. "Names of outer scope, however, are available in local or inner scopes provided the "names have not been redefined locally. Any such local "declaration would mask the outer scope definition for the local scope and any inner scopes. Local definitions also serve as outer scope definitions for any nested scopes that, once again, do not have their own local definitions. 1a3a11

.12 Resolution of a "name of outer scope used in a nested body of code begins from the current or local scope and works outward accepting the first "declaration encountered. For example, in Figure 7-1, "table:name _XX is of main scope and local to the "program:declaration. It is of outer scope to "procedure:declaration _EE. _XX as an "item:name is local to "procedure:declaration _BB and outer to both "procedure:declarations _CC and _DD. Within _DD the prevailing "declaration is the one closest to _DD in an outer direction. 1a3a12

.13 "Program:names, "define:names, "form:names, "status:list:names, "data:block:names, "item:names, and "table:names, but not necessarily "statement:names or "procedure:names, that are to be defined by "declaration must be declared before they are used in their respective scopes. 1a3a13

7,4 "Declarations 1a4

"Declarations are the principal means of naming and defining the various parts of a program. 1a4a

.1 "Processing:declarations declare programs and procedures and, together with "form:declarations, are the subject of Chapter 8. "External:declarations cut across "data:declarations, the "name:declaration, and "processing:declarations and are discussed separately in Chapter 9. The others are discussed in this chapter. 1a4a1

7,5 "Null:Declaration 1a5

The "null:declaration is a means for satisfying a language requirement for the appearance of a "declaration even when no significant "declaration is desired. 1a5a

.1 The "null:declaration can, for example, be used as a "specified:table:body to satisfy the requirement that the "specified:table:heading be followed by a "specified:table:body. In this case, the table would not have any items; the utility of the table would have to derive from the entry structure specified in the "specified:table:heading. 1a5a1

.2 Note the optional use of the "primitive _NULL. Although the "semicolon alone is a "null:declaration, some users may not choose to have a "mark so small take on so much significance. Those who feel that using the "primitive _NULL will make the "program:declaration more readable and less error prone are encouraged to do so. 1a5a2

7,6 "Data:Declarations 1a6

"Data:declarations serve to declare and describe the data on which a program is to operate--the inputs, the initial elements of information, the intermediate results, the final results, and the outputs. The "names given to the data follow the "primitives that begin the "declarations. They are chosen at the arbitrary discretion of the programmer (or programming supervisor) and have no necessary connection with names used in the outside world--on input manuscripts or printed output, for instance. 1a6a

.1 Much of the remainder of this chapter is concerned with the specification of the various kinds of "data:declarations. First, some concepts common to all

data structures are developed. Then the individual "declarations are considered. 1a6a1

7.7 Fixed and Controlled Allocation 1a7

The data structures about to be described in the various "data:declarations are materialized in data space. There are two aspects of this process. First, space must be provided to the program and second, the data structure must be associated with that space. Depending upon the nature of a data structure these two aspects may either merge and tend to appear as a single process performed by the compiler or they may remain distinct from each other and require the involvement of the program employing the data structure. 1a7a

,1 There are generally three ways to get data space for a program. One way is to compile it into the program. A second way is to provide space during the program loading process. The third way is to request it dynamically from the system at execution time. In addition to the ways of obtaining space, there are two basic ways of associating a data structure to that space. First, the association can be made by the system. This is known as fixed allocation. Second, the association can be made dynamically by the program during the operation of the object program. This is known as controlled allocation. 1a7a1

,2 Fixed allocation is achieved by declaring the structure (either in the compool or the program) without indicating that controlled allocation is to be applied to the structure. The association between the space and structure is determined by the system, generally fixed at compile time with the space provided during the load process. (Whether the space is obtained at compile time or load time is only dependent upon whether the space is part of the program's own environment or part of another object module's environment. Actually, in either case, the loader provides the space.) 1a7a2

,3 Controlled allocation of data space means that space assignment for a particular data structure is established by the program dynamically at run time as opposed to being fixed during compilation or during linking and loading of the program, or by the system during activation of a scope. Data structures are identified as controlled allocation structures by the inclusion of an "allocation:specifier within the "data:declaration. Pointers are thereby established which locate the data structure. 1a7a3

.4 Controlling the location of data structures within dynamic space is left to the JOVIAL programmer who, by assigning values to pointers, establishes at least temporary allocation of a pointed-to structure. Each reference to a variable declared to have controlled allocation must employ, either explicitly or implicitly, an associated pointer. The value of the pointer will be used in the calculation of the effective address of the structure. 1a7a4

.5 Dynamic association is independent of when space is obtained. With dynamic association, (i.e., controlled allocation) the association of structure to space actually occurs when the value of the pointer to the space is appropriately established. Space can be received from the system or from some large block in the program's own environment, but attaching a structural definition (table, item, data block) to the space is performed during the program's execution by setting the value of the pointer to the structure to be equal to some address in the space. 1a7a5

.6 Considerable system support is required to provide the capability to get space at program execution time but the method of associating a structure to the space is the same whether space is loaded with the program's environment or whether obtained dynamically. JOVIAL does not provide a special "primitive for getting space dynamically. If a given system has the capability to provide space dynamically, there presumably would be a system procedure that could be called to supply the space. Its input "parameter might be the size of the space requested, and its output "parameter might be the pointer which is to receive the address of the space. For example, 1a7a6

```
GETSPACE (10 ; POINTER1) ;                    1a7a6a
```

might call a procedure named _GETSPACE to provide ten words of storage for the caller. The address of the storage would be returned in _POINTER1. 1a7a7

.7 In systems with the dynamic storage ability there might be a wide range of procedures for management of the space. Procedures are required for releasing space as well as reserving it, and for "garbage collection" and reassigning ownership of space from one program to another. 1a7a8

,8 Data space which is compiled into the program or which is reserved at load time can have controlled allocation structures dynamically associated with it just as can space which is reserved at execution time. If a program were required to operate on several different data structures but could arrange its processing sequence to finish processing some of the structures before starting to process others, data space in the program could be kept to a minimum if the program were to use controlled allocation to overlay different structures on the same space. Such space can be reserved at compile time by declaring a table or data block large enough for the requirement. The structures to be processed would be declared for controlled allocation and at appropriate points in the execution of the program the values of the pointers to the controlled allocation "variables would be set to equal addresses within the large table. 1a7a9

,9 The differences between fixed allocation and controlled allocation structures must be taken into account by the JOVIAL programmer. Consideration must be given, for example, to the relative costs of indirect addressing, allocating, associating, and releasing space under programmer control or (if available) under system control, and the alternate costs of increased static storage. 1a7a10

,10 Lists, stacks, queues, and other linked data structures can be created and manipulated using controlled allocation. The basic requirement in such structures is that part of the elementary structure of a hierarchy contain control information. JOVIAL 73 provides data structures and manipulative "statements that permit any required degree of forward, backward, sideways, and cross linking. Although such activities can be accomplished using JOVIAL procedures, it is often more efficient if system procedures are provided that have been coded in machine language to mesh well with the local operating system. 1a7a11

7,8 Pointers and Their Association with Structures 1a8

Controlled allocation structures are also called pointed-to structures. This name expresses the requirement that there be a pointer that locates these structures as space is dynamically allocated them. The pointer is expressed as a "pointer:formula. 1a8a

,1 "Pointer:formulas describe the address of pointed-to

structures. The "pointer:formula may contain "pointer:variables as well as other "numeric:variables and "constants. The result of the evaluation of the "formula (truncated to an integer) is the pointer value used in referencing the structure. ("pointer:formula is simply a special case of the "numeric:formula--one which evaluates to a location.) 1a8a1

.2 If the "pointer:formula is not a "constant, it is evaluated at each reference to the pointed-to structure. Even unto the nth generation--if the pointer is pointed to, .B=1;its.B=0; pointer is evaluated at this time, etc. If the "pointer:formula is anything more than a "constant or a "name, it must be enclosed in "parentheses. 1a8a2

.3 A "pointer:variable is a special case of the more general "pointer:formula. 1a8a3

.4 A "pointer:variable is a storage element which contains an address of some program element. Its "declaration and usage is as an unsigned integer item, and all the rules which apply to unsigned integers also apply to "pointer:variables. A possible syntax for explicitly declaring a "pointer:variable is: 1a8a4

```
_ITEM  "name  _@        "pointer:formula
U  ;                                        1a8a4a
```

The "name becomes an "item:name. Type size for the "pointer:variable need not be declared; the compiler will supply the appropriate size for a pointer on a given system. "Pointer:variables can be indexed. In other words, they can appear in "table:declarations. In some cases, an indexed "pointer:variable is required. Also, as implied by the above syntax, "pointer:variables can themselves be pointed-to. 1a8a5

.5 The compiler does not require special awareness of "pointer:formulas except for their association with the data structures to which they point. The necessary aspect of pointers is their association with data structures and the implication that their content is an address. 1a8a6

.6 There are two ways in which associations of pointer and controlled allocation data structures can be formed. They can be formed either in the "declaration of the structure or by explicit scripting at point of reference. 1a8a7

,7 An association established in a "data:declaration is in effect unless an override (explicit association at reference) is encountered. At every reference, the established "pointer:formula is supplied automatically by the compiler. The programmer need only script the "variable and the compiler will supply the associated pointer. 1a8a8

,8 If no pointer is established in the "declaration of a pointed-to structure, then one must be supplied with each reference to the structure. Such explicit scripting of a pointer also serves to override a declared pointer for the current reference. Association at reference (explicit association) is achieved by the following syntax: 1a8a9

```
"name      _(    "index      _)  _@
"pointer:formula                          1a8a9a
```

In this form, "name is the "name of a controlled allocation item or table. If the named entity is in a data block or a table, the "pointer:formula is assumed to point to the first word of the data block or table. The value of the "pointer:formula is used instead of the value of any implicit pointer from an association within the "declaration. 1a8a10

7.9 "Allocation:Specifier 1a9

The "allocation:specifier is an optional part of "simple:item:, "table:, and "data:block:declarations. Its appearance in a "declaration marks the concerned data structure as a controlled allocation entity. 1a9a

,1 The _@ "ideogram is the only required "symbol. This "ideogram marks the data structure pointed to and signifies that it will receive dynamic allocation. 1a9a1

,2 The "pointer:formula, if present in a "data:declaration, provides the location of the structure. Its presence in a "declaration establishes an association between the data structure and the pointer such that the pointer is employed automatically with each reference to the data structure unless an explicit override occurs at some subsequent point. 1a9a2

,3 Absence of the "pointer:formula indicates that while the data structure is indeed pointed to, there is to be no pointer associated with the structure at the point of

"declaration. The required association must be made explicitly at some subsequent point during reference. 1a9a3

.4 All "variables in a "pointer:formula must be declared prior to being referenced within a "data:declaration. 1a9a4

.5 The "variables in a "pointer:formula associated with some structure at its "declaration may, but need not, have wider scope than the structure "name being declared. They must, of course, be known at the point of the "declaration, and thus have scope at least as wide as the structure "name. If the structure "name is redeclared in some inner scope, then, of course, the original structure cannot be referenced in that inner scope. If, however, the structure is referenced in some inner scope in which one or more of the "variables in the associated "pointer:formula are redeclared, the inner scope meaning of such "variables is ignored. More specifically, wherever an implicitly pointed-to structure is referenced without a explicit "pointer:formula, the meanings of all the "names in its associated "pointer:formula are those in effect at the "declaration of the structure. On the other hand, if there is an explicit "pointer:formula associated with the structure "name at the point of reference, the current scopes of all the "names explicitly stated are in effect. 1a9a5

7.10 Data Permanence 1a10

Data that is allocated in a fixed manner (as opposed to controlled allocation data) may be considered to be either private or environmental to a given scope. Data that is private to a scope is available only while that scope is active. Data that is environmental to a scope is protected even while that scope is not active. 1a10a

.1 Data environmental to a scope is of greater permanence than data private to the scope. It is private to some outer scope and is protected while that outer scope is active. 1a10a1

.2 Data that is environmental to the main scope is said to be "reserved" or "in reserve." All data declared in a "program:declaration, whether it be of main scope or procedure scope, automatically becomes reserved unless its permanence is restricted by being made private to some inner scope by the occurrence of an "environmental:specifier. Data that is to be initialized or preset must be in reserve. Reserved data are

allocated when a program is loaded and remain until the program is unloaded even though the program is not active (executing). A program can be entered and exited many times and at each entry it can rely on the validity of reserved data as long as the program has not been unloaded. The loading and unloading is a function of a program's use as part of link-edited segments and of the loader employed by the operating system; in short, loading and unloading is system-dependent and outside the scope of the language. 1a10a2

.3 Data of compool scope that is not pointed to exists independent of any program and is considered to be environmental to all programs and therefore in reserve. Similarly, external data is environmental to all referencing programs as well as the program in which it is declared so it too is in reserve. 1a10b

.4 Data which exist and must be protected only when a procedure or program is active are known as private data--private to the procedure or program. The compiler, or system, of course, is not required to destroy private data when its procedure or program is not active, but it may do so. Private data comes into existence when a scope is activated and disappears when the scope is left (leaving may be effected by _RETURN, _STOP, or a branch to a parameterized "statement:name, external "statement:name or an outer scope "statement:name). Reentrance to the scope again activates the private data space; however, values left in the private data at the previous exit cannot be assumed valid. An implementation may choose to dynamically allocate private data at each entrance to the scope. Notice that entrance may be effected by invoking the scope (program or procedure) by its "name or by calling an alternate entrance to the scope. (Branching to a "statement:name defined for external reference within a scope does not activate the scope. The scope must already be active--perhaps it had invoked a reciprocally declared external procedure.) 1a10c

.5 Data generated by the compiler incidental to the processing of forms other than "data:declarations become part of the unnamed data space of the procedure. This may include things such as temporary procedure space, register save areas, return address, perhaps some parameter space, and other linkage convention space. As a part of the procedure's data space they will be in reserve unless the entire data space is restricted by being made private to some lesser scope (see Section 8,6). 1a10d

7.11 "Environmental:Specifier 1a11

The "environmental:specifier is an optional part of "data:declarations. It is normally used to restrict the permanence of data by making it private to some scope. Without the restrictions imposed hereby the data being declared would become environmental to the program and in reserve. 1a11a

.1 The "program:name and "procedure:name serve to name the scope to which data is to be private. This scope must either be the scope of "declaration or some outer, containing scope; it cannot be an inner or disjoint scope. If neither _RESERVE nor a "program:name or "procedure:name is included, the data is made private to the local scope. 1a11a1

.2 The entire environment of a procedure can be made private (see Chapter 8) by so stipulating the "procedure:declaration. An "environmental:specifier in a "data:declaration --either a "data:block:declaration, "table:declaration, or "simple:item:declaration --affects only the associated data structure. If a procedure's data space is made private to some scope, particular data can nevertheless be placed in reserve by the use of one of the forms of the "environmental:specifier that incorporates the "primitive _RESERVE. 1a11a2

.3 Note that "local" and "outer" do not mean the same as "private" and "environmental." Local data may be either environmental depending on its allocation as controlled by the "environmental:specifier. Outer refers to a scope and therefore determines the legal range for data references while environmental refers to the allocation and permanence of the data. 1a11a3

.4 For recursive and reentrant procedures or programs, a separate copy of private data is made for each activation. Extra copies of environmental data are not made--unless, of course, that environment is itself called recursively or reentrantly. 1a11a4

.5 "Formal:input:parameters, "formal:output:parameters, and other local data must be environmental or reserved if they are to be used as communication to subsequent activations of a procedure. (An environmental "formal:input:parameter can be used for inter-invocation communication only if it is .B=1;not.B=0; a "parameter for at least one entrance.) 1a11a5

.6 A peculiar situation arises if an inner scope "procedure:name, "alternate:entrance:name or "statement:name has been made external and then invoked externally. The reserve data associated with this called program comes into existence when the called program is loaded. Private data is allocated when a scope is activated. Environmental (but not reserved) data is not allocated unless and until the scope to which it is private is activated, even though the inner scope is active. Branch to a "statement:name (external or not) never activates any scope, although it can deactivate the scope of the "go:to:statement. Referencing of outer scope data is permitted, of course, with the usual caveat that the result of referencing data which is not active, was not present, or has not been assigned a value since the program was loaded (or data was activated) is undefined. Although branching to an outer scope "statement:name is legal in all cases, the return address and other linkage data will not be valid if the outer scope has been exited since the last entrance. 1a11a6

.7 "Statement:names may be external. A branch to an external "statement:name does not activate a scope. The scope must already be active through other action. A branch to an external "statement:name is considered an exit from the program and all private data is deactivated. Similarly a branch to an outer scope "statement:name deactivates all data private to scopes inner with respect to the target scope. 1a11a7

.8 The "environmental:specifier and the "allocation:specifier must not appear in the same "declaraton. Controlled data are neither private nor reserved, although every pointed-to datum must be ultimately tied to a private or reserved pointer, perhaps through a long chain of pointed-to pointers. The permanence of controlled data depends on both the permanence of their pointers and the execution of relevant "statements. Certainly if a program or procedure obtains space from the system and allocates controlled data to that space, the controlled data exist so long as the program or procedure is active, the space is not relinquished, and all elements of the pointer exist. It is entirely the programmer's responsibility if he allocates other data to that same space. If the pointer ceases to exist, then so does what it points to (unless another pointer has taken up the burden). It is dependent on the system whether space is automatically

relinquished when the program or procedure that has obtained it goes inactive, 1a11a8

7.12 "Packing:Specifier 1a12

The "packing:specifier directs the compiler in the positioning of items or entries in a word. The information it provides is also used in the generation of code to access the items or entries in the most efficient manner, 1a12a

.1 _N indicates no packing, i.e., items do not share words. Item accessibility is enhanced at the cost of storage requirements, 1a12a1

.2 _D means dense packing such that items are immediately adjacent to each other but not overlapping. Only character items or items whose size is longer than a word may cross word boundaries. A byte of a character item must not cross a word boundary. The intention of dense packing is to pack items in such a way as to utilize the minimum number of words to contain them; however, it is recognized that optimum packing is a combinatorial process and may not be feasible for all implementations, 1a12a2

.3 _M specifies medium packing. Although this degree of packing is machine dependent, the intention is to provide an effective compromise between item accessibility and space utilization. An additional degree of specification is provided for medium packing by following the _M with a "number. This "number represents a machine-dependent field size or pattern to be used in arranging space for the item. For instance, M 2 might indicate on a particular compiler that the item is to be allocated to a half word, 1a12a3

.4 Use of the "packing:specifier in a "simple:item:declaration affects the positioning of the item in a word. In an "ordinary:table:item:declaration, the position of the item in an entry is affected. In the heading of an "ordinary:table:declaration, the "packing:specifier serves as default for "item:declarations that do not contain their own packing specification; it also determines the packing for entries of a tight structure table, 1a12a4

.5 For a specified table in which the programmer arranges the items within an entry or for a simple item which the programmer positions in a word, the

"packing:specifier does not direct the compiler in the packing of the items. It does, however, inform the compiler of any convenience provided in the specified packing. This information is used by the compiler as it generates code to reference the items. Normally, the "packing:specifier should be at least as dense as the actual packing of the item. If it is not, for example, if an item actually medium packed is designated as being unpacked, code may result that can erroneously disturb surrounding items. 1a12a5

7.13 "Constant:List 1a13

A "constant:list provides initial values for the items and entries of tables. Such provision of initial values is possible only if the table is declared to be in reserve and the extent of the table is known at compile time with the length of every dimension given in terms of "constants. 1a13a

.1 Each "constant:list consists of a list of signed or unsigned "constants separated by "commas. There may be extra "commas. Parts of the list may be enclosed in "parentheses, with a "count or "number preceding the "left:parenthesis. There may be "indices (enclosed in "brackets) among the elements of the list--but not within any of the "parentheses in the list. 1a13a1

.2 The parts of the list enclosed in "parentheses preceded by a "count are equivalent to that number of repetitions of the contents of the "parentheses, separated by "commas. For example: 1a13a1a

```
3 ( + 5,1 )  = + 5,1 , + 5,1 , + 5,1                    1a13a1a1

2 ( -4,05 , ) = -4,05 , , -4,05 ,                       1a13a1a2

 5 ( , )     = , , , , , , , , ,                        1a13a1a3

2 ( 'NO' , 3 ( 'YES' )) = 'NO' , 'YES' , 'YES' ,
'YES' ,                                                 1a13a1a4

'NO' , 'YES' , 'YES' , 'YES'                            1a13a1a5
```

For every _constant:list written in terms of this repetition notation, there exists, therefore, an equivalent, "parenthesis free, fully expanded "constant:list. The repetition notation is available as a shorthand for the user. The compiler algorithms which associate the "constant:list with the table to

be initialized are described in terms of the equivalent fully expanded "constant:list, 1a13a1b

.3 The mechanism which assigns "constants of the "Constant:list to entries of the table can be considered to comprise the following steps: 1a13a1b1

a. A pseudo-entry counter is set to the first entry, of the first row, of the first plane, etc, 1a13a1b1a

b. A pointer points to the beginning of the fully expanded "constant:list. If the list starts with a "comma, the pointer points to "null". If the list starts with a "constant, the pointer points to the (possibly signed) value of the "constant. Otherwise, the list starts with an "index and the pointer points to the "index, 1a13a1b1b

c. If the pointer points to null, the initial value of the entry (or of the item in the entry) indicated by the counter is undefined and this process proceeds to step f, 1a13a1b1c

d. If the pointer points to a value, this value becomes the initial value of the entry (or of the item in the entry) indicated by the counter and this process proceeds to step f, 1a13a1b1d

e. Otherwise, the pointer points to an "index. The counter is reset to indicate the entry corresponding to the "index. This process proceeds to step g, 1a13a1b1e

f. The counter is advanced to indicate the next entry. The next entry is the next element in the same row unless the row is completed, in which case it is the first element of the next row unless the plane is completed, in which case it is the first element of the row of the next plane unless the volume is completed, etc, 1a13a1b1f

g. The fully expanded "constant:list may be thought of as consisting of "constants and nulls separated by "commas and "indices. Between any two "commas there is either a "constant or a null. Between a "comma and a following "index there is either a "constant or a null. After

each "index there is either a "constant or a null. Before a "comma beginning the list or after a "comma ending the list there is a null. The pointer is now moved from the null, "constant, or "index it points to so that it then points to the next null, "constant, or "index, unless there are no more nulls, "constants, or "indices in the list, in which case this presetting process is completed. Otherwise, return to step c. 1a13a1b1g

.4 An "index fixes the location within the table from which initialization is to proceed. If the "constant:list does not begin with an "index, initialization will proceed from the first element of the first row of the first plane, etc. The "index is enclosed in "brackets and all components of the "index must be "constants. Also, the "index must be compatible with the dimensionality of the table. Leading and embedded null "index:components must have their place marked by "commas. Trailing null components will be assumed by the compiler. The value of null "index:components will be the "lower:bound of the corresponding dimension of the table. 1a13a1b2

.5 The order of values in a "constant:list is elements of a row, rows of a plane, planes of a volume, etc. This is reflected in the components of an "index starting from the left: 1a13a1b3

The _n-1 dimensional manifold within the _n dimensional table. 1a13a1b3a

the _n-2 dimensional manifold within that _n-1 dimensional manifold. 1a13a1b3b

. 1a13a1b3c

. 1a13a1b3d

. 1a13a1b3e

. 1a13a1b3f

The plane within that volume. 1a13a1b3g

The row within that plane. 1a13a1b3h

The particular entry within that row, 1a13a1b31

.6 The ability to include "indices within the "constant:list allows the user to specify that different parts of the "constant:list are to be applied to different parts of the table. In terms of the processing algorithms, the "indices supply new values for the pseudo-entry counter. The counter can be increased or decreased. There is a risk, however, that the "constant:list may be reapplied to entries already initialized. Whenever multiple initialization of the same bits is attempted the result is undefined, 1a13a1b4

.7 The following is an example of a "constant:list that attempts such an undefined initialization: 1a13a1b5

[2] 1, 2 [5] 3 , 4 , 5 [7] 6 1a13a1b5a

The pseudo-entry counter is initially set to _2 and advanced as the "constants _1 and _2 provide values for entries _2 and _3. The second "index then advances the counter to entry _5. Entry _4 is skipped and receives no initial value. "Constants _3, _4, and _5 provide values for entries _5, _6, and _7. The last "index sets the counter to entry _7 but entry _7 is already set. The result is therefore undefined, 1a13a1b6

.8 Different items occurring in the same word of a table may be preset, depending on the particular "indices and "constants appearing in their respective "constant:lists. If the bits of the items in the specific entries given preset values do not overlap, there is no conflict. In that case, the preset value for a given word is the union of the preset values given for its various bits. The initial values of any bits not specified within the word are undefined. If there is a conflict in that two "constant:lists each specify values for the same bits in a word, the initial values of those bits are undefined. If words or entries are completely skipped over in presetting entries of a table, the initial values of those bits are undefined. Words partially preset by two or more overlaid items ,B=1;not,B=0; in the same table, even though not interfering in terms of bits within the word, are undefined, 1a13a1b7

7,14 Data Structures 1a14

It is often necessary to specify more complex data structures than simple items. In JOVIAL, it is possible to form complex structures from more simple ones. 1a14a

,1 The simplest data "structure" is the bit string; all other structures are based on certain combinations of and interpretations of bit strings. The basic JOVIAL data structure is the item. An item is a bit string of specified length, with a specified interpretation (e.g., a portion of the bit string might be interpreted as significand, etc.). Normally, the internal structure of an item is ignored and the item is treated as a single entity. ("Intrinsic:functions such as _BIT, _BYTE, _INT and _FRAC are used to access subparts of an item when necessary.) 1a14a1

,2 The larger JOVIAL data structures are created from items. They are the entry, table, and data block. An entry consists of one or more (possibly overlaid) items. Each item (known as a table item) is separately named and separately accessed; the entry may also be accessed as a unit. Entries are not declared as separate entities, but are associated with tables. A table is a (multiply-) indexed list or array of entries, all having the same structure. Each table is separately declared and named; the "declaration also names the items in the entry and may specify their location, possibly overlaid, within the entry. A table is referenced by its "name; normally a table entry is referenced by the "tablename and the "index of the entry; an item within an entry is referenced by the "item:name and the "index of the entry. 1a14a2

,3 A simple item is simply an item that is not part of an entry. 1a14a3

,4 A data block is like an entry in that it consists of one or more (possibly overlaid) data structures. However, these data structures may be simple items, tables, or other data blocks. The data block cannot be indexed (placed in a table), and has its own "name. Data blocks may be accessed as a whole, but are permitted in only a very few operations; their primary use is for controlling data allocation. 1a14a4

7,15 "Item:Declarations 1a15

"Item:declarations name and describe both simple items and table items. 1a15a

.1 In the paragraphs to follow, first the "item:description is discussed as an element common to all "item:declarations. Then comes the "status:list:declaration which is referenced within "item:descriptions. Next, the "simple:item:declaration is specified. "Ordinary:table:item:declarations and "specified:table:item:declarations are then discussed together. Each will also be individually discussed in brief with their corresponding "table:declarations. 1a15a1

7.16 "Item:Description 1a16

The "item:description is used in an "item:declaration to give the type, size, and certain other information about the declared items. It may also be used in the heading of "table:declarations for similar purposes. 1a16a

.1 "Abbreviations in the "item:description give the basic type of the item (or items) as follows: 1a16a1

_C character 1a16a1a

_F floating 1a16a1b

_S signed (integer or fixed) 1a16a1c

_U unsigned (integer or fixed) 1a16a1d

.2 For character items the "size:specifier tells how many bytes in the item. If the "number is omitted, the default size is one byte. 1a16a2

.3 For floating items an _R says to round upon setting the value of the item if the "formula providing the value has extra bits in the significand--absence of the _R says do not round. The "significand:specifier, if present, gives a minimum size of the significand in bits--excluding the sign. The "exrad:specifier following the "comma, if present, gives a minimum size of exrad in bits--excluding the sign. The size of the exrad is based on the assumption that the radix is 2 regardless of the actual representation of floating numbers. If both of these "numbers are omitted, the default is the system-dependent single precision. If the system can provide alternative forms for floating values, it chooses one to accommodate the stated sizes of the significand

and exrad. If it cannot do this, the "declaration is in error. 1a16a3

.4 Signed and unsigned items can be integer or fixed. The _R, if present, says to round upon setting the value. If the _R is absent, do not round. The "size:specifier, if present, gives the size of the item in bits--excluding the sign for signed items. If this "number is omitted, the size is system-dependent--the size normally used by the compiler for addresses (at least for unsigned items). If the "precision:specifier is omitted, the item is an integer. If the "precision:specifier is present (even if its value is zero), the item is fixed and the "number (together with its sign, if present) tells how many fraction bits there are. This "number may be of any value. It may be positive and larger than the "number (or default) giving the size--in which case there are assumed or understood leading zeros between the binary point and the significant fraction bits. It may be negative--in which case there are trailing bits of unknown value between the significant integer bits and the binary point. 1a16a4

.5 Integer items (either signed or unsigned) may have "status:constants associated with some of their possible values by including either a "status:list or a "status:list:name in the "item:description. The values given to each "status:constant must be compatible with the other specifications in the "item:description. 1a16a5

7.17 "Status:List and "Status:List:Declaration 1a17

A "status:list lists "status:constants in such a way that each is assigned a unique value. The "status:list can appear directly in an "item:description or it can be declared in a "status:list:declaration for subsequent reference by "name in "item:descriptions occurring within the scope of the "status:list:declaration. 1a17a

.1 Within a "status:list, each "number (and the sign if present) provides a value to be the meaning of the first "status:constant following it. Sequential "status:constants then take on sequential values, unless a new "number (and optional sign) sets a new value for the next "constants. There may be gaps in the sequence of values, but the sequence must be absolutely increasing. The "number immediately following the "name may be omitted--giving a starting value of zero. 1a17a1

,2 The "status:list:declaration associates a "status:list:name with the "status:list. The "status:list can now be removed from the "item:declaration and only referenced therein by "name. This allows several items to be defined in terms of the same "status:list. In fact, all status values over an entire program can be collected within a single "status:list unless conflicting uses of the values of the "status:constant occur. 1a17a2

,3 The following is an example of a "status:list:declaration incorporating a "status:list with a gap in the sequence of values. 1a17a3

```
   STATUS QV [-2] V(Q),V(R),V(S),V(T) [4] V(U),V(V);      1a17a3a
```

From the above example, the values associated with the "status:constants are: 1a17a4

```
   V(Q) V(R) V(S) V(T) V(U) V(V)                          1a17a4a

    -2   -1    0    1    4    5                          1a17a4b
```

,4 A particular "status:constant may be in more than one list--and may therefore have more than one value, but it must not occur more than once in a given "status:list. More examples: 1a17a4c

```
      STATUS LIST V(A), V(B), V(C), V(D), V(E);           1a17a4c1

      STATUS BOOL V(ON), V(OFF);                          1a17a4c2

      STATUS RANGE [-6] V(POOR), V(FAIR);                 1a17a4c3

                        [37] V(GOOD) [91] V(EXCELLENT);   1a17a4c4

      STATUS SIZE V(SMALL), V(MED), V(LARGE);             1a17a4c5

      STATUS TEMP [1] V(HI), V(MED), V(LO);               1a17a4c6
```

Note that in the "status:lists for both _SIZE and _TEMP the "status:constant _V(MED) is declared. In one case, it has the value _1, and in the other, it has the value _2. The compiler will be able to resolve references either from context or by qualification. 1a17a5

"Simple:item:declarations name and describe those items not associated with tables. 1a18a

.1 Each "simple:item:declaration names one or more items. The "name or "names follow the "primitive _ITEM. If more than one "name is present they are separated from each other by "commas. A "simple:item:declaration declaring more than one "name is a shorthand method of declaring several items to have all stated attributes exactly the same. Its meaning is the same as several "simple:item:declarations all exactly alike except that each declares one different "name. 1a18a1

.2 Space is normally allocated to each item independently as reserved data. These storage criteria can be modified by the use of "independent:overlay:declarations and the optional "environmental:specifier or "allocation:specifier in the "simple:item:declaration. 1a18a2

.3 The "declaration may optionally contain indications that controlled or environmental allocation is to occur. Use of an "environmental:specifier naming an outer scope or the scope of the "declaration provides that storage for the item or items declared hereby shall be enviornmental or private, but not reserved (see Section 7.11.1). Use of an "allocation:specifier causes controlled allocation for the "named:variables. If a "numeric:variable or "numeric:formula follows the _@, it is taken as the implicit pointer to each item declared hereby--the same "pointer:formula is used for each item declared. If there is no "pointer:formula, every reference to each item requires an explicit "pointer:formula. 1a18a3

.4 The "item:description gives the type, size, and certain other information about the items declared in this "declaration. The same "item:description applies to each item declared hereby. 1a18a4

.5 Following the "item:description, there may be a "packing:specifier. The meanings of the "packing:specifier are as follows: 1a18a5

_N No packing 1a18a5a

_D Dense packing. 1a18a5b

_M "number Medium packing. There may be a

"number following the _M. The "number refers to a specific type or position within a type of medium packing. The detailed meanings of these "numbers are system dependent. 1a18a5c

.6 If the "simple:item:declaration contains no positioning information ("bit:number), the "packing:specifier tells the compiler if and how it may pack the item or items with other simple items. It defaults to no packing. If positioning information is present, the "packing specifier tells the compiler how it must access this item. In this case, the default is to dense packing. 1a18a6

.7 The programmer can control the positioning of simple items in a computer word by including positioning information in the "declaration. The "bit:number follows the "packing:specifier and tells in which bit of a word the item begins. The positioning of the item applies to all bits of the item, including sign bits, even if not counted in determining size. When coupled with the "independent:overlay:declaration an effective packing of simple items can be achieved. 1a18a7

.8 A single "constant may be included in the "declaration of simple items that are in reserve. If more than one item is declared all will initially receive the appropriate initial value. If the size of the item is defaulted in the "item:description and the initializing "constant requires a size larger than the default size, the size implied by the "constant becomes the size of the item. If the size of the "constant is larger than the stated size of the item, the "constant is truncated or rounded as if for assignment to the item. 1a18a8

.9 The "simple:item:declaration terminates with a "semiColon. 1a18a9

7.19 "Ordinary: and "Specified:Table:Item:Declarations 1a19

"Ordinary:table:item:declarations and "specified:table:item:declarations name and describe an item (or items) of ordinary and specified tables. The "declarations of the items occur within the bodies of the corresponding "table:declarations. 1a19a

.1 These "declarations name one or more items of the table corresponding to the containing "table:declaration. Each "declaration may include more than one "name as a

shorthand notation for declaring several items all of which have all stated attributes exactly the same. The meaning is exactly the same as several "declarations all alike except that each declares a different "name. This capability is likely to be more useful for "ordinary:table:item:declarations than for "specified:table:item:declarations. For the ordinary table, the items are individually declared hereby. They are then arranged within a table entry by the compiler. However, the "specified:table:item:declaration must include positioning information as an attribute and therefore multiple "names in a single "declaration serve only to give different "names to a single location (with but a single definition) within an entry. 1a19a1

.2 Space is allocated to the items as a part of the containing table. There is no provision within either the "ordinary:table:item:declaration or the "specified:table:item:declaration to affect the allocation method or permanence of the containing structure. If controlled allocation or unusual permanence is desired it must be accomplished within the "table:declaration or, if the table is contained in a data block, the specification occurs in the "data:block:declaration. 1a19a2

.3 The "item:description serves the same purpose as in the "simple:item:declaration. It gives the type, size, and certain other information about the declared items. The same "item:description applies to each item declared hereby. 1a19a3

.4 The "packing:specifier functions differently in the two "declarations being considered. In the "ordinary:table:item:declaration, the "packing:specifier directs the compiler as to how it should pack the item in an entry; either no packing, some degree of medium packing, or dense packing can be requested. If no "packing:specifier is included, the value found in the heading of the "ordinary:table:declaration (or the default for the table) will be used. In a "specified:table:item:declaration, the "packing:specifier tells the compiler how it must access this item. The default value is _D, dense packing. 1a19a4

.5 The "specified:table:item:declaration must include information to position the item within an entry. If the table is serial or parallel structured, the "bit:number gives the position of the first bit of the item in a word

of an entry, Numbering of bits starts with zero on the left and counts to the right, The "word:number tells the word of the entry (starting with zero) in which the item resides or begins, For tight structured tables only the "bit:number, the position of an item within an entry, is specified, The position information is enclosed in "brackets, 1a19a5

,6 Initial values may be provided for items by means of the "constant:list, There must be no attempt to preset (i,e,, no "constant:list) items that are not in reserve, The "constant:list provides initial values for the item in certain entries depending on the number of "constants and on the "indices in the "constant:list, If the size of the item is defaulted in the "item:description and any "constant in the "constant:list requires a size greater than the default size, the largest size implied by the "constant:list becomes the size of the item, If the size of any "constant in the "constant:list exceeds the size stated in the "item:description, the "constant is truncated or rounded as if for assignment to the item, 1a19a6

,7 The "item:declaration terminates with a "semicolon, 1a19a7

7,20 "Table:Declarations 1a20

Tables are collections of entries and the entries are themselves collections (possibly empty) of items, Tables have size and dimensionality; they are packed and they are structured, They may be statically allocated at compile time or they may be dynamically allocated at run time, The "table:declaration provides the means to describe these various traits of a table, 1a20a

,1 The paragraphs that follow first specify some elements and concepts common to both types of "declaration, Then each type of "table:declaration is discussed in turn, 1a20a1

7,21 "Allocation:Increment 1a21

An "allocation:increment is an optional part of "table:declarations, It specifies the unit of allocation for controlled allocation tables and the unit of usage for fixed allocation tables, It also affects the meaning of parallel structure, 1a21a

,1 For controlled allocation tables the "allocation:increment indicates the units in which space

is to be allocated to the table. The units can be either an entry, an entire table, or some submanifold. (A manifold is: a row, a plane, a volume, and so on. A submanifold is a manifold that is something less than the entire table.) 1a21a1

.2 The values for the "allocation:increment have the following meaning: 1a21a2

.B=1;Value.B=0;	.B=1;Unit of Increment.B=0;	
0	1 entry	1a21a2b
1	1 row	1a21a2c
2	1 plane	1a21a2d
3	1 volume	1a21a2e
4	1 hyper volume	1a21a2f
.		1a21a2g
.		1a21a2h
.		1a21a2i

.3 If not present, the default "allocation:increment means the entire table. A value of the "allocation:increment equal to the dimensionality of the table (e.g. _2 for a two dimension table) also means the entire table and can, therefore, be omitted. A value greater than the dimensionality of the table is undefined. 1a21a3

.4 If "allocation:increment is given for a table not pointed to, it gives the compiler information that may be useful (perhaps if storage is organized as pages) with regard to the programmer's intended usage. It also affects the meaning of parallel structure. 1a21a4

7.22 "Dimension:List 1a22

The "dimension:list of a "table:declaration provides both the dimensionality of the table and the extent (size or number of entries) of the table in each dimension. 1a22a

.1 The bounds are enclosed in "brackets. Absence of a "lower:bound (and a "colon) before an "upper:bound means

the "lower:bound has the implied value zero. Each "lower:bound present and each "upper:bound must be a "number or a "simple:integer:variable. If the table is to be statically allocated or is to be in an allocated data block, each "lower:bound present and each "upper:bound can only be a "number (and no "allocation:specifier can be present). 1a22a1

,2 "Lower: (explicit or implicit) and "upper:bounds are given for each dimension of the table. The number of "upper:bounds in the "declaration is the number of dimensions (_t) in the table. The "lower:bound (or the implied zero) and the "upper:bound in each position give the range of values for an "index:component in the corresponding position. In subsequent references to "variables of this table, the corresponding component of the "index must be within this range; an out-of-range "index:component is undefined. 1a22a2

,3 Under some circumstances, reference to "table:variables can employ suppressed "index:components. The value assumed for such missing "index:components is the "lower:bound of the corresponding dimension of the table. 1a22a3

,4 The extent of the table in a particular dimension is "upper:bound + 1 - "lower:bound. The extent of the entire table is the product of the extent of each dimension of the table. A 3 x 3 x 3 volume has a size of _27 entries. 1a22a4

,5 If the "allocation:increment of a controlled allocation table is less than the whole table, the pointer for the table is not a simple pointer but a pointer structure. The pointer structure must have dimensionality commensurate with the difference between the dimensionality of the table and its "allocation:increment. (A three dimensional table allocated by rows requires a two dimensional pointer structure.) In the "dimension:list for the table, the values for the leftmost bounds are derived from the "dimension:list of the pointer structure. The corresponding positions in the "dimension:list must be left blank and their positions must be held by "commas. (The "dimension:list of the three dimensional table from the example above might be [, , 1:10].) 1a22a5

7.23 Harmony of "Allocation:Increment, "Dimension:List, and "Allocation:Specifier 1a23

In controlled allocation tables, the "allocation:specifier allows for a "pointer:formula which will point to each unit of allocation as specified by the "allocation:increment. The dimensionality of a table is determined by the "dimension:list. It is essential that these elements of "declaration be in harmony. 1a23a

.1 The "allocation:increment must not indicate a manifold of greater dimensionality than that of the table as expressed in the "dimension:list. 1a23a1

.2 The dimensionality of the "pointer:formula within the "allocation:specifier must be complementary with those of the "allocation:increment and the table itself. If "allocation:increment indicates allocation of less than the entire table, "pointer:formula must be a "pointer:variable of dimension (_d) equal to the number of dimensions in the table (_t) minus the "allocation:increment (_a), or it must be a pointer function with _d input "parameters. For example, if a table had _t=4 dimensions, and the "allocation:increment (_a) were _2, a "pointer:variable of _d=2 dimensions would have elements each of which would point to a plane of the table, or a pointer function with two input "parameters would yield a value that points to a plane of the table. When referencing the table, the first two components of the "index would actually be indexing the "pointer:variable or be input to the pointer function and the last two would be indexing within the plane pointed to by the element of the "pointer:variable or the value returned by the pointer function. 1a23a2

.3 The dimensionality of the pointer structure as determined by the "allocation:specifier must be in balance with the dimensionality of the table as expressed in the "dimension:list. The "upper: and "lower:bounds of the "pointer:variable may be "constants or (if the pointer is itself pointed to) "variables. They become the "upper: and "lower:bounds for the entire set of allocated manifolds. The corresponding positions in the "dimension:list of the "table:declaration (the ones to the left within the "brackets) must be blank, their positions being held by the appropriate number of "commas. 1a23a3

.4 If the "allocation:increment indicates allocation of the entire table, a single pointer is associated with the table. The pointer is of _d=0 dimensions which is consistent with the requirements. 1a23a4

7.24 "Structure:Specifier 1a24

"Table:declarations, whether ordinary or specified, employ the "structure:specifier to determine the basic structure of the table. The basic structure of a table refers to the arrangement of table entries in the words of a computer. Tables may have either a serial, parallel, or tight structure. 1a24a

.1 In the absence of either the _P or _T the table structure is serial. Otherwise, a _P indicates parallel structure and a _T indicates tight structure. 1a24a1

.2 For serial and parallel tables there are one or more computer words for each entry. In a serial table, all the words for an entry are allocated contiguously in storage or memory. In a parallel table, the first words of all the entries in an "allocation:increment are stored contiguously, followed by the second words of all the entries of an "allocation:increment, etc. Figure 7-2 depicts a single one-dimensional table, _MN, with both a serial and parallel structure. 1a24a2

.3 For a table with a tight structure, there are one or more table entries per computer word. Therefore, in a tight table an entry can be no longer than one word. If the size of the entry permits, there may be more than one entry per word. Figure 7-3 depicts a table with a tight structure. 1a24a3

.4 In an ordinary table, the compiler is guided by the "packing:specifier in the table heading as it packs entries into a word of a tight structure table. For a tightly structured ordinary table, absence of a "packing:specifier in the heading causes the compiler to assume dense packing for the table. If "no packing" is specified, the table is not really tight. In a specified table an auxiliary set of "numbers is available to inform the compiler of the desired entry placement. 1a24a4

.5 The basic reason for having both parallel and serial structured tables is efficiency. Serial tables lend themselves to more efficient sequential operations, while parallel tables tend to allow more efficient indexed operations. It may be necessary, in some implementations, to establish prohibitions against situations which would run counter to the goals of efficiency. For example, implementation-specific requirements may force restrictions on the use of

multiple-word numeric items in parallel tables, If the system includes machine instructions which depend upon sequential storage locations for multiple-word operands, then, in the interest of efficient object code, it may be necessary to relocate these items in serially structured tables, 1a24a5

.6 For any table, a parallel structure applies only to the "allocation:increment, If the "allocation:increment is an entry, there is no difference between parallel and serial structure, 1a24a6

.7 For a controlled allocation table of tight structure it must be realized that an "allocation:increment of an entry has meaning only when there is a single entry in each word, With more than one entry per word, requests for space on the basis of the entry are meaningless, 1a24a7

7,25 "Ordinary:Table:Declaration 1a25

An "ordinary:table:declaration consists of an "ordinary:table:heading usually followed by an "ordinary:table:body, 1a25a

.1 An ordinary table is one for which the compiler determines the size and composition of an entry based upon the information in its "declaration, It is distinguished from the specified table in which the entry is completely described in the "declaration, 1a25a1

7,26 "Ordinary:Table:Heading 1a26

The "ordinary:table:heading contains information necessary to describe and name an ordinary table, 1a26a

.1 Every table must be named, The "name becomes a "table:name, 1a26a1

.2 If the table is to be independently, dynamically allocated, the "allocation:specifier must be present, If the permanence of the table is to be restricted to something less than in reserve, the "environmental:specifier must be present, If neither an "allocation:specifier nor an "environmental:specifier is present, the table has fixed allocation and is in reserve, If the table is to be collected within a data block, there must be no "allocation:specifier or "environmental:specifier in the "ordinary:table:declaration, The desired effect must be

achieved at the data block level via the "data:block:declaration. 1a26a2

.3 For controlled allocation tables, the "allocation:increment is a "number that gives the dimensionality of the subset of the table to be allocated dynamically. Omitting the "allocation:increment means the entire table is allocated at one time. If an "allocation:increment is given for a fixed allocation table it gives the compiler information that may be useful (perhaps if storage is organized as pages) with regard to the programmer's intended usage and it affects the meaning of parallel storage. 1a26a3

.4 The "dimension:list gives the number of dimensions of the table as well as the size of the table in each dimension. 1a26a4

.5 The "structure:specifier marks the table as being of serial, parallel, or tight structure. Serial is the default structure; parallel is obtained by scripting _P; and tight by _T. 1a26a5

.6 The "packing:specifier allows the table to be designated for dense, medium, or no packing. This occurrence of the "packing:specifier serves as a default value for "item:declarations of this table that do not contain their own and it also directs the packing of entries of a tight structure table. The overall default packing specification is "no packing" for an ordinary table of serial or parallel structure. If the structure is tight, the default packing is dense. A table really is not tight unless its entries are medium or densely packed. It is, of course, meaningless to try to pack the items of an entry more loosely than the entries themselves are actually packed. 1a26a6

.7 An "item:description can be given to apply to the "table:name. Its effect upon the size of the entry and its position within the entry are the same as if it were the description of a single item contained in the table. If the "item:description occurs in the "ordinary:table:heading, the "ordinary:table:heading is then the entire "ordinary:table:declaration and there is no "ordinary:table:body. If this "item:description is present, the "table:name may be used as a "table:variable. If this "item:description is missing, such a reference means a reference to an "entry:variable of type "bit". 1a26a7

.8 A "constant:list may be used to give initial values to some or all of the entries of a table in reserve. This "constant:list is independent of the occurrence of an "item:description in the "ordinary:table:heading. The programmer must avoid conflicts with "constant:lists in the "ordinary:table:body. 1a26a8

.9 The "ordinary:table:heading terminates with a "semicolon. 1a26a9

7.27 "Ordinary:Table:Body 1a27

The "ordinary:table:body follows the "ordinary:table:heading if that heading did not include an "item:description. The "ordinary:table:body lists those "item:declarations and "subordinate:overlay:declarations which make up an entry of the table. 1a27a

.1 The "ordinary:table:body may consist of only a "null:declaration. The "null:declaration indicates that no named items exist for the table. An entry size of one word is assigned the table. 1a27a1

.2 The "ordinary:table:body can be a single "ordinary:table:item:declaration. A complete discussion was presented in Section 7.19; the "ordinary:table:item:declaration as presented here is derived from that earlier, more complete discussion. 1a27a2

.3 More than one item can be named and declared by a single "declaration. The "item:description gives the type, size, and precision of the items. The "packing:specifier gives the packing of the item in an entry and if absent the default is to that specified in the "ordinary:table:heading. The "constant:list can provide initial values for the item if the table is in reserve. 1a27a3

.4 The "ordinary:table:body can alternatively be a compound "declaration comprising "ordinary:table:item:declarations and "subordinate:overlay:declarations enclosed in the brackets _BEGIN and _END and optionally terminated by a "semicolon. The items hereby declared are arranged by the compiler in some manner that it considers efficient. In packing the items to form an entry the compiler is constrained by the appropriate "packing:specifiers and "subordinate:overlay:declarations. 1a27a4

7.28 "Subordinate:Overlay:Declaration 1a28

An "ordinary:table:declaration may contain one or more "subordinate:overlay:declarations. The "subordinate:overlay:declaration provides for the sharing of space in an entry of the table by items of the table. 1a28a

.1 A "subordinate:overlay:declaration may appear only between the _BEGIN and _END "primitives that delimit an "ordinary:table:body. Only the "names of items declared in that same "ordinary:table:body may appear in the "subordinate:overlay:declaration and the "ordinary:table:item:declarations must precede the "subordinate:overlay:declaration. 1a28a1

.2 Items named as "subordinate:overlay:elements of a "subordinate:overlay:string are assigned nonoverlapping space within an entry of the table. The order of appearance of "item:names does not determine the order in which space is assigned the items. Item packing is performed in a manner the compiler considers efficient in accordance with relevant "packing:specifiers. 1a28a2

.3 While nonoverlapping space is assigned the items referenced in a "subordinate:overlay:string, the space controlled by the various "subordinate:overlay:strings of a "subordinate:overlay:expression does overlap. The amount of space controlled by the "subordinate:overlay:expression is determined so as to contain the most demanding "subordinate:overlay:string. The items referenced in the remaining "subordinate:overlay:strings share the determined space. Since "subordinate:overlay:expressions may be enclosed in "parentheses and treated as "subordinate:overlay:elements, item arrangements of varying complexity can be built up. 1a28a3

.4 The following example depicts a possible arrangement of items in response to a "subordinate:overlay:declaration with two "subordinate:overlay:strings, one of which has an embedded "subordinate:overlay:expression. 1a28a4

Items _BY and _BZ are assigned nonoverlapping space yet the space they occupy is shared with item _BX; the size of _BX is larger than the combined sizes of _BY and _BZ, so it, therefore, determines the amount of space controlled by the embedded "subordinate:overlay:expression. Items _BC and _BA are

each assigned space that does not overlap either that of the other or the common space assigned _BX, _BY and _BZ. Items _AA, _AB, and _AC are, in the same way, assigned nonoverlapping space. Finally, the space assigned the _A *items does overlap the space assigned the _B *items. The size of the space controlled by the "subordinate:overlay:string referencing the _B *items is the larger of the two so it determines the space controlled by the "subordinate:overlay:declaration. 1a28a5

.5 An "item:name must not appear in more than one "subordinate:overlay:string. If "subordinate:overlay:declarations contradict one another, the meaning is undefined. 1a28a6

7.29 "Specified:Table:Declaration 1a29

A "specified:table:declaration consists of a "specified:table:heading usually followed by a "specified:table:body. 1a29a

.1 As distinguished from the ordinary table for which the compiler determines the size and composition of an entry, a specified table is one for which the entry is completely described by the user. Entry specification is performed by the contents of both the "specified:table:heading and the "specified:table:body. 1a29a1

7.30 "Specified:Table:Heading 1a30

The "specified:table:heading contains information necessary to name a specified table and describe its entry makeup. 1a30a

.1 Every table must be named. The "name becomes a "table:name. 1a30a1

.2 The "environmental:specifier, "allocation:specifier, "allocation:increment, "dimension:list, "structure:specifier, and "constant:list all function in the same way they did in the "ordinary:table:heading. They determine the unit of allocation, permanence, dimensionality, extent in each dimension, structure, and initialization values of the table. They were reviewed in the discussion of the "ordinary:table:heading subsequent to being individually introduced and need no further explanation at this point. 1a30a2

.3 The size of an entry of a specified table and perhaps its disposition within a computer word is described by a

"number or series of "numbers. If the table is of parallel or serial structure, the "words:per:entry must be scripted as a "number to tell how many computer words are occupied by each entry. For a tightly structured table the entries are located within a computer word by one, two, or three "numbers. "Bits:per:entry gives the size of an entry in bits. The size must be no greater than the size of a computer word. "Bit:number gives the position within the word of the first bit of the first entry (in each word). It defaults to zero. "Entries:per:word tells how many entries go in each word. This must be compatible with the packing:specifier (which defaults to "dense"). If the third "number is absent, it defaults to the maximum possible in light of the other two "numbers and the "packing:specifier. 1a30a3

.4 The "packing:specifier may occur in two places within the "specified:table:heading. The first occurrence is connected with the disposition of an entry of a tightly structured table. The second is concerned with the "item:description which may optionally occur within the "specified:table:heading. The "packing:specifier does not serve as a default value for items of the table. The actual packing of the tightly structured entry is accomplished by the "bits:per:entry, "bit:number, and "entries:per:word values and is only described by the "packing:specifier. The "packing:specifier must be an accurate description of the location of the entry in a computer word judged in terms of any convenience the computer provides to access parts of a word; otherwise unauthorized use of surrounding bits may occur. The default value for the packing of tightly structured entries is dense. 1a30a4

.5 An item of the same "name as the table can be declared within the "specified:table:heading. An "item:description, optional "packing:specifier, and location information serve to declare the item. The various components declaring the item function in the same way as in the "specified:table:item:declaration (see Section 7.31.3). If an item is declared by the "specified:table:heading then the "table:declaration is judged complete and there can be no accompanying "specified:table:body. If the item is declared, then the "table:name can be used as a "table:variable; otherwise, such a reference means a reference to an "entry:variable. 1a30a5

.6 The "specified:table:heading terminates with a "semicolon. 1a30a6

7.31 "Specified:Table:Body 1a31

The "specified:table:body follows "specified:table:headings that do not declare items. It declares those items which make up an entry of the named specified table. 1a31a

.1 The "specified:table:body resembles the "ordinary:table:body. It can consist of a single "null:declaration. Also, the "specified:table:body may either be a single "specified:table:item:declaration or several of them collected as a compound "declaration by the "primitives _BEGIN and _END. "Specified:table:item:declarations include information to position the items within an entry; no "subordinate:overlay:declarations are allowed. Regardless of positioning information in the "specified:table:body, the entry size is taken from the declared size as specified in the "specified:table:heading. 1a31a1

.2 The "specified:table:item:declaration is discussed in brief below. The discussion is derived from an earlier more complete discussion of "item:declarations (see Section 7.19). 1a31a2

.3 More than one "item:name can be included yet all items named have exactly the same attributes and occupy the same position in an entry. The "item:description gives the type, size and precision of the item. The "packing:specifier tells how the item is to be accessed and defaults to dense. The "bit:number and "word:number locate the item within an entry of the table; for tight structured tables the "word:number must be suppressed. The "constant:list can provide initial values for the item if the table is in reserve. 1a31a3

.4 The positioning information for an item can be incompatible with the entry size as specified in the "specified:table:heading. To what extent this can have any meaning is system dependent. It is possible that for entries within one row of a serial table and within one allocation submanifold (and not too near the end of the row) reference to an item declared to be beyond the size of an entry may just utilize space in the following entry or entries in a straightforward manner. 1a31a4

7.32 "Data:Block:Declaration 1a32

A data block is a convenient structure for grouping and allocating simple items, tables and other data blocks, 1a32a

,1 The "environmental:specifier and "allocation:specifier function the same as in other "data:declarations. If neither is present, the data block is environmental to the program and in reserve. The "environmental:specifier can make the data block less permanent by declaring it to be private to some encompassing procedure or program. The "allocation:specifier can provide controlled allocation for the data block. In this case, space is only allocated to the data block dynamically in response to instructions within the object program and every reference to the data block or to the data declared within it requires resolution in terms of either an implicit or explicit pointer, 1a32a1

,2 Controlled allocation and restricted permanence cannot be attempted individually by elements of the data block. These specifications must be made collectively for the data block as a whole. Additionally, for tables declared within a data block, the "lower:bounds and "upper:bounds for all its dimensions must be constant, 1a32a2

,3 The items, tables and data blocks declared within this "data:block:declaration make up the associated data block. The order of occurrence of the "data:declarations does not imply the order of allocation for the associated structures. Unless otherwise directed by an "independent:overlay:declaration, the compiler is free to allocate space in a manner it considers convenient and efficient. In all cases, allocation of all elements of a data block is fixed and contiguous. Reference to an element within the data block always causes its relative position within the data block to be added to either the fixed location of the data block or to the value of the data block pointer, in the case of a controlled allocation data block, effective at the point of reference, 1a32a3

,4 "Independent:overlay:declarations can be used within the "data:block:declaration to arrange the named data structures of the data block. An "overlay:declaration within a "data:block:declaration is restricted to arranging elements declared within the data block; it cannot attempt to relate the data block to external members of the environment and is, therefore, prohibited

from mentioning any named data structures that are not declared within the "data:block:declaration. 1a32a4

7,33 "Independent:Overlay:Declaration 1a33

The "independent:overlay:declaration may be used to specify the relative allocation of data within a data block or in fixed allocation storage. It may also serve to specify the allocation of data to "absolute locations." The meaning of an "absolute location" is system dependent. It may be truly absolute, it may be a specific location in a program's private environment, or it may depend on the specific address whether it is private or truly absolute. 1a33a

,1 Just as there are data structures--data blocks, tables, and items--associated with "data:block:declarations, "table:declarations, and "item:declarations, so can there be thought to be overlay structures associated with the various components of "independent:overlay:declarations. When the data structures referenced by the "independent:overlay:declaration are materialized in storage they give rise to overlay structures known by similar words in English rather then in the italicized metalanguage. Thus, we have overlay element for "independent:overlay:element, overlay string for "independent:overlay:string, and overlay expression for "independent:overlay:expression. As will be seen, some overlay structures are also data structures and some have no counterpart data structure. Other overlay structures express a necessary relationship of data structures. 1a33a1

,2 An overlay element has a length that is measured in words. The length of an overlay element that is referenced as a "data:block:name, "table:name, or "simple:item:name is the length of the associated data structure. An overlay element that is a "spacer, on the other hand, serves to define an overlay structure that is pure length. The length associated with a "spacer is the number of words equal to the value of the "spacer. There is no data structure associated with a "spacer; the "number simply defines a length that is used to space other overlay structures in storage. An overlay element designated as an "independent:overlay:expression in "parentheses has the same length as the associated overlay expression. 1a33a2

,3 An "independent:overlay:string expresses a required relationship of the space allocated to the structures

associated with the "independent:overlay:elements it comprises. The overlay structure associated with each "independent:overlay:element following a "comma is allocated storage space following that allocated the overlay structure associated with the "independent:overlay:element preceding the same "comma. In the case of "independent:overlay:elements that are "spacers, there is no data structure to be allocated; however, space is left in the allocation process equal to the length expressed by the "spacer. Thus, an overlay string also has length and its length is the sum of the lengths of the overlay elements. 1a33a3

.4 An "independent:overlay:expression usually expresses the requirement that different data share storage space. The overlay structures associated with two or more "independent:overlay:strings connected with "colons are allocated storage space so that they begin with the same word. Thus, the length of an overlay expression is the length of the longest overlay string it contains. 1a33a4

.5 Note that the relationships in a complex "independent:overlay:expression do not require, in this partial example: 1a33a5

, T5), T15 , 1a33a5a

that _T15 immediately follows _T5. It may be that the overlay string associated with some "independent:overlay:string in the "independent:overlay:expression in "parentheses is longer than the one ending with _T5. _T15 immediately follows the longest overlay string associated with the "independent:overlay:expression in "parentheses. 1a33a6

.6 The absolute location of data referenced in "independent:overlay:declarations can be determined in several ways. Compool data occupy locations assigned outside the writing and compilation of the program. If a compool datum is mentioned in an "independent:overlay:declaration, it thus fixes the locations of all data mentioned in the same "independent:overlay:declaration. If the optional "number or "pattern:constant in "brackets is included in an "independent:overlay:declaration, it thus fixes (by whatever meaning the system ascribes to such a "number) the location of the first overlay element, and therefore of all overlay elements, of the "independent:overlay:expression. If any datum mentioned

in an "independent:overlay:declaration is also mentioned in an earlier "independent:overlay:declaration, the previously allocated location fixes the location of all overlay elements of the current "independent:overlay:expression. If none of these constraints are placed on the locations of the data, the compiler places the entire structure determined by the relationships of the "independent:overlay:declaration in contiguous locations it considers efficient and convenient. It does not consider, in assigning these locations, any data overlaid onto absolute locations. 1a33a7

.7 When the "independent:overlay:declaration contains the optional "number or "pattern:constant in "brackets or when the leftmost "independent:overlay:element is a reference to a compool datum or a datum mentioned in a previous "independent:overlay:declaration, then allocation proceeds in the most obvious way. With the location of the first overlay element fixed, there follows sequential allocation for overlay elements of each "independent:overlay:string while overlay strings themselves share a common origin. However, the reference point need not appear as the leftmost "independent:overlay:element; the "name of a compool datum or a datum referenced in another "independent:overlay:declaration may appear as an internal or even terminal "independent:overlay:element. In these cases, the basic allocation relation still holds but it may now be conceived of as allocating space for the overlay element on the left of the "comma to immediately precede that allocated to the overlay element on the right of the "comma. 1a33a8

.8 All "independent:overlay:declarations that have in common the mention of any particular data, together define a structure of interrelated data as if they had all been mentioned in one "independent:overlay:declaration. Consider, for example, the three "independent:overlay:declarations: 1a33a9

```
OVERLAY    AA,100,(BB : EE);                    1a33a9a

OVERLAY    CC,EE,DD;                            1a33a9b

OVERLAY    200,FF,GG : DD;                      1a33a9c
```

Although their entire structural relationship cannot be described in a single "independent:overlay:declaration, they form an interrelated and contiguous block of

storage--which is allocated accordingly. Figure 7-4 depicts the allocation of space to the related structure. In this diagram the origin for each of the "independent:overlay:expressions is represented by an asterisk. 1a33a10

.9 It is the responsibility of the programmer to avoid contradictions due to these allocation rules. A program is illegal if it requires the arrangement of data mentioned in an "independent:overlay:declaration to be allocated different locations, perhaps due to mention of two "names from a compool in a single "overlay:declaration or the same "name in two "overlay:declarations with absolute locations. It is illegal to attempt to relate any data declared to be independently pointed to, via "overlay:declarations, to other pointed-to data, to fixed allocation data, or to absolute locations. 1a33a11

.10 "Independent:overlay:declarations may be used to relate data blocks, tables and simple items to one another. The entire set of related structures (including any data blocks containing them), in effect, forms a data block. None of the related data structures can be declared to be pointed to. Within a pointed-to data block, however, "independent:overlay:declarations are permitted to influence the arrangement of the data declared therein. Any table mentioned in an "independent:overlay:declaration must have constant dimension bounds. 1a33a12

.11 An effective data block, just like a declared data block, has a certain degree of permanence. The permanence of a declared data block is expressed in the optional "environmental:specifier with a default to reserved. For an effective data block the permanence derives from the permanence of the contained data structures. All the structures of an effective data block must be at the same level of permanence; i.e., they must all be in reserve or all at the privacy level of a single procedure or program. If the effective data block created by one or more "independent:overlay:declarations is to be private to some scope, at least one of the overlay elements must be declared (by an appropriate "environmental:specifier) to be private to the selected scope. Other data elements may, but need not, be declared private to that same scope. None may be declared to be pointed to. None may have a contradictory

"environmental:specifier, "Contradictory" can best be explained by means of an example: 1a33a13

```
   PROC P4;                                                    1a33a13a
   _BEGIN                                                      1a33a13b
      ...                                                      1a33a13b1
      PROC P3 IN P4;                                           1a33a13b2
      _BEGIN                                                   1a33a13b3
         ITEM Q31, Q32 F;                                      1a33a13b3a
         OVERLAY Q31  ;  Q32;                                  1a33a13b3b
         ...                                                   1a33a13b3c
         PROC P2 IN P2;                                        1a33a13b3d
         _BEGIN                                                1a33a13b3e
            ITEM Q21 IN P3 F;                                  1a33a13b3e1
            ...                                                1a33a13b3e2
            PROC P1;                                           1a33a13b3e3
            _BEGIN                                             1a33a13b3e4
               ITEM  Q11 F;                                    1a33a13b3e4a
               OVERLAY Q11, Q21, Q31;                          1a33a13b3e4b
               ...                                             1a33a13b3e4c
            END                                                1a33a13b3e5
            ...                                                1a33a13b3e6
         END                                                   1a33a13b3f
         ...                                                   1a33a13b3g
      END                                                      1a33a13b4
      ...                                                      1a33a13b5
```

END 1a33a13c

;12 Procedure _P1 contains an "independent:overlay:declaration mentioning data local to _P1, _P2, and _P3, respectively. This serves to make _Q11 private to _P3, due to its relationship to _Q31. It would serve to make _Q21 also private to _P3 except that the "procedure:heading for _P2 makes all the otherwise defaulted data of _P2 private to _P2. To avoid the contradiction, _Q21 must be declared private to _P3. In _P3 there is another "independent:overlay:declaration relating _Q32 to _Q31. This adds _Q32 to the effective data block containing _Q31, _Q21, and _Q11. Now the "environmental:specifier in the "procedure:heading for _P3 makes this effective data block private to _P4, along with all the otherwise defaulted local data of _P3. This effective data block can only be at the privacy level of the unnamed data space of _P3 --it cannot be affected by an "environmental:specifier in a "data:declaration within _P3 --because the declaration for _Q21 specifically mentions _P3 in its "environmental:specifier. 1a33a14

,13 Each "independent:overlay:declaration must occur within a "program:declaration or "procedure:declaration such that all the data elements it mentions are local or outer, but never of an inner or disjoint scope. An "independent:overlay:declaration occurring in a "data:block:declaration is restricted to arranging data elements declared therein; it is prohibited from attempting to relate the data block to other data structures through the mention of data structures not in the data block. Conversely, an "independent:overlay:declaration cannot attempt to arrange the data structures collected in a "data:block:declaration unless it too is contained therein. It can, however, reference a data block or an element of a data block as a means of controlling the position of the data block relative to other structures of the environment. 1a33a15

,14 Any optimization done by the compiler will take cognizance of the relationships among the elements of program data established by "overlay:declarations. The relatedness of all data connected through relationships to "data:names mentioned in more than one "overlay:declaration is recognized, but no cognizance is taken of relatedness due to the following circumstances. 1a33a16

a. Interference is ,B=1;not,B=0; recognized if it

arises out of two data structures being independently overlaid on absolute locations. 1a33a16a

b. Interference of program data overlaid on absolute locations with system data, with data belonging to independent, concurrently operating programs, or with data the compiler allocates is ,B=1;not,B=0; recognized. 1a33a16b

Similarly, it is the programmer's responsibility, with regard to pointed-to data, to make sure that the pointer values do not cause interference. The compiler, of course, recognizes interference among overlaid data in a pointed-to data block or table. Example: 1a33a17

```
OVERLAY [ 0 ] ALPHA : 100, BETA;                    1a33a17a

OVERLAY [ 100 ] GAMMA : DELTA, EPSILON;             1a33a17b

OVERLAY ZETA, (ETA : EPSILON);                      1a33a17c
```

Interference of _ZETA with _DELTA and _GAMMA is recognized, but not so the interference with _BETA. Interference of _ALPHA with system data at zero is ,B=1;not,B=0; recognized. 1a33a18

7.34 "Define:Declaration 1a34

The "define:declaration provides the means to manipulate the source program at compile time. 1a34a

.1 The "define:name must not occur within the "definition, in such a manner as to cause recursive "definition:invocation (see Section 7.34.10). The following example shows the use of an unusual "define:declaration containing its own "name but not leading to recursive invocation: 1a34a1

```
ITEM FLIP U ;                                       1a34a1a

PROC TRIP ;                                         1a34a1b

BEGIN DEFINE FLIP " END FLIP " ;                    1a34a1c

      ALPHA = BETA ** GAMMA ;                       1a34a1d

      FLIP = DELTA / EPSILON ;                      1a34a1e
```

Subsequent references to the "define:name beyond the

terminating "semicolon, within the scope of the "define:declaration, are treated as invocations of the associated "definition. Invoking a "definition causes the compiler to substitute the "definition for the "define:name in the source program. Logically, "define:declarations are of two kinds: those with and those without "formal:define:parameters. After developing a few notions central to the concept of "define:declaration, the following sections specify first one and then the other. 1a34a2

.2 A "comment is not permitted between the "define:name and the "definition. Since the same "ideogram is used to delimit both a "comment and a "definition, the compiler will confuse one for the other and be unable to detect the error. 1a34a3

.3 "Definition means any string of JOVIAL "symbols. Two exceptions are necessary to make this rule complete. Use of the "quotation:mark and use of the "exclamation:point are restricted within a "definition. Restrictions are necessary because these "ideograms are a part of the syntax of the "definition. 1a34a4

.4 A single "quotation:mark is not permitted within a "definition. This is an "ideogram signaling the end of the "definition so its occurrence within the "definition is illegal. Therefore, a convention is necessary which allows the inclusion of "quotation:marks in a "definition. The convention is this: each "quotation:mark desired in the "definition should be scripted as two "quotation:marks. Upon "definition:invocation, the processing algorithms reduce all sequences of "quotation:marks. If, in a sequence of _n "quotation:marks, _n is an even number, then _n/2 "quotation:marks are copied from the "definition. If _n is an odd number, _(n-1)/2 "quotation:marks are copied from the "definition and the odd one is interpreted as the end of the "definition. An important application of the use of "quotation:marks arises in those cases where a "definition which must be bracketed by "quotation:marks (usually an instance of an "actual:define:parameter) occurs in a "definition. The additional "quotation:marks at each successive level of nesting permit the proper matching of "quotation:marks as delimiters. 1a34a5

.5 The use of a single "exclamation:point within a "definition is restricted to the identification of "formal:define:parameters (see Section 7.34.16).

However, in a way similar to that for "quotation:marks, a convention has been established to allow the use of "exclamation:points in a "definition. Each desired "exclamation:point that does not identify a "formal:define:parameter should be scripted as two. At invocation, strings of "exclamation:points are halved as they are copied. 1a34a6

,6 "Define:names have scope. The rules are, as much as possible, the same as for all other declared "names. A "define:declaration local to any scope is not effective in any outer scope. An outer scope "define:name is effective within an inner scope unless there is some explicit "declaration using that "name in the inner scope. For example, if _AA is a "define:name in an outer scope, then declaring _AA as a "name in an "item:declaration or a "name:declaration in an inner scope overrides the use of _AA as a "define:name in the inner scope; declaring _AA in a "define:declaration in the inner scope is a "definition of _AA effective for the inner scope only and having no effect on the "definition of _AA for the outer scope. 1a34a7

,7 There is no redefinition and removal of definitions. In no instance is a "name permitted to have more than one meaning within a particular scope. Neither can a first-level "procedure:name be the same as a main scope "define:name, since the "procedure:name has main scope. A main scope "item:name cannot be the same as a main scope "define:name. The recognition of a declarative "primitive renders an immediately following "name an error if that "name is the "name associated with any "declaration (including a "define:declaration) in the same scope. This does ,B=1;not,B=0; prevent the "definition:invocation of a "define:name which causes the introduction of any declarative "primitive. 1a34a8

,8 A main scope "statement:name cannot be the same as a main scope "define:name, but the attempt to name a "statement following the "define:declaration is not treated as an error. It is treated as an invocation of the "definition. The "name:declaration (see Section 7.35) is provided to allow an inner scope "statement:name to duplicate the spelling of an outer scope "define:name and still be treated as a "declaration of a new local "statement:name rather than as a "definition:invocation of an outer scope "definition. 1a34a9

,9 No "define:names are permitted in a

"procedure:declaration between the "primitive _PROC and the first terminating "semicolon. It is not always possible to differentiate between intended "definition:invocations and the "names of "formal:input:parameters and "formal:output:parameters of the "procedure:declarations. Therefore, to avoid such ambiguities it is necessary to make "define:names illegal in that range. 1a34a10

.10 The occurrence of a "definition:invocation causes the "definition to be substituted for the "definition:invocation in the source code of the "program:declaration. 1a34a11

.11 "Definition means the same here as it does in a "define:declaration. In most cases, the bracketing "quotation:marks are optional, but if the "definition contains one or more "commas or "right:parentheses they are mandatory. While it is generally permitted for "definition:invocations to supply punctuation for source "statements and "declarations, that freedom is not allowed within "definition:invocations themselves or in "actual:define:parameters. The "parentheses which delimit the "actual:define:parameters and the "commas which separate the "actual:define:parameters must not derive from "definition:invocations. For example, _DFNAM(ZZ,BB) may be a legal "definition:invocation but DEFINE COMMA ","; DFNAM(AA COMMA BB) is an illegal sequence. This does not, of course, restrict a "definition from containing imbedded "define:names but they are not recognized as "definition:invocations until the "definition is processed as source code. 1a34a12

.12 A "definition:invocation may refer to "definitions existing external to the current program. Either a compool, a library, or both may include "definitions that are to be associated with "names as if through the use of _DEFINE. All such external "define:names must be introduced to the program by the "compool:directive. 1a34a13

.13 A "define:name is not considered to be a "definition:invocation where it occurs as part of another "symbol such as a "comment, "character:constant, "status:constant or "name. A "definition:invocation may occur in another "definition even before its own "define:declaration; however, it is not recognized as a "definition:invocation except when the encompassing "definition is invoked and the substituted string of "symbols is examined as source code. Then the embedded

"definition:invocation invokes its own "definition. Of course, circular or recursive "definitions are not permitted. 1a34a14

.14 When a "definition:invocation occurs other than in a "definition, the associated "definitions are substituted. For a "define:name without "formal:define:parameters, there is only one "definition and it is simply substituted for the "name. If the "definition contains sequences of two or more "quotation:marks or two or more "exclamation:points in juxtaposition, one "mark is deleted from each pair of such "marks during substitution. The resulting string of "symbols may include "definition:invocations for which substitutions are made as this "definition is processed as source code. 1a34a15

.15 An example of "define:declarations without "formal:define:parameters and their use is: 1a34a16

```
DEFINE ALPHA " BETA + 2 " ;                    1a34a16a

DEFINE BETA " GAMMA + 3 " ;                    1a34a16b

DELTA = ALPHA + BETA ;                         1a34a16c
```

The above sequence of code has the same effect as: 1a34a17

```
DELTA = GAMMA + 3 + 2 + GAMMA + 3 ;            1a34a17a
```

.16 Within a list of "formal:define:parameters a "letter may occur only once. Consequently, a "define:declaration cannot have more than 26 "parameters. A correspondence is established between the "actual:define:parameters of a "definition:invocation and these "formal:define:parameters. Occurrence of the "letters as "parameters in the "definition must be preceded by an "exclamation:point. They then indicate where the corresponding "actual:define:parameters are to be inserted in the "definition. Examples: 1a34a18

```
DEFINE INC (X) "!X = !X + 1 ;";                1a34a18a

INC (ABC)                                      1a34a18b
```

is now equivalent to: 1a34a19

```
ABC = ABC + 1 ;                                1a34a19a
```

The "semicolon within the "definition is not required;

however, it affects the proper way to invoke the "define:name 1a34a20

```
   INC (PDQ) ;                                          1a34a20a
```

yields: 1a34a21

```
   PDQ = PDQ + 1 ; ;                                    1a34a21a
```

with an extra "semicolon, in this case standing for a "null:statement. 1a34a22

```
   DEFINE GOING (F,G,K) " FOR K(!F BY !G UNTIL K>!K); " ;
                                                        1a34a22a
```

Substitution would not be made for the _F in _FOR or for the first and second _K's because they are not preceded by "exclamation:points. 1a34a23

.17 Substitution of the "actual:define:parameter for the "formal:define:parameter neither removes nor adds significant "spaces that may lead or trail the "formal:define:parameter so that juxtaposition of other "signs in the "definition with the substituted "actual:define:parameter can create "symbols from the combination. Example: 1a34a24

```
   DEFINE TDEC (A,B,C,D)                                1a34a24a

   "TABLE !A [!B,!C,!D] P N;                            1a34a24b

     BEGIN !AF1 F ; F2!A F ; END " ;                    1a34a24c
```

The "definition:invocation 1a34a25

```
   TDEC (RX,5,ALPHA, "4,BETA") ;                        1a34a25a
```

would cause the resulting string 1a34a26

```
   TABLE RX [5,ALPHA,4,BETA] P N ;                      1a34a26a

     BEGIN RXF1 F ; F2RX F ; END                        1a34a26b
```

Leading or trailing "spaces with the "actual:define:parameter are omitted unless the "actual:define:parameter is delimited by "quotation:marks. In that case, all "signs within the delimiting "marks, including leading and trailing "spaces, form the string to be substituted for the "formal:define:parameter. 1a34a27

,18 A "definition:invocation must not be used to create "symbols by juxtaposing the "definition with the "symbols preceding or following the invoking "define:name. For example: 1a34a28

```
DEFINE STAR "*" ;                                    1a34a28a

A = B STAR* C ;                                      1a34a28b
```

cannot be used to generate 1a34a29

```
A = B ** C ;                                         1a34a29a
```

Note the distinction between this and the previous case, "Symbols can be created by proper positioning of "formal:define:parameters within a "definition but "symbols cannot be created by juxtaposition of a string of "characters substituted for a "definition:invocation within the immediate environment of the "definition:invocation. 1a34a30

,19 When a "define:declaration contains a list of "formal:define:parameters it is invoked by using a "definition:invocation which includes "actual:define:parameters. The "define:name is, of course, the matching key. The "definitions in the list of "actual:define:parameters are matched with the "letters in the list of "formal:define:parameters. There must not be more "actual:define:parameters than there are "formal:define:parameters. "Commas must be used as needed to indicate missing "actual:define:parameters where the temporary "definition of the corresponding "formal:define:parameter isto be a null string. In any case, if the "define:declaration is parameterized, any corresponding "definition:invocations must include the "parentheses. For example, in each of the "definition:invocations _DEF1() and _DEF3(,AA,BB) a null string is to be substituted for the first "formal:define:parameter, and in the invocation _DEF3(XX,YY,) a null string is to be substituted for the third "formal:define:parameter. Note that DEF1(" ") is necessary to cause a "space to be substituted for the "formal:define:parameter. The mechanism at "definition:invocation when there are "parameters can be considered as the following steps: 1a34a31

a. A copy is made of the "definition corresponding to the "define:name. Sequences of two or more "quotation:marks or two or more "exclamation:points

are halved during the copying. The processing of other "definition:invocations is postponed until step e. 1a34a31a

b. The "formal:define:parameters become temporary "define:names with temporary "definitions, corresponding by position, from the list of "actual:define:parameters. 1a34a31b

c. The temporary "definitions corresponding to the "formal:define:parameters replace the occurrences of those "formal:define:parameters in the copy. Sequences of two or more "quotation:marks or two or more "exclamation:points in the temporary "definitions are halved during the replacement. 1a34a31c

d. The modified copy replaces the "definition:invocation. 1a34a31d

e. The resulting string of "symbols is processed as source code. If it contains "definition:invocations, the substitutions are made at this time. The occurrence of "definition:invocations with "parameters leads back to step a. 1a34a31e

,20 The following example showing the use of a "define:declaration with "formal:define:parameters makes use of previous examples given in Section 7.34.15. Consider the following additional "define:declarations: 1a34a32

```
DEFINE ASSIGN "=" ;                                          1a34a32a

DEFINE MAX (A, B, C)                                         1a34a32b

  "!C ASSIGN !A ; IF !C < !B ; !C ASSIGN !B ; " ;            1a34a32c
```

and a "definition:invocation: 1a34a33

```
MAX ( ALPHA , "ARCTAN(AA,BB)" , F1 ) ;                       1a34a33a
```

Recognition of the "definition:invocation of MAX results in the following processing. First, according to step 7.34.19a, a copy of the "definition is made. 1a34a34

```
!C ASSIGN !A ;                                               1a34a34a

IF !C < !B ;                                                 1a34a34b

!C ASSIGN !B ;                                               1a34a34c
```

By step b, the following associations are made: 1a34a35

_A is associated with _ALPHA 1a34a35a

_B is associated with _ARCTAN(AA,BB) 1a34a35b

_C is associated with _F1 1a34a35c

By step c, replacement is made in the copy and by d, the copy replaces the "definition:invocation in the source stream. 1a34a36

```
F1 ASSIGN ALPHA ;                                   1a34a36a
IF F1 < ARCTAN(AA,BB) ;                             1a34a36b
F1 ASSIGN ARCTAN(AA,BB) ;                           1a34a36c
```

Following the substitution, the "definition is processed as source, by step e, with the remaining "definition:invocations recognized and resolved as follows: 1a34a37

```
F1 = GAMMA + 3 + 2 ;                                1a34a37a
IF F1 < ARCTAN(AA,BB) ;                             1a34a37b
F1 = ARCTAN(AA,BB) ;                                1a34a37c
```

.21 The following example illustrates the use of sequences of two or more "quotation:marks as delimiters of "definitions: 1a34a38

```
DEFINE FUNC(A, B) "(!A + !B)/2" ;                   1a34a38a
DEFINE ARCFUNC "FUNC(A1, ""ARCTAN(A2,A3)"")" ;      1a34a38b
XX=ARCFUNC ;                                        1a34a38c
```

This must be done in two stages and yields first: 1a34a39

```
XX=FUNC(A1, "ARCTAN(A2,A3)") ;                      1a34a39a
```

and finally: 1a34a40

```
XX=(A1 + ARCTAN(A2,A3))/2 ;                         1a34a40a
```

.22 Multiple, bracketing "quotation:marks are also required if a "define:declaration occurs in the

"definition of another "define:declaration. In such a case, the including "define:declaration may be invoked only once in any scope (unless the included "define:name is modified by an "actual:define:parameter). More than one such use would be an attempt to redeclare the included "define:name. 1a34a41

7.35 "Name:Declaration 1a35

The "name:declaration is a scoping mechanism. It provides the means of clarifying the intended scope for "procedure:names and "statement:names. 1a35a

.1 The "name:declaration can occur in any scope. In the main scope it occurs as a part of an "external:declaration (see Chapter 9). In this usage, it designates "statement:names that are either referenced or declared in the current program as being of external scope. As a part of the "external:declaration this "declaration is preceded by either of the "primitives _REF or _DEF. 1a35a1

.2 In an inner scope, the "name:declaration resolves any possible conflicts between local "statement: and "procedure:names and outer scope "define:names. Since the "declaration of local "statement:names and the invocation of local procedures are not introduced by defining "primitives, it is necessary to be able to make a distinction between this use of a "name and the invocation of an outer-scope "definition with a similarily spelled "define:name. To avoid such conflicts, "procedure:names for procedures that are referenced before they are declared and "statement:names can be listed in a "name:declaration. 1a35a2

NJN 28-MAY-74 13:12 30821

A Vote for the Filter Restructuring election

I don't know if my vote counts since my opportunities to use NLS will be severely restricted inn the future, but I vote for method 2 in the Filter Restructure election. I am sending this to you because I suspect that it may not get through the OFFICE-1 feedback mechanism. 1

bridge

Significant contributions to USAF operation systems or capabilities.

Information Processing

RADC/IS, over the past ten years, has been conducting an exploratory and an advanced development program in computer software which has been aimed at improving the performance and reducing the cost of the data processing portion of Air Force operational systems. The following represents some of the significant contributions that have been generated by RADC/IS,

bridge

DM-1

The design and implementation of the DATA Management System DM-1 (Data Manager) was completed. The DM-1 system is extremely flexible in design, is programmed in a higher order language (JOVIAL) and its overall design contains the best features of many other systems. From DM-1 will come many of the items that will aid in the future Data Management System acquisition, such as complete separation of logical and physical files and a library service which will allow application program generation from a pool of common processes. The Reliability Analysis Information System (RACIS) is the principle DOD activity for one collection, storage, analysis and dissemination of information and experience data bearing on the reliability of microelectronics. This system, which runs on the Honeywell 600/6000 computers has been implemented for the Reliability Analysis Information System (RACIS). A subset of RACIS applications are currently being processed through the DM-1 software.

bridge

MULTICS

RADC was the first military environment to implement the ARPA sponsored Multics operating system. This beyond-the-state-of-the-Art software, which represents a potentially powerful interactive computer environment, was tested and evaluated in the RADC computer facility. The results of this in house evaluation are recorded in RADC TR-71-121, "RADC/MULTICS EVALUATION." The work emphasized the advantages MULTICS provides over conventional operating systems in the areas of data management, security and protection, resource sharing and manipulation of large files on the order of 30 million words. This RADC study had a major impact in providing a solution to an urgent requirement for a powerful interactive data processing system to support the Assistant Secretary of Defense (System Analysis). RADC personnel gave technical assistance to the Air Force Data Services Center who had prime responsibility to satisfy this requirement. The solution was provided by the installation of a complete MULTICS system at the Air Force Data Service Center, Pentagon. RADC, in its continuing study of MULTICS completed the design and implementation of a set of tools to be used to construct data management functions in the MULTICS environment. The tools provide a development framework in the DMS functional areas of

storage management, process control, access control and related maintenance operations. One of the products of this work is the GCOS/MULTICS File Transfer Facility to be used at the Air Force Data Services Center to convert their GCOS computing load to the newly implemented MULTICS system. The tools are currently being used in the design of a secure data management within a MULTICS environment.

bridge

WWMCCS TESTING

The Joint Tecnical Support Activity requested RADC to pursue research studies to determine adequate testing procedures for testing future releases of WWMCCS and associated software packages. Three studies were completed and methodologies were provided for: imbedding software monitors for data collection, stating user requirements consistent with DMS technology, testing security and restart/recovery features of a DMS. These efforts are impacting the community of WWMCCS users by allowing users to collect data down to the level of the operating system, to analyze the DMS functional attributes against their operational requirements and to assist in evaluating the security of their operational environment. The experience gained by RADC personnel through these and other efforts had a major impact on the HQ AF directive that RADC be designated the technical laboratory for providing technical support to AF WWMCCS sites.

TRANSACTION PROCESSOR

An investigation was completed which integrated the transaction processor (a newly provided executive service of GCOS) to the DM-1 data management system to provide a multi-user, priority driven capability for access to the DM-1. The significance of this experiment is that it established the fact that a large computer program executing in a GCOS environment could be made adaptable to varying user requirements. It also provided a basic framework in which the real-time processing requirements of several Air Force operational systems could be studied, in order to provide practical solutions. This work has become part of the technical strategy being employed by RADC for the Air Force Data Design Center in their responsibility for providing standard computer packages for the Air Force WWMCCS sites.

bridge

FEMIS

The RADC developed FEMIS system was implemented to provide an on-line capability for the Air Force Communication Services (AFCS) management information system called EIMS (Engineering Installation Management System). The EIM system was previously operated in a computer batch environment which could only provide bi-weekly standard formatted printouts. Basically, while the data was available in the computer system, management was unable to extract it on an as-required basis. Information necessary to answer a specific question had to be weeded out of an array of from one to some 85 formatted products depending on the particular data desired. The RADC on-line system accepted all of the EIMS data and provided the capability to extract specific information as needed. The system was service tested with NCA's engineering and installation/maintenance program (which includes the Northern United States, European, and Alaskan Engineering) to assure the systems capability, flexibility and responsiveness to satisfy management requirements. The system worked so well that the entire AFCS EMIS master file was fed into the computer. All AFCS sub-commands have a direct or indirect access to the EIMS Retrieval System. Commands in the continental United States have directed lines to the computer. Pacific and European

bridge

Communication Areas must go through AFCS to obtain retrieval information.

bridge

EXTENDED INSTRUCTION SET

An in-house study directed by HQ USAF (ACD) was completed to determine the capabilities of Honeywell's Extended Instruction Set (EIS) which was being proposed as an addition to the WWMCCS computers, The study which resulted in two reports "EIS and Hardware Compatability Study", and "Effectiveness of WWMCCS Operating System on H/600" had a major impact on the final decision by the HQ USAF and the Joint Technical Support Activity to upgrade all Air Force WWMCCS computers and all other WWMCCS computers to processors containing the Extended Instruction Set, The results provided by RADC indicated that; (1) both execution time and number of required instructions for Cobol programs decreased approximately 20% using EIS, (2) EIS would have potential benefit for other high level languages programs (FORTRAN and JOVIAL), and (3) the batch version of the WWMCCS operating system which normally executes on the H/6000 computer could also be executed on the H/600 computer,

bridge

TIPI

The Tactical Information Processing and Interpretation System (TIPI) is aimed at providing a mobile environment for the gathering, processing, and dissemination of tactical intelligence. The effort is jointly sponsored by the USAF and the US Marine Corps. Personnel from the Information Processing Branch (RADC) provided support to all major components of the TIPI system. These included the Operating System, JOVIAL Language Compiler, and Data Management System. The extent of this effort encompassed all phases of the program development from the evaluation of bidder's proposals through the design and implementation of system software.

bridge

SEMANOL

The SEMANOL (SEMANtics Oriented Language) System and a SEMANOL description of the JOVIAL/J3 programming language was delivered to RADC under Contract F30602-72-C-0047. The SEMANOL System consists of a meta-language called SEMANOL, which is capable of describing both the complete syntax and semantics of a programming language, and an interpreter which is capable of processing that description to produce a processor for the language described. Heretofore, no meta-language existed which could unambiguously and completely specify the total syntax and semantics of a programming language and yet be readable by laymen with minimal instruction. In the past, language specifications were written in natural languages, which are very ambiguous, and the specifications were open to personal interpretation. This led to drastic differences in any one language's compilers. The language processor produced by the SEMANOL System, although too inefficient for production use, can be used to test the SEMANOL description of the language, serve as an interpretation standard which compilers must match, and check out the impact of proposed changes to the language. SEMANOL was used to described JOVIAL/J3 as literally specified in AFM 100-24, and proved its worth by easily pointing out several ambiguities and incompletely specified portions, some

of which were known and some not previously brought out, It is hoped that SEMANOL will become widely used in the specification of programming languages so that language implementations will be more consistent and transferability of software increased.

bridge

CMS-2

The USN Fleet Combat Direction Systems Support Activity (FCDSSA) San Diego, California, is pursuing a definition and test of CMS-2, a JOVIAL derivative language, via the SEMANOL meta-language developed for RADC by TRW Systems Group, Redondo Beach, California. The definition of CMS-2 is being handled under a separate Naval procurement from the current RADC SEMANOL/J73 effort. From RADC's viewpoint, the Naval work is an excellent testbed for SEMANOL; it can help flag potential problem areas in the SEMANOL language or the H-635 hosted interpreter not currently recognized as lacking in universality among computer programs.

JOVIAL-J-73

AFM 100-24, the Air Force language specification for JOVIAL/J3, was published in 1967 and described a dialect of JOVIAL, called J3, which first appeared in early 1963. By mid 1969, it was officially recognized that J3, which was designed for "second generation" computer hardware, had inherent problems which made it both impossible to completely implement on a modern computer system and unable to take advantage of the power of new hardware. In addition, J3 lacked certain capabilities required by the Air Force Command and Control community. In July 1969, RADC was given the job of correcting these deficiencies for JOVIAL as specified in AFM 100-24. A number of meetings were held and attended by JOVIAL users and language experts in the hope that AFM 100-24 could be updated to describe a version of JOVIAL similar to J3 which would correct the deficiencies of J3. However, when an attempt to incorporate the desired changes was made by a subcommittee of language and compiler experts, it became immediately evident that to do so would result in a language that had a grotesque structure and would be difficult and impractical to implement. It was decided that rather than produce such a product (a "hodge-podge" of old JOVIAL and new concepts), it would be better to make some basic changes to JOVIAL to accommodate the new language structures,

This was accomplished with the new JOVIAL J73 language, JOVIAL-J73 has several features which are not in J3 as defined in AFM 100-24, These features, or the capabilities which they add to the language, were requested by personnel on Government projects, both presently using JOVIAL or planning to use JOVIAL, The following are the major capabilities which JOVIAL/J73 has over JOVIAL/J3: (a) Conditional and loop statements which allow better "structured" programs, (b) Reentrancy and recursion, (c) Attribute guidance (d) Formatting (e) Pointers and programmer core allocation control, (f) Alternate entrances to procedures, (g) Nested procedures, (h) Macro-like DEFINE statements, (i) External data, programs, and procedures, (j) Compiler directives, (K) Multi-dimensional tables, Since it is intuitively felt by several prominent computer scientists that structured programs are less error prone, and these feelings have been partially backed up with empirical results, the loop and conditional statements of JOVIAL/J3 were modified to permit structured programming, Reentrancy and recursion were added to make JOVIAL better suited to building operating systems, compilers, and data management systems, and taking advantage of multi-processing hardware, Attribute guidance enables the programmer to "override" the standard arithmetic scaling rules

of the compiler without having to resort to a machine oriented language. The lack of this capability was a popular complaint against JOVIAL/J3. Formatting enables the programmer to automatically convert from character data to numeric representations which can be arithmetically operated on or vice-versa. In addition, it allows the programmer to write or read an I/O buffer in any format.

Pointers and programmer core allocation control allow the programmer himself to "create" additional space in the computer only when he needs it, and "return" it for other use when he does not, so that optimal use may be made of core resources of the computer system. Alternate entrances to procedures permit the programmer to design procedures which have multiple entrance points so that the same piece of code can accomplish distinct yet related tasks. Nested procedures allow each subprogram or subroutine to be treated the same as a main program and thus be developed independently of their subprograms much more easily. Macro-like DEFINE statements give the programmer the convenience of assembly language macros in a higher order language and make programs more readable. External data, programs, and procedures enable the programmer to build, compile, debug, and maintain his programs in smallerpieces which are easier to comprehend. Compiler

directives provide supplemental information to a compiler about the program being compiled. This information allows the compiler to produce better object code, simplify its job, provide better debugging aids, etc. Multi-dimensional tables make for arrays of highly structured data, and in this case, simplified the language by allowing the data type ARRAY to be eliminated at no loss in capability. JOVIAL/J73 fills in some of the programming application gaps of the old standard USAF programming languages; namely JOVIAL/J3, FORTRAN, and COBOL. It is generally felt that J73 will be adopted by several non-Command and Control related agencies. For instance, the FAA has expressed a serious interest in developing programs written in J73 to automate the functions of air traffic control.

VALIDATION OF THE WWMCCS JOVIAL COMPILER

In March 1972 RADC (ISIS) received a request from CSAF to conduct the WWMCCS JOVIAL Acceptance Tests for the Joint Technical Support Activity (JTSA) at Honeywell Information Systems (HIS), Phoenix, Arizona. To accomplish this task, the following steps were taken by RADC: (a) A Joint Acceptance Test Team was organized consisting of representatives from RADC, SAC, NORAD, & ADC; (b) A test plan/procedures were written and distributed to JTSA and all test team representatives; (c) The JOVIAL Acceptance Tests were conducted by RADC, with the assistance of team representatives and a preliminary report of test results as prepared and distributed; (d) A final Test Report was then prepared and distributed which summarized all test results and included recommendations on acceptance or rejection of the HIS JOVIAL compiler.

All test materials-documentation were furnished to the JTSA and plans were made to assist JTSA personnel in conducting their own Acceptance Tests. To insure that the WWMCCS JOVIAL J3 compiler (version 50WW) met the requirements, as amended by the WWMCCS contract, several classes of tests were developed which include: (1) language features (JCVS); (2) error detection and reporting; (3) capacity; (4) efficiency; (5)

machine/implementation dependent features; and, (6) generalized/other tests. Validation of items in class 1 was accomplished through use of the JOVIAL Compiler Validaton System (JCVS) as debugged and maintained by RADC. JCVS objectives were: (1) to establish that each language feature in AFM 100-24 is accepted by the compiler; and (2) to establish that execution of the resulting source code produces the correct result(s). Proof of these objectives were indeed accomplished was obtained by running the JCVS against JOVIAL compilers on virtually all major computer systems, including earlier versions of the HIS 600 series Compiler. Tests in classes 2,3 were procured primarily from NORAD with test variations/alternatives from RADC. Tests in class 4 were representative SAC programs used for determining efficiency or code expansion ratio. Class 5 tests were written by RADC and class 6 by ADC. The complete package of tests has been documented and copied in a variety of formats to insure ease of use on other computer systems. The growth of capabiliy at RADC in JOVIAL compiler validation is particularly significant since JOVIAL is playing an ever increasing role in computer languages. The WWMCCS tests are a milestone in that these tests represent the most thorough analysis of this type ever conducted on a compiler procured by a Government agency. Such

documentation and test materials should serve as a valuable compiler validation tool in future procurements.

bridge

(J30822) 29-MAY-74 06:20; Title: Author(s): Edward F. LaForge/ELF;
Distribution: /FJT JLM RFI ELF; Sub-Collections: RADC; Clerk: ELF;
Origin: <LAFORGE>BRIDGE.NLS;1, 28-MAY-74 13:06 ELF ;

RJC 29-MAY-74 07:51 30823

Kenyon Memo to DCLP

oEd, am journaling this particular item to y then deleting file from my directoryb

Kenyon Memo to DCLP

ISIM (E, J, Kennedy/X3857)

Requested Engineering Support From RADC For AFBITS

RADC (ISI)

(IS)

(DCLP)

IN TURN

1, The REQUESTED ENGINEERING SUPPORT FROM RADC FOR AFBITS and the PROPOSED ISI TASKS have been compared at your request and the following comments are provided for your information:

a, There appears to be a very close correspondence between the Task on Information System science and the proposed ISI Task USER ACCEPTANCE OF ON-LINE TERMINALS*, However, Sub-Task 2 does not appear to be particularly suitable to our area of expertise, For example, to make a definitive assessment of the relative prevalence of forms, to determine if more than one set of text preparation facility is needed and then to detail two or more sets, and finally to correlate these desired capabilities with existing equipment is a very labor intensive effort, We do not have the manpower to pursue labor intensive efforts that are not directly relevant to our own interests, The alternatives are these; we can collect such data on a small group of users at RADC in the course of our normal activities, or we can make more efficient use of our manpower and manage a contract to meet the larger objectives of this subtask, It is expected that if the second choice is made that funds for a contractual effort would be made available to IS, A good estimate is that $50K would be required,

(1) Task - Information System Science

(a) The purpose of this task is to determine the preferred classes of equipment for typical administrative applications and to develop concepts, guidelines and criteria for indoctrinating all levels of Air Force personnel in the use of that equipment,

Emphasis must be placed on applications used by non-technical personnel.

(b) Sub Task 1: Determine the effects of forced use of video terminal equipment by non-typists for preparation of letters, messages and forms. Investigate use of formatting aids and protected fields on such preparation. The product of Sub Task 1 should be a report detailing problems, solutions, learning curves, levels of performance attained and, if possible, personality correlations with performance capabilities. The basic question to be answered is: Can typical Air Force personnel perform these tasks adequately to conduct daily business using equipment?

(c) Sub Task 2: Determine the optimum mix of text preparation capabilities needed for day-to-day Air Force operations. This should include a definitive assessment of the relative prevalence of forms, letters, messages and documents exceeding ten pages. If more than one set of text preparation capabilities is required, detail both sets. If possible, correlate these desired capabilities with existing stand-alone, shared logic, and computer aided equipment and software. RADC role is formal RT&E support.

(2) USER ACCEPTANCE OF ON-LINE TERMINALS*

(a) Study and evaluation of the degree of user acceptance of the use of on-line interactive terminals in the performance of daily work. The degree of user acceptance is to be measured in terms of the ability to do the work using an on-line terminal, the time spent in doing it compared to more conventional methods, and the amount of computer time used during predetermined and regular intervals. In this evaluation, consideration will be paid to the type of personnel, their level, ages and other variables that might be of interest to various efforts on-going in the Air Force such as the BCM, AFBITS, SADPR-85 and project ADMIN.

(b) Estimated cost = .2 MY $10K

b. The Automation of Administrative Procedures and Correspondence Transmission task is directly in line with two of the efforts that have already been proposed by ISI and a third

task which has not yet been specifically proposed, The two proposed efforts are: 1, FORMS SYSTEM* 2, COST/EFFECTIVENESS*, The third effort, PERSONNEL SUPPORT ORGANIZATION (PSO), which has not been previously discussed, covers work that is being done in ISI and it too is directly in line with the task proposed by ESD, We feel that these efforts will satisfy the needs of the AFBITS and related programs,

(1) Task - Automation of Administrative Procedures and Correspondence Transmission

(a) This task should demonstrate the feasibility of automating typical administrative procedures or parts of procedures, Products should include: (1) benefits achieved in terms of manhour savings, (2) an assessment of which procedural steps are amenable to automation and which are not, and why, (3) an exposition of regulatory barriers and hierarchical impediments, (4) a demonstration that adequate administrative controls and audit trails can exist in such a system, and (5) an assessment of operator and user response to automated procedures, RADC role is formal RI&E support,

(2) COST/EFFECTIVENESS*

(a) Development and measurement of methodologies and procedures for doing meaningful work in the Air Force environment using on-line interactive terminals and text-processing systems, Special emphasis will be placed on the use of this technology in the context of the problems unearthed in the BCM, the SADPR-85, and the AFBITS systems, and in anticipating the problems that will be encountered, This will be done in the laboratory using existing capability, rather than by the expenditure of capital funds for equipment and the use of scarce personnel and training resources at a later time,

(b) Estimated cost = ,3 MY $20K

(3) FORMS SYSTEM*

(a) Development of a system to take care of the great bulk of the routine, but important work performed in the Air Force, The work that is, and has been, performed over the years by the use of forms, Special

emphasis will be placed on capturing the data at the keystroke, and having done this, to devise the proper ways of using it to avoid having to copy, extract, merge, distribute, sort, file, find (unfile), modify by redoing, and all the other onerous things that should be done by text processing systems rather than by hand. These vital functions depend on making the tools, the terminal and the computer, available at all levels and particularly to workers. The data base development depends primarily on the worker, not the supervisor.

(b) Estimated cost = .3 MY $70K

(4) PERSONNEL SERVICE ORGANIZATION (PSO)

(a) Develop a Personnel Service Organization (PSO) that will meet the unique needs of the RADC, as a prototype of AF PSO's. The organization will provide the services normally provided by secretarial, and administrative personnel, and will also provide unique services such as inputting textual material to a text-processing system, editing, and outputting textual material. Additionally, it will capture data to fill in and print forms, strip data from forms and routine reports in order to provide inputs for a data base for later use. In addition, routine reports will be prepared, using textual material already collected and available, without the need for key people to do more than suggest changes and approve the final result. This will free the creative people from the more burdensome aspects of their routine activities by transferring much of the responsibility to the PSO.

(b) Estimated cost = 5.0 MY $100K

NOTE: Much of this time and System cost is chargeable to specific programs.

c. The task on Information System Security is within the charter of ESD/MCI. We do not intend to support this task.

(1) Task = Information System Security

(a) Prepare appropriate specifications for security devices and equipment needed to handle unclassified and classified data of all levels within the same

information system, This specification should address both data base security and communications security, Emphasis should be on low cost components, RADC role is formal RT&E support,

d, The task on Archival Storage Tradeoff Comparison is an interesting study that should be performed using the most up-to-date information available on Microfiche systems and computer memories, RADC with very recent experience in both these areas is well qualified to contract for such a study, We would certainly be willing to do so if the funds were made available,

(1) Task - Archival Storage Tradeoff Comparison

(a) A need exists to store and maintain a body of data that changes rarely, if ever, but which may be needed on short notice for reference or other purposes, Air Force regulations, manuals and technical orders are examples of this category, Because of their bulk, and because the data is not dynamic in the base environment, there is question as to the desirability of computer storage for such data, This trade-off study should assess what data are appropriate for this class of storage and determine the costs and risks associated with each form of storage, Consideration must be given to legal requirements for records and documentation, to accessibility requirements, and to update techniques, RADC role is formal RT&E support,

(2) STUDY OF MICROFORM SYSTEMS VS TEXT-PROCESSING SYSTEMS

(a) Perform a study based on the current state-of-the-art of both data processing and microfiche systems that can lay to rest once and for all the controversy over the candidacy and the relative merits of the video/microfiche approach to processing textual information as compared to modern text-processing technology, where archival requirements are not of paramount importance,

(b) Estimated cost - ,2 MY $50K

e, The Task Mobile Subscriber Communications task has no apparent intersection with the interests and capabilities of IS, We plan to do nothing in this area,

(1) Task Mobile Subscriber Communications

(a) This task addresses the development and test of a number of alternative approaches to providing the mobile subscriber services described in the Base Communications Mission Analysis. Each alternative must be capable of being integrated into the AFBITS communication system so that frequent consultation with ESD/MITRE will be required. Various signal structures, frequency spectra, and multiplexing techniques should be examined, and at least two alternative approaches carried through breadboard tests. General objectives include frequency conservation, low cost, light weight and small size. Product will be a performance specification for the unit showing the most promise. RADC role will be technical direction on this project. This activity is unlikely to be completed in FY-75 and intermediate schedules and objectives are open to negotiation.

2. The following is a brief restatement of the efforts, and manpower and estimated System cost:

a. FORMS SYSTEM*

Estimated cost = .3 MY $70K

b. COST/EFFECTIVENESS*

Estimated cost = .3 MY $20K

c. USER ACCEPTANCE OF ON-LINE TERMINALS*

Estimated cost = .2 MY $10K

d. STUDY OF MICROFORM SYSTEMS VS TEXT-PROCESSING SYSTEMS

Estimated cost = .2 MY $50K

e. PERSONNEL SERVICE ORGANIZATION (PSO)

Estimated cost = 5.0 MY $100K

NOTE: Much of this time and System cost is chargeable to specific programs including WWMCCS Support.

TOTAL COST = 6.0 MY $250K

3, The costs of manpower and the cost of the system in performing the tasks marked with the '*' will be borne by IS in their entirety, However, the contractual costs of $50K for the FORMS SYSTEM and the $50K for the STUDY OF MICROFORM SYSTEMS VS TEXT-PROCESSING SYSTEMS cannot be borne by IS, and we request that money be provided for these two valuable studies,

JOHN L, MC NAMARA, Chief
Information Management Sciences Section
Information Processing Branch

RJC 29-MAY-74 07:51 30823

Kenyon Memo to DCLP

(J30823) 29-MAY-74 07:51; Title: Author(s): Roberta J. Carrier/RJC;
Distribution: /EJK; Sub-Collections: NIC; Clerk: RJC;
Origin: <CARRIER>KENNEDYJOB.NLS;1, 23-MAY-74 11:22 RJC ;

The SUBSTITUTE TEXT Command

A reference I sent Brian McNally of ETS,

The SUBSTITUTE TEXT Command

The Substitute command allows you to very easily edit the text within an NLS statement. NLS has a rich assortment of commands which allow you to freely manipulate the information in your files. This command is presented early in the training because it is simple but very powerful, though sometimes awkward. (The full NLS vocabulary comes with the next phase of training.) 1

The Substitute command replaces one string of text in a statement with another. 2

You begin by telling the system to Substitute Text in a Statement, and then tell it which statement to operate on.
If you type:
sts,1<CR>
it should look like:
*Substitute Text in Statement A: ,1
<New Text> T: 2a

You may then type in the new string of text, followed by a Carriage Return:
this is what I want<CR>
and it should look like:
<New Text> T: this is what I want
<for Old Text> T: 2b

Then type the string which you would like replaced:
this is what I had<CR>
and it should look like:
<for Old Text> T: this is what I had
Finished? 2c

A final Carriage Return will begin the process. 2d

Actually, the Substitute command replaces EVERY occurance of a given string of text by another (in that statement). Therefore you must be very careful to define a string uniquely. 3

For example, if statement one looked like:
that man is over thare
and you wished to correct the misspelled "there": 3a

It is easy to make the mistake of: 3a1

*Substitute Text in Statement A: ,1
<New Text> T: the
<for Old Text> T: tha
Finished?
Substitutions made = 2
* 3a1a

Your statement would then look like:

```
   thet man is over there
    *                 *
```

note that it replaced both occurances of "tha" 3a2

What you should have done is made sure the strings are uniquely defined: 3a3

```
*Substitute Text in Statement A: ,1
<New Text> T: there
<for Old Text> T: thare
Finished?
Substitutions made = 1
*
```
3a3a

In longer statements, this is even more important. It is better to type too much (a bit of the previous word and a bit of the next, perhaps) and be safe. Remember to construct your new string so that it puts back in any extra that you included to uniquely define the string (e.g. "this for that" to replace "this fr that"). 3b

The SUBSTITUTE TEXT Command

(J30829) 29-MAY-74 09:06; Title: Author(s): N. Dean Meyer/NDM;
Distribution: /JHB TO; Sub-Collections: SRI-ARC; Clerk: NDM;
Origin: <MEYER>SUBS.NLS;2, 29-MAY-74 09:02 NDM ;

Response to IMM about Archive, Privacy and an Ident

Inez, 1

The question about archive is a good one. The reason they stay there forever is because the tapes are written sequentially, with each file that is archived added to the reel of tape in chronological order. Once a tape is used up, the record ring is removed to ensure that the files are not errased. The only people who can restore a file from archive are the people who have the password to the directory from which the file came. But we have no way of going back to a particular place on the tape to errase a file without endangering the rest of the tape, 1a

I am in the process of preparing a detailed description of the protection mechanism that has recently been built. Sorry it's taking awhile to get it together. In the Journal when you set a Journal item to private by typing the "PRivate" command, only those persons on the distribution list can load or read the file. The mechanism for this interacts with Tenex to use the file protection codes -- it's pretty safe, 1b

We'll be glad to set up the ident for you, but will need more information: 1c

Full name
Address
Phone number
He will receive Journal mail in hardcopy unless you request a directory for him,,,it will be mailed to the address specified for the ident system, 1c1

FEED 29-MAY-74 16:08 30830

Response to IMM about Archive, Privacy and an Ident

(J30830) 29-MAY-74 16:08; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /IMM JHB; Sub-Collections: SRI-ARC; Clerk: FEED;

copy of a norton message to bedford re the SEND construct for L-10; includes sample programs

(NORTON) 13-MAY-74 0819-PDT 110 and the send construct
Distribution: BEDFORD, norton
Received at: 13-MAY-74 08:19:31 1

- - - -

Mike: I have made a file that has two versions of a program that reformats sndmsgs that have already been brought into NLS via inmes pprogram. These programs are different from each other in that the first one actually reformats each statement that passes the search test,,like those we were doing when I was there. The second one has the "send" construct we were looking for. It only "pretends" to reformat statements, sending the special reformatted view to your terminal, but not changing the st
ement. When viewspec j is returned, the statement is as it was in the first place. BUT when you use the assimilate command and have the second program compiled and loaded and use viewspec i, the new statemennts that you assimilate ARE reformated permanently, but leaving the original statements (the ones you are copying in the assimilate) untouched. I hope this is clear enough. If not, let me know. Note that you have to declare a string,,like *out* and have to place the (sw) after PROCEDUR
and have to use the EXACT wording in send8,,,,etc) and then set *out* _ NULL;,, I tthink these re the onlyy difference,, Good luck,,and if troubles,,see me. here's are the programs: 1a

```
FILE 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
FILE reform2
   DECLARE TEXT POINTER sf, pt1, pt2, pt3, pt4;
   DECLARE STRING out[2000];
   (reform2) PROCEDURE (sw);
      IF FIND "sf $NP $D '= $L '= $D $NP $PT "pt3 CH $NP "pt1
[SP/'];
      "pt4 < CH "pt2 > THEN
           *out* _ '(, pt1 pt2, '), "  ", SF(sf) pt3, " ", pt4
SE(sf);
         send (sw,$out);
          *out* _ NULL;
       RETURN(FALSE) END,
    FINISH
```

1b

MIKE 30-MAY-74 10:27 30831

copy of a norton message to bedford re the SEND construct for L-10;
includes sample programs

1c

MIKE 30-MAY-74 10:27 30831

copy of a norton message to bedford re the SEND construct for L-10;
includes sample programs

(J30831) 30-MAY-74 10:27; Title: Author(s): Michael T. Bedford/MIKE;
Distribution: /PAN; Sub-Collections: NIC; Clerk: MIKE;

Chapter 10 1

INDEXING 1a

10.1 Introduction 1a1

"Indices, in JOVIAL, have two purposes. A "go:to:statement that references a "switch:statement may have an "index. In this case, the "index is a determining factor in the selective operation of "statements within the "switch:statement. This use of an "index is completely specified in Section 5.13.5. But the most predominant use of an "index is selecting a particular occurrence of an item or entry from a table. The rest of this chapter discusses various aspects of this indexing process. 1a1a

10.2 Data References 1a2

An "index is used to select a particular entry from a table composed of many similarly constructed entries. In usage, an "index is enclosed in "brackets. It is composed of "index:components which, if there are more than one, are separated by "commas. Reference to an item or an entry in a multidimensional table requires a multicomponent "index. "Index:components, however, do not have to be explicitly stated in some circumstances (see Section 10.3). 1a2a

.1 Every "table:declaration has a "dimension:list giving the number of dimensions and, for each dimension, the "lower: and "upper:bounds that give the extent or size of the table in each dimension (see Section 7.22). These same "numbers give the range of values permissible for "index:components for each dimension. Every "index:component must reduce to an integer value such that for its corresponding dimension it obeys the following relation: 1a2a1

"lower:bound $\leq$ "index:component $\leq$ "upper:bound 1a2a1a

.2 Since "lower:bounds may be other than zero, the meaning of a "linearized" "index is difficult to determine. Therefore, if any "index:component is outside the range of its "lower:bound and "upper:bound, its meaning is undefined. It is also undefined to have more "index:components than there are dimensions in the corresponding table. 1a2a2

.3 Each "index:component is concerned with one dimension of the table to which it is applied. If each

"index:component is considered in turn, it is seen that, starting from the left, they select: 1a2a3

The _n-1 dimensional manifold within the _n dimensional table, 1a2a3a

The _n-2 dimensional manifold within that _n-1 dimensional manifold, 1a2a3b

. 1a2a3c

. 1a2a3d

. 1a2a3e

The plane within that volume, 1a2a3f

The row within that plane, 1a2a3g

The particular entry within that row, 1a2a3h

.4 The above order corresponds to the order in which dimension bounds occur in a "dimension:list. A table _MN with the "dimension:list [2, 3, 1] has three planes, four rows within each plane and 2 entries in each row. Note that since the "lower:bounds for each dimension default to the value zero, there is one more volume, plane, and row than the value that expresses the "upper:bound. The following example is a reference to a particular entry within that table; it so happens that the reference is to the last entry, 1a2a4

The number _2 (or third) plane 1a2a4a

The number _3 (or fourth) row 1a2a4b

The number _1 (or second) entry 1a2a4c

MN[2, 3, 1] 1a2a4d

.5 In any reference to a table or item in a pointed-to structure, the pointer, whether implicit or explicit, points to the beginning of the structure. Indexing within a pointed-to structure proceeds as a displacement from the beginning of the structure as determined from the value of the pointer plus the displacement, if any, from the beginning of the pointed-to structure to the beginning of the referenced structure plus, finally, the displacement due to the indexing. Notice that generally

the value of the pointer is not the value of the "location:function:call. The value of the "location:function:call incorporates all the relevant displacements. If such a value is used as an explicit pointer to an "indexed:variable within a data structure, the relevant displacements are counted again in completing the reference. 1a2a5

10.3 Missing "Index:Components 1a3

"Index:components or even the complete "index may be sometimes omitted. The meaning depends upon the situation. If initial or embedded "index:components are missing, their places must be marked by a "comma. Trailing "index:components can merely be omitted. 1a3a

.1 The primary meaning of an omitted "index:component in a data reference is to use the corresponding "lower:bound instead. If all "index:components are omitted, the "brackets and the "commas that hold the positions of "index:components may also be omitted. Example: 1a3a1

```
TABLE  $1  [9,9,1:10] P F ;                                  1a3a1a

TABLE  $A  [9,9 ] S;                                         1a3a1b

$1[5,6] = $A[,9];    $A[,9] = $1;                            1a3a1c
```

.2 In the above example, neither table has any infrastructure. _$1 is a three-dimensional, parallel table of floating entries. The first two dimensions run from _0 through _9 and the third dimension runs from _1 through _10. _$A is a two-dimensional, serial (default) table of signed integer entries. The first "assignment:statement puts the _[0,9]th value of _$A into the _[5,6,1]th entry of _$1. Then the second "assignment:statement puts the first value of _$1 (entry _$1[0,0,1]) into the _[0,9]th entry of _$A. 1a3a2

.3 The second meaning of the omitted "index in a data reference is involved with a pointed-to table having a submanifold "allocation:increment. In a reference to such a table in which the pointer is implicit, omitting an "index:component usually has the primary meaning of "lower:bound. In the "declaration of the pointed-to table, "dimension:list components relating to the pointer must be omitted. Example: 1a3a3

```
TABLE  PTR [5:10,17:23] U ;                                  1a3a3a
```

```
TABLE  TAB @ PTR:1 [ , , 1: 100];                                  1a3a3b

ITEM   AA, BB  S 47, 15;                                           1a3a3c

AA[7, , 47] = BB[8,20];                                            1a3a3d
```

,4 In this example, _PTR is a two-dimensional table of unsigned integers, with unusual "lower:bounds Table _TAB is three dimensional, allocated by rows, and pointed to by _PTR. In its "declaration, the first two "dimension:list components, actually relating to _PTR, must be empty. _TAB contains two fixed, signed items, _AA and _BB. In the "assignment:statement the first two "index:components of the "indices for _AA and for _BB refer actually to _PTR. The value of the first element of _BB (defaulted "lower:bound for the third "index:component) in the row pointed to by PTR [8,20] is given to the instance of _AA designated by an "index:component of _47 in the row pointed to by _PTR[7,17] (defaulted "lower:bound for the second "index:component). 1a3a4

,5 If there is an explicit pointer in the reference to a pointed-to entity allocated by submanifolds, all "index:components must be omitted except those indexing into the submanifold. If any of those indexing into the submanifold are missing, they default to their respective "lower:bounds. Example: 1a3a5

```
AA @ 985 = AA@985 +AA[,,K]@985;                                    1a3a5a
```

,6 In the first two occurrences of _AA in the above example, the entire "index, including the "brackets, is omitted. The first two "index:components must be omitted because the explicit pointer, _985, is used. Omitting the third "index:component defaults to its "lower:bound, _1 in this case. In the third occurrence of _AA, the first two "index:components must be omitted and the third one, _K, selects the element of the row starting at location _985. 1a3a6

,7 A missing "index:component in connection with a "number:of:entries:function:call causes that "index:component to be ignored in computing the number of entries (see Section 10,4,10). 1a3a7

10,4 "Index:Ranges 1a4

In a list of data elements occurring in an

"assignment:statement, whether being assigned values, or used in assigning values, or involved in formatting translations, it is often desirable to have an "index:component take on a sequence of values from some "low:point to a "high:point. This can be done by using an "index:component:range or, more generally, an "index:range. 1a4a

.1 "Low:point and "high:point must reduce to integer values such that for the corresponding dimension: 1a4a1

"lower:bound _< "low:point _< "high:point _< "upper:bound 1a4a1a

.2 If "low:point is missing, it defaults to "lower:bound; if "high:point is missing, it defaults to "upper:bound. If only the "colon is given for an "index:component:range, the meaning is from "lower:bound through "upper:bound, inclusive. The general meaning of "index:component:range is from "low:point through "high:point, inclusive. Thus: 1a4a2

_I: means from _I through "upper:bound, and 1a4a2a

_:J+2 means from "lower:bound through _J+2, and 1a4a2b

_: means from "lower:bound through "upper:bound. 1a4a2c

.3 An "index:range from the definition given above is seen to be an "index in which one or more "index:components may be replaced by "index:component:ranges. An "index:range used with an "indexed:variable to represent a list of data elements in an "assignment:statement results in an "indexed:variable:range. 1a4a3

.4 The meaning of the first form of "indexed:variable:range given above is the same as a list of several occurrences of the "item:name or "table:name in which each "index:component:range is replaced with the successive values of the "index:component it replaces. If "index:component:ranges replace more than one "index:component, the meaning is as if such expansion of occurrences of the "item:name ("table:name) happened first with respect to the leftmost "index:component:range and then progressed through "index:component:ranges to the right. For example, consider the "indexed:variable:range 1a4a4

BB[, :19, 99:] 1a4a4a

which references the data structure of Section 10,3,3, This can be seen to be equivalent to the partial expansion 1a4a5

BB[,17,99:], BB[,18,99:], BB[,19,99:] 1a4a5a

which in turn is equivalent to the full expansion 1a4a6

BB[,17,99], BB[,17,100], BB[,18,99], BB[,18,100], 1a4a6a

BB[, 19,99], BB[,19,100] 1a4a6b

,5 Notice, in the example above, that the missing first "index:component for _BB means "lower:bound, ,B=1;not,B=0; "lower:bound through "upper:bound, 1a4a7

_BB is shorthand for BB[, ,] 1a4a7a

it is ,B=1;not,B=0; shorthand for _BB[:,:,:] 1a4a7b

,6 There is a special "primitive, _ALL, that may be used as a shorthand for a reference to an "indexed:variable:range in which all "index:components are replaced with "colons: 1a4a8

ALL (BB) ,B=1;is,B=0; shorthand for _BB[:,:,:] 1a4a8a

,7 Note that _ALL may occur only in an "assignment:statement to represent a list of data elements, In this case 1a4a9

_ALL _("name_) means "name _[: _] 1a4a9a

where there are as many "colons as the dimensionality of the named entity, _ALL must not be used unless the entire table is allocated space at this time, 1a4a10

,8 An "index:range may be used in one place other than an "indexed:vaiable:range, This is with the "number:of:entries:function:call, 1a4a11

,9 "Index:component:range has a different meaning when used in the context of _NENT, When used with an "indexed:variable:range, the meaning is "low:point through "high:point indicating a sequence of values to be used as "indices, With _NENT, "index:component:range means a single value to be used as a multiplier in determining the number of entries, That value is: 1a4a12

"high:point + 1 - "low:point 1a4a12a

,10 In the "number:of:entries:function:call, there is no concern about specific "index:components. In fact, the value of the "number:of:entries:function:call is just the product of multiplying together the extent of the referenced table in all the dimensions indicated by "index:component:ranges. The "index:components not represented by "index:component:ranges can be considered as multipliers of value _1 whether they are missing or not. With reference to the examples in section 10.3.3, the value of 1a4a13

NENT (TAB[, , :]) 1a4a13a

is "upper:bound + 1 - "lower:bound (using default values for "low:point and "high:point) for the third dimension which gives 100 + 1 - 1 = 100, the length of a row. The value of 1a4a14

NENT (PTR[:]) is the same as the value of 1a4a14a

NENT (PTR[:,]) and the same as the value of 1a4a14b

NENT (PTR[:,20]) 1a4a14c

which is 10 + 1 - 5 = 6 1a4a14d

,11 Since, in the context of _NENT, there is no utility in omitting all "index:components, there is an exceptional meaning to a "table:name without any "index or "index "brackets. Only in context with _NENT, 1a4a15

NENT("table:name) 1a4a15a

means the same as 1a4a16

_NENT _("table:name _[: _] _) 1a4a16a

wherein there are as many "colons as the dimensionality of the table. 1a4a17

10.5 Indexing Tables With Variable Bounds 1a5

Pointed-to tables can have "upper: and "lower:bounds that are "simple:integer:variables. Ordinarily, the "program:declaration would set the bounds before beginning a process involving the table, allocate space for the table (perhaps using the "number:of:entries:function:call and the

"words:per:entry:function:call), start and complete the process, and then release the space. To change the bounds during processing of the table could have undesired effects on the indexing of other entries, but this is not illegal--in case the programmer needs to make such changes and understands the ramifications. 1a5a

.1 Such tables can involve the "parameters of a procedure. The "lower: and "upper:bounds of a local table might be "formal:input:parameters; the table itself might be a "formal:input:parameter; or both the table and its bounds might be "formal:input:parameters. Here the ordinary sequence of events is: set the "lower: and "upper:bounds from the corresponding "actual:input:parameters, allocate space, process, release space. 1a5a1

.2 If the table itself is a "formal:input:parameter, the allocation and releasing of space is accomplished by the calling program. The location provided as the "actual:input:parameter is the location to be used by the procedure as the beginning of the table. The entry whose "index is all the "lower:bounds of this table begins at this location. 1a5a2

.3 This is the situation whether the "lower:bounds are set by "parameters, outer data, local data, or "constants. Even if the bounds of the table are changed during execution of the procedure, the entry whose "index is all the "lower:bounds always begins at the location specified by the pointer--implicit if not mentioned at the reference, explicit otherwise. 1a5a3

10.6 Indexing as Related to Storage Allocation 1a6

The order for storage allocation for a multidimensional table is elements of a row, rows of a plane, planes of a volume, etc. This order is the same as that of the values in a "constant:list used in presetting a multidimensional table (see Section 7.13). 1a6a

.1 Entries are allocated storage so that the first "index:component varies most slowly, the second "index:component the next most slowly, and so forth. The allocation for a table with a "dimension:list _[2,3,1] is given in Figure 10-1. Each entry of the table has a two-word item _AB (_1 _AB means the first half of _AB; 2 AB means the second half of _AB) followed by a one-word item _XY. The figure relates the "index, "item:name and

relative position within the table for both parallel and serial structures, 1a6a1

.2 A table with tight structure might be allocated storage in different ways. Figure 10-2 depicts two possible allocations on a machine that has 32 bits per word for the table whose declaration is: 1a6a2

```
TABLE BB [5,7] T D U 10 ;                                    1a6a2a
```

The allocation on the left requires more words of storage than the one on the right, but might result in more efficient usage since the storage structure for each row is the same, 1a6a3

.3 The examples show effective allocation structures. There is no requirement that storage actually be allocated in the manner shown. 1a6a4

J73 = Chapter 10 = Completely Edited

(J30835) 3-JUN-74 11:14; Title: AUthor(s): Roberta J. Carrier/RJC;
Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;
Origin: <CARRIER>JOVIALCHAP10EDIT.NLS;1, 3-JUN-74 11:06 RJC ;

Summary of Execute Edit Commands...not important unless you're thinking of using E E in a trial basis.

MIKE 3-JUN-74 11:20 30836

Summary of Execute Edit Commands,,,not important unless you're thinking of using E E in a trial basis.

(J30836) 3-JUN-74 11:20; Title: Author(s): Michael T. Bedford/MIKE; Distribution: /IMM MIKE; Sub-Collections: NIC; Clerk: MIKE;

NJN 3-JUN-74 14:16 30837

Journal message recovery

Jim--
If it is not too much trouble, I guess I would appreciate your making a pass through the journal to pick out stuff authored by me or sent to USING or INFORMAN groups. The personal mesages could not have been that important anyway.
Thanks very much, Nancy 1

Journal message recovery

(J30837) 3-JUN-74 14:16; Title: Author(s): Nancy J. Neigus/NJN;
Distribution: /JEW; Sub-Collections: NIC; Clerk: NJN;

RMS 3-JUN-74 15:34 30838

FTP Inquiry - Attn: John Postel

- - - -

John,

The FTP presently available at socket 3 supports a very small subset of

the old protocol. A full implementation of the new protocol is being

developed, but won't be available until August. ---Ron Stoughton

----

RMS 3-JUN-74 15:34 30838

FTP Inquiry - Attn: John Postel

(J30838) 3-JUN-74 15:34; Title: Author(s): Ronald M. Stoughton/RMS; Distribution: /JBP; Sub-Collections: NIC; Clerk: RMS;

JRP 4-JUN-74 11:25 30839

undelivered mil

- - - -

Ron,

Did you notice that your msg to jbp didnt get delivered?

J, Pickens

JRP 4-JUN-74 11:25 30839

undelivered mil

(J30839) 4-JUN-74 11:25; Title: Author(s): John R. Pickens/JRP;
Distribution: /RMS; Sub-Collections: NIC; Clerk: JRP;

FEED 4-JUN-74 18:37 30840

Bug at Office-1, tertiary distribution

Jim, If you haven't already would you please look at this? (Dean, the ident for Office-1 is FEED, dir is Feedback @ Office-1 for sndmsg. Thanks.)

Bug at Office-1, tertiary distribution

4-JUN-74 0548-PDT MEYER at OFFICE-1: Load Remite doesn't work this morning,
Distribution: FEEDBACK, white at SRI-ARC
Received at: 4-JUN-74 05:50:49 1

I'm at office-1, Tried Load Remote, Got error message (two times) 431 User name not valid, 1a

Bug at Office-1, tertiary distribution

(J30840) 4-JUN-74 18:37; Title: Author(s): Special Jhb Feedback/FEED; Sub-Collections: SRI-ARC; Clerk: FEED; Origin: <MJOURNAL>23229,NLS;1, 4-JUN-74 11:31 XXX ; Title: Author(s): Susan R. Lee/SRL; Distribution: /NDM JEW; Sub-Collections: SRI-ARC; Clerk: SRL;

MIKE 5-JUN-74 13:50 30841

propose agenda (abbreviated) for BPSC in Dewittville

we can get together later to talk this over. We should probaly route the agenda before the meeting to give people a chance to look it over and formulate a few thoughts.

propose agenda (abbreviated) for BPSC in DeWittville

(bpsc) Notes for BPSC, Dewittville Branch 1

review different levels of involvement/expertise 1a

review different applications and uses 1b

develop some sort of common expectations about where this thing is going over the next year, 1c

plan for raising the level of expertise throghout the group, 1d

MIKE 5-JUN-74 13:50 30841

propose agenda (abbreviated) for BPSC in Dewittville

(J30841) 5-JUN-74 13:50; Title: Author(s): Michael T. Bedford/MIKE;
Distribution: /IMM; Sub-Collections: NIC; Clerk: MIKE;

NJN 5-JUN-74 15:28 30842

Journal Recovery

Thanks very much for your efforts, Jim. At times like this one can really apppreciate all the nice packrat qualities of the journal. It may take some effort but at least things are recoverable. Thanks again, Nancy 1

NJN 5-JUN-74 15:28 30842

Journal Recovery

(J30842) 5-JUN-74 15:28; Title: Author(s): Nancy J. Neigus/NJN;
Distribution: /JEW; Sub-Collections: NIC; Clerk: NJN;

Chapter 11 1

"DIRECTIVES 1a

11.1 Introduction 1a1

"Directives provide supplemental information to a compiler about the program being compiled. One "directive supplies descriptions of definitions external to the program. Others control the compilation process or request debugging facilities. Some permit a compiler to do a better job of code generation by specifying referenced data, frequency of reference, etc. 1a1a

.1 The "directives listed above are part of standard JOVIAL. Other "directives may be added to the language at the implementer's option. Any "directives added must fit the form described in the next paragraph. 1a1a1

.2 A "directive is indicated by the use of an "exclamation:point followed immediately by a unique key word. Optional parameters (depending upon the "directive) and a "semicolon complete a "directive. The key word may be the same as a JOVIAL "primitive or a "name in the program since the "exclamation:point flags it as a "directive:key. The following definition gives the "directive:keys in standard JOVIAL: 1a1a2

.3 "Directives may appear before a "program:declaration and before or after a "statement or "declaration. "Directives are not considered as "statements or "declarations. Some "directives are restricted more as to where they may appear. These restrictions are noted for the individual "directives. A "directive following the single "statement that forms the body of a "program:declaration is lost; the compiler does not look beyond the final _END for further direction. 1a1a3

11.2 "Compool:Directive 1a2

All "names to be used in a JOVIAL program must be declared except those known intrinsically to the compiler. "Names whose attributes exist in a compool or library are "declared" using the "compool:directive. 1a2a

.1 If multiple compools exist for an implementation, "compool:name identifies the compool to which the "directive applies. "Compool:name may be omitted for single-compool implementations or if a default system

compool exists. If there is only one compool and the entire compool is to be included, the "compool:directive is: 1a2a1

_!COMPOOL; 1a2a1a

.2 The "names in the "compool:directive direct the accessing of those "names' attributes from the indicated compool. "Table:names, "item:names, "data:block:names, "form:names, "define:names, "status:list:names, "statement:names, "program:names, "procedure:names, and "alternate:entrance:names may all be included in a compool. The significant attributes associated with "program:names and "alternate:entrance:names would be identical to those of "procedure:names. 1a2a2

.3 Certain "names may be enclosed in "parentheses with the following significance: 1a2a3

a. A "data:block:name in "parentheses declares the data block and all data declared within it. 1a2a3a

b. A "table:name in "parentheses declares the table and all of its table items. 1a2a3b

c. A "compool:name in "parentheses declares all "names within it. 1a2a3c

.4 All "compool:directives precede the "program:declaration and serve to establish a compool scope outside the source program in which the "names indicated in the "directive are assumed declared. This provides for overriding any "name declared in a compool at any level of source program scope; a "declaration would be in effect for the local scope at which the "declaration occurred as well as any inner scopes not containing a "declaration for the same "name. Ordinarily, compool data are in reserve, but there is no reason why pointed-to structures cannot be defined in the compool--perhaps with other compool data as implicit pointers. 1a2a4

11.3 Conditional Source "Directives 1a3

The following "directives are used to control or bracket conditional source compilation. 1a3a

.1 The "begin: and "end:directives bracket the "statements and "declarations which are to be

conditionally compiled. A "begin:directive must always be followed by a corresponding "end:directive and corresponding pairs of these "directives must not be overlapped or nested. 1a3a1

.2 The "skip:directive directs that "statements within the related "begin: and "end:directives (as indicated by "reference) are to be skipped and not compiled. If the "reference is absent from the "skip:directive, the "statements within all "begin: and "end:directives are skipped. If "reference is omitted from the "begin:directive, the "statements following, until the next "end:directive, are affected only by a "skip:directive with "reference omitted. 1a3a2

.3 In the program example below, the calls on _PRINT are suppressed by the "directive !SKIP 1;. !SKIP 3; causes the "statement printer _ERR to be suppressed. The "statement bracketed by !BEGIN 2; and its corresponding _!END; are not suppressed because there is no _!SKIP; or !SKIP 2; "directive. 1a3a3

```
                    PROGRAM PROG;  !SKIP 3;                    1a3a3a
            _BEGIN                                             1a3a3b
                    !SKIP 1;                                   1a3a3c
                    TABLE  AA [100]  S  10;  NULL;             1a3a3d
    LAB;            !BEGIN 1;  PRINT('LAB');  !END;            1a3a3e
            FOR  I(0  BY  1  UNTIL  I  >  100)   J(0 THEN
    J);                                                        1a3a3f
            _BEGIN                                             1a3a3g
                    !BEGIN 2;                                  1a3a3h
                        IF  I  >  50;  OUT('I', AA[I]);        1a3a3i
                    _!END;                                     1a3a3j
    LOOP;           _!BEGIN 1;  PRINT('LOOP');  !END;          1a3a3k
                    IF  AA[I] = 0;                             1a3a3l
                    _BEGIN                                     1a3a3m
```

```
                                AA[I]=AA[J];  J=J+1;                    1a3a3n
                                IF  J > 100;                            1a3a3o
                                _BEGIN                                  1a3a3p
                                     !BEGIN 3;  OUT('ERR', I);
          !END;                                                         1a3a3q
                                     GOTO EX'IT;                        1a3a3r
                                _END                                    1a3a3s
                           _END                                         1a3a3t
                           _AA[I]=AA[I]+1;                              1a3a3u
                     _END                                               1a3a3v
          EX'IT;           !BEGIN 1 ; PRINT('EXIT');  !END;            1a3a3w
                                _STOP;                                  1a3a3x
                     _END                                               1a3a3y
```

11,4 "Trace:Directive 1a4

The "trace:directive provides for dynamic tracing of control flow and the recording of data at the point of modification, 1a4a

,1 If a "statement:name, "procedure:name, or "alternate:entrance:name appears in the list of "names, execution at the named point in a program is noted symbolically at run time; execution in this context is meant to describe either a branch or fall through to a "statement:name or invocation of a procedure or alternate entrance, 1a4a1

,2 If the "name in the list is a "data:name, modification of the datum is noted along with the "data:name and its new value; modification may occur by assignment, exchange, or by usage as a "parameter. If the "data:name is a "table:name, tracing is performed for the table and its table items; for a "data:block:name, tracing is for the data block and all items, tables, and data blocks within it, 1a4a2

,3 The optional "conditional:formula, if present, is tested at any trace point and tracing is not performed if

the "formula is zero (false). If the "conditional:formula is _1 (true) or not present, the trace is performed. 1a4a3

.4 It is expected that trace output will be transmitted to the standard output file but the actual disposition of this output is system dependent and is specified by the implementation. 1a4a4

11.5 "Copy:Directive 1a5

The "copy:directive is used to cause an inclusion of JOVIAL source code into the "program:declaration from an external source. 1a5a

.1 The "character string is implementation defined and contains parameters to describe the source to be included; i.e., the file name, version name, library name, device name, line range, etc. It must not, of course, include a "semicolon. 1a5a1

.2 The copied source code is processed as if it had occurred in the "program:declaration at the point of the "copy:directive. 1a5a2

11.6 Compiler Optimization Philosophy and Assumptions 1a6

Ideally, a compiler optimizes code to produce a program which occupies minimal space and has a minimal run time. In practice, this is difficult, if not impossible, to achieve. With the inclusion in the language of such features as "overlay:declarations and "specified:table:declarations (which describe space sharing) and pointed-to data, any optimization at all might have a deleterious effect upon the result of a program. 1a6a

.1 The philosophy of JOVIAL with regard to optimization is to define the language with a reasonable compromise between object program efficiency and insurance of achieving the intended result. To accomplish both, the basic philosophy leans toward "safe" processing of the program "statements with a supplemental facility of using "directives to inform the compiler more completely of ramifications, associated with the data and with program logic, which would allow for more extensive optimizations. 1a6a1

.2 In general, if the supplemental information is not supplied, the compiler proceeds on a safe basis.

However, if any "directive information is supplied, the compiler may assume no impact exists other than that supplied, for the indicated "directive subject. For instance, when a ref (see Section 9,1,3) procedure is invoked, the compiler must assume all ref and compool data are set and used by the "procedure:call:statement. However, if the programmer includes "sets: and "uses:directives for the "procedure:declaration, the compiler may assume only those data elements named in the "directives are set or used. 1a6a2

.3 Taking the safe approach has the obvious benefit of protecting the novice or unsophisticated user. On the contrary, assuming the worst in the name of safety precludes many optimizations which probably are safe since the events which could have an adverse impact on these optimizations are very unlikely to occur. Consequently, the compiler compromises safety by making possibly unsafe assumptions in the critical areas mentioned below (Section 11,6,4 through section 11,6,11) unless the programmer provides "directives to do otherwise. 1a6a3

.4 The compiler assumes that data are normal, i.e. data are not modified by interrupt processing or concurrent tasks (parallel processing). This is not generally a bad assumption since many systems do not have a capability for parallel processing and the compiler may know that interrupt processing does not modify program data. The programmer may use the "abnormal:directive (Section 11,7,1) to override this assumption. 1a6a4

.5 From the viewpoint of the "procedure:call:statement, the compiler assumes that a procedure references only entities within its scope, "actual:input: and "actual:output:parameters, and def (see section 9,1,3) entities. It further assmes that the results of functions having no "parameters other than "simple:items, no reference to outer entities, and no pointed-to, environmental, or reserved data are reproducible for identical "parameters. The "sets:directive and "uses:directive (Section 11,7,2) may be used to override these assumptions. 1a6a5

.6 The compiler assumes that overlay or interference does not result from the use of pointers. The "pointer:directive (Section 11,7,3) can inform the compiler that interference may result from the use of pointers. 1a6a6

.7 The compiler assumes that computations may be reordered (even around "procedure:call:statements providing it can determine by scanning the procedure that pertinent data are not referenced). The programmer may use the "order:directive (Section 11.7.4) to prevent this assumption. 1a6a7

.8 The compiler assumes that procedures are not recursive unless there is direct self-invocation. The "recursive:directive (Section 11.7.5) can be used to indicate that a procedure can be called recursively or reentrantly. 1a6a8

.9 The compiler assumes that interference results from item overlays within an entry word only if their bits overlap (by "specified:table:item:declarations or "subordinate:overlay:declarations) and that interference also results from overlays as described by "independent:overlay:declarations. It assumes that interference does not result from entry overlaps caused by variable-entry-length tables or from any implied overlay caused by the use of absolute addresses. The "interference:directive (Section 11.7.6) can be used to point out where such interference does occur, but would not be assumed by the compiler. 1a6a9

.10 A variable-entry-length table is a table declared by a "specified:table:declaration containing "specified:table:item:declarations placing items in the entry beyond the size of the entry specified in the "specified:table:heading. Whether any significant use can be made of such a structure is system dependent, but it seems highly unlikely unless the activity is confined to an allocation submanifold of a serial table. The chances for successful usage decrease as the attempt tries to encompass submanifolds (even within an allocation submanifold) of greater dimensionality. The greatest probability for success lies within a row. 1a6a10

.11 The compiler assumes that "formal:input:parameters to a procedure are independent of each other and of other data. Again, the "interference:directive may state that this is not the case. 1a6a11

11.7 "Directives for Code Optimization 1a7

The "directives described in this section are intended to provide a compiler with information that is useful for optimizing the object program. Some of them may be used to

override assumptions that might normally be made by the compiler (see Section 11.6). Others are independent of compiler assumptions and could be ignored by the compiler without affecting the validity of the object program. 1a7a

.1 "Data:name in the "abnormal:directive is that of any previouly declared data element and indicates that modification of the data element may occur by interrupt processing or concurrent tasks. This "directive must appear before the first reference to the "data:names in the list. 1a7a1

.2 The "sets:directive and "uses:directive apply to "direct:statements and certain procedures. In "direct:statements one or both "directives immediately follow _DIRECT and are used to inform the compiler of data referenced in the "direct:statement. If either "directive occurs in a "direct:statement, the compiler may assume the list or lists of "data:names are all inclusive. In a ref "procedure:declaration (see Section 9.4), these "directives inform the compiler of data referenced in the external procedure. The compiler may assume the list or lists include all "data:names the compiler need be concerned with for optimization purposes. In the "declaration of a procedure that is a "formal:input:parameter these "directives inform the compiler of data referenced in the procedures that will be "actual:input:parameters corresponding to this "formal:input:parameter procedure. The compiler may assume, if either "directive is used with a particular procedure, the sets and uses information is all inclusive. The "sets:directive and the "uses:directive for a procedure are placed between the "procedure:heading and the "procedure:body. 1a7a2

.3 The "pointer:directive indicates that reference to data via the "pointer:formula should be considered a reference to the "data:names in the list following the "colon. Several "pointer:formulas may be described with a single "directive. The "pointer:directive is placed anywhere after the "declaration of the "data:names referenced in the "directive but prior to any uses of the "pointer:formula. 1a7a3

.4 The "order:directive dictates that "formula computation order be from left to right and in accordance with precedence rules (Section 4.15) and that any "function:call within the "formula must be evaluated even though the eventual result is known. This "directive is

effective for the "statement immediately following it and prevents the compiler from reordering for code optimization in or with respect to that "statement. In addition, if this "directive is placed between a "procedure:heading and its "procedure:body, any "statement referencing this procedure or function is effected as if the "directive had preceded the referencing "statement. 1a7a4

.5 The "recursive:directive is placed between a "procedure:heading and its "procedure:body if the current procedure may be called recursively or reentrantly. This "directive restricts certain optimizations performed across a "procedure:call:statement which might be unsafe if a recursive entrance occurs. The "recursive:directive is assumed for procedures that call themselves directly. The recursive attribute does not cascade to called procedures; if called procedures are also recursive, it must be indicated by use of a direct self-invocation or the "recursive:directive. "Direct self-invocation" or "calling itself directly" means that the procedure uses its own "name or one of its own "alternate:entrance:names as a "procedure:call:statement or a "function:call. Direct self-invocation includes, of course, being called by an internal nested procedure (this does not necessarily make the nested procedure recursive). An example of the need for the "recursive:directive is exemplified by procedure _ABC which calls procedure _XYZ which calls _ABC. If, in fact, these calls will never be executed to the extent that either _ABC or _XYZ is doubly active, then there is no need for the "recursive:directive. On the other hand, if one of these procedures can be doubly active, then, of course, it must be programmed so its data space will be properly managed and it must contain the "recursive:directive for proper optimization. 1a7a5

.6 The "interference:directive is used to specify data elements that interfere with each other. The interference is specified by sets; the data element named to the left of the "colon interfereres with the data elements named in the list to the right of the "colon and vice versa. The data elements named in the list do not interfere with each other. More than one set may be specified; the sets are not related. The "interference:directive is placed after the "declarations for and before the first reference to any "data:name specified in the "directive. In the example: 1a7a6

```
    !INTERFERENCE  AA  :  BB,CC  DD  :  EE,FF,GG  FF  :
    GG;                                                    1a7a6a
```

mutual interference is indicated for _AA and _BB, for _AA and _CC, for _DD and _EE, for _DD and _FF, for _DD and _GG, and for _FF and _GG; no other interference exists for these data elements. The compiler may assume there is no interference (except that already known to the compiler) for data elements not named in "interference:directives. 1a7a7

.7 The "space:directive or "time:directive is intended to provide generalized direction that execution space or execution time, respectively, should be favored where the compiler has a choice. The extent to which space or time is compromised in favor of the other may be limited by an implementation-dependent parameter given in the optional "character string. For instance, the following "directive might direct the compiler to favor time versus space in its optimization but not to the extent that execution space increases by 10%: 1a7a8

```
    !TIME  10;                                             1a7a8a
```

A "space: or "time:directive is effective from the point at which it occurs in the "program:declaration until a subsequent "space: or "time:directive occurs. Ignoring these two "directives has no effect on the validity of the object program. 1a7a9

.8 The "linkage:directive is placed between a "procedure:heading and the "procedure:body and is used to describe a non-standard linkage for the procedure. The parameters given in the optional "character string are implementation dependent. For a system that has only one form of non-standard linkage, 1a7a10

```
    _!LINKAGE;                                             1a7a10a
```

could be the way to request that non-standard linkage. This "directive might not, in general, affect code optimization unless, for example, there were a parameter value that would indicate to the compiler that two "actual:input:parameters of the procedure could be passed in one word. 1a7a11

.9 The "frequency:directive has two purposes. When placed between the introductory clause of a "conditional:statement and its first

"controlled:statement, this "directive allows the compiler to favor the most frequent direction of evaluation. When placed between a "for:clause and its "controlled:statement, the "frequency:directive can provide the compiler with an expected repetition count (or perhaps a minimum count and a maximum count) to aid the compiler in determining whether certain optimizations that have set-up cost will pay off. In both cases the parameters given in the "character string are implementation defined. Ignoring a "frequency:directive has no effect on the validity of the object program. 1a7a12

J73 - Chapter 11 - Completely Edited

(J30844) 6-JUN-74 06:59; Title: Author(s): Roberta J. Carrier/RJC;
Distribution: /DLS; Sub-Collections: NIC; Clerk: RJC;
Origin: <CARRIER>JOVIALCHAP11EDIT.NLS;1, 6-JUN-74 06:44 RJC ;

The Future of USING

To USING members:

The recent changes in network funding have had some implications for USING's work. We are interested in hearing your opinions on "USING and the network", and in clarifying our role. In addition we would like to clarify the group's membership and activities, at least for the immediate future.

The network cutbacks affect us in two ways. First, the reduction in NIC services will make it more difficult to generate reports. While network mail is still available for communication, "group" writing and editing will be impacted; the more primitive methods of telephone and U.S. Mail will often have to be used. In addition, any online resources we use will have to come out of local contracts. Which brings us to the second point. Local budgets are tightening up and the funds for "community work" such as USING are becoming scarce. Many people feel they should spend their time tending their home fires first.

We sent a note to Craig recently asking his opinion on the role and responsibilities of USING. He replied, in part:

> I HAVE ALWAYS CONSIDERED THE USING COMMITTEE TO BE A TECHNICAL ADVISORY GROUP FOR ARPA/IPTO. THIS MEANS THAT YOU WOULD GIVE US TECHNICAL ADVICE ON HOW TO MAKE THE ARPANET BETTER FOR USERS, AND SO YOUR "OUTPUT" SHOULD BE REPORTS.
>
> [...]
>
> 1. I WOULD LIKE THE USING COMMITTEE TO PRODUCE THE TWO RECOMMENDATION REPORTS THAT WERE PROMISED, AND TO SUGGEST OTHER TECHNICAL AREAS IN WHICH WE (ARPA) MIGHT SOLICIT USING'S ADVICE.
>
> 2. SUGGESTIONS IN THE FUTURE WILL BE USED AS I HAVE USED SUGGESTIONS IN THE PAST - SELECT THE BEST PARTS AND INCORPORPORATE INTO OUR ARPANET PLANNING.
>
> Craig is expecting the other reports promised for the end of June and also hopes to attend the July meeting to discuss future plans. ARPA/IPTO is unable to provide any financial support for USING or its members, but will offer any "conversational" assistance required.

An alternate role for USING is that of direct consumer lobby, collecting complaints and suggestions of real (ss) users and then talking with servers to lobby for specific changes. To some project administrators, report writing may seem rather esoteric, whereas lobbying for specific change may be easier for them to justify. USING

members may therefore find it easier to gain authorization to participate in USING with this role. 5

All of this leaves us unsure of what steps to take. We would appreciate a response from all of you indicating the nature of your interest in USING, what function you would like to see it serve, how (and to what extent) you would be able to participate, and, more immediately, whether you expect to attend the July meeting (see journal item -- LJOURNAL, 30738, 1:W). 6

In view of the current situation, perhaps we should postpone the meeting until August or September (in the hope that at least some of the dust will have settled by then.) 6a

Please reply promptly. The strength and existence of USING depend on you. 7

Nancy and Dave 8

The Future of USING

(J30845) 6-JUN-74 08:54; Title: Author(s): Nancy J. Neigus, David H. Crocker/NJN DHC; Distribution: /USING(IMPORTANT Please read and reply promptly); Sub-Collections: NIC USING; Clerk: NJN;