Teletype changes relative to the PDP-10 transfer

1

Dataset numbers 2

The telephone numbers on our datasets will be changed on December 10. This is being ne so that all lines can work in rotary when all are on the PDP-10. (On the 10 it will not be necessary to get to a particular teletype port to connect to a particular job, and all ports will have variable speed.) 2a

The new numbers are as follows: 2a1

New	Old	Machine	Speed	Channel	
329-8220	327-8683	940	30	17	2a1b
329-8221	327-8751	940	10	11	2a1c
329-8222	327-6303	940	30	18	2a1d
329-8223	327-8795	PDP-10	Variable		2a1e

2a1a

Variable character rates 3

The variable speed feature on the PDP-10 will operate as follows: 3a

The channel speed for all local (directly connected) terminals will be determined by the cable connector on the terminal. Speed will be clearly marked on the connector. 3a1

Terminals such as Terminet or Texas Instruments with variable speed must use a different cable for each speed. 3a1a

For datasets, when the number is first called the channel will be in the "reset" state. That is, nothing can be transmitted or received. After the handset is placed in the coupler (carrier communication established) digits may be dialed on the calling telephone to select the desired character rate. 3a2

Each time a "1" is dialed the channel is stepped successively through four states -- three speeds, 10, 15, and 30, and a reset state. 3a2a

For example: If a single "1" is dialed the channel

will operate at 10 cps. If three "1"s are dialed the channel will be operating at 30 cps. 3a2a1

At any time during connection to the dataset a digit may be dialed to step to the next state. 3a2a2

Don't forget: 3a2a3

To go from a higher speed to a lower speed you must go through the reset state. 3a2a3a

If the connection is broken, when the number is redialed the channel is again in the reset state. 3a2a3b

Local channels 4

Local channels (the office and work area teletype connectors) may be patched to either the PDP-10 or the 940. There is a new teletype patch panel for the PDP-10 located in front of the disc. 4a

Please do not change patching unless absolutely necessary 4a1

During normal hours ask Ed VanDeRiet or Martin Hardy for any needed changes. 4a2

If you must change patching, the numbers on the new panel correspond to the numbers on the teletype outlets. A table on the side of the patch panel rack relates these numbers to actual outlet locations (i.e. offices). 4a3

Most teletype connectors appear on both the new PDP-10 patch panel and on the old 940 panel. Things do not work well if a teletype is patched to both at once. 4a4

Local teletypes are being be modified to operate with the PDP-10. Teletypes so modified will show a number on the teletype connector indicating the character rate (i.e. all 33s will show 10). 4b

Once modified it is relatively easy to switch a teletype back and forth between the PDP-10 and the 940, but internal changes are required. Please ask Ed or Martin to do this for you. 4b1

```
':5230', 12/01/70 0807:18 JGN ; .DPR=1; ':TELETYPES', 11/30/70 1537:56 WKE
;  .DPR=0;
```

ARC RADC REPORT INTEGRATED OUTLINE
25 Nov 70 DVN

Current outline for February ROMAC Report (11/17) 1

I Abstract (1page) (DvN) 1a

II Prefactory Credits (1page) (DvN) 1b

III Summary 1c

History and Philosphy (2 pages; brief rough = <1 page) (DvN) 1c1

Highlights of the contrat year (6 pages; brief rough= 1 page) (DvN) 1c2

Future plans (1 page; page; brief rough =<1page) (DCE) 1c3

IV Working in the ARPA Net. 1d

NIC (13 pages; brief rough = 1 page) (JCN) 1d1

Goals, strategy, and philosophy 1d2

Establish first contacts 1d3

Build and maintain collection 1d4

Establish Network Dialogue 1d5

Activity support surveys 1d6

Stimulate dialogue 1d7

Use of network facilities (3 pages; brief rough = 1 page) 1d8

In change to the PDP 10 1d8a

Use of PDP 10's on the net to bootstrap the compiler (WHP) 1d8a1

Running TODAS on another PDP 10 on the net (WHP) 1d8a2

Connection to the network (3 pages; brief rough = 1page) (WKE) 1d9

Hardware Connection 1d9a

ARC RADC REPORT INTEGRATED OUTLINE
25 Nov 70 DVN

Software 1d9b

V Changing from XDS 940 to PDP-10 (37 pages) 1e

Hardware (WKE) (10 pages;brief rough =2 pages) 1e1

Reasons for the chanege 1e1a

The PDP-10 facility 1e1b

Considerations for Design of the facility 1e1c

Adapting non-DEC Equipment 1e1d

Addition of the BB&N Paging Box 1e1e

Monitor + Exec. (JTM)(5 pages; brief rough = 1page) 1e2

Related to Hardware 1e2a

Not Related to Hardware 1e2b

Compiler (DIA+WHP) (7 pages; brief rough = 1 page) 1e3

Convert compiler to produce PDP-10 code on 940 1e3a

Use of Network to bootstrap compiler 1e3b

Rewrite NLS to new compiler language 1e3c

Paging modifications 1e3d

NLS/TODAS (CHI) (10 pages; CHI has some files that will serve as brief rough) 1e4

Getting a version runningon the network 1e4a

Augmentation system adaptation to 10 1e4b

Making the Tenex monitor run with our hardware 1e4c

VI New Tools 1f

Hard ware (WKE) (13 pages; brief rough = 2 pages) 1f1

Univac drums 1f1a

ARC RADC REPORT INTEGRATED OUTLINE
25 Nov 70 DVN

Remote terminals of various kinds 1f1b

Imlac 1f1c

Higher level processes(WLB) (5 pages; brief rough=1page) 1f2

Content Analyser 1f2a

Analyser formatter 1f2b

Collector - Sorter 1f2c

New features in executable text 1f2d

Core NLS---design philosophy (WSD) (3pages; brief rough=<1page) 1f3

New Features for Users 1f4

Journal (JCN) (3 pages; brief rough = 1 page) 1f4a

Concept 1f4a1

What We Did 1f4a2

Next Steps 1f4a3

Relations to Network Dialogue Subsystems 1f4a4

Comments 1f4a5

Mail (WSD) (3pages; brief rough = 1 page) 1f4b

New NLS features (CHI) (7 pages; CHI has a file that will serve for a brief rough) 1f4c

Calculator 1f4c1

Several others (see CHI's file) 1f4c2

Design Team Planning 1f5

Central Planning File 1f5a

Individual files for each task, plan, design, schedule 1f5b

ARC RADC REPORT INTEGRATED OUTLINE
25 Nov 70 DVN

Automatic collecton and integration of schedules 1f5c

When many people plan 1f5d

Updating 1f5e

Remoter terminal experiments (WKE) (2 pages; brief rough =<1page) 1f6

Plans for the future (DCE) (15 pages; rough brief =3 pages) 1g

Glossary (DvN) (2 pages) 1h

References JCN) (2 pages) 1i

Bibliography (JCN) (2pages) 1j

Emphasis in this report should be in this order: first on what we need second on what the people who will use NIC need, third on what NET experimentos need, fourth on what other developers of interactive systems can use 1k

Hence, for example, descriptions of a given use of the NET in converting to the new machine should be most developed under section 6 rather than in connection with the NET. 1k1

Page assignments are tentative. Please find contradictions, redundancies etc and tell me about them. 1l

Report schedule 2

Month	>!	NO	NO	NO	NO	DE	DE	DE	DE	JA	JA	JA	JA	JA
Day	>!	07	14	21	28	05	12	19	26	02	09	16	23	30

F2a 02b

d=draft w=write r=review x=other activity CAP MEANS D2c

Planning and start !xxx 2d

Sec.2 Pref credits ! 2e

DVN >! www 2e1

Sec.6 New features ! 2f

ARC RADC REPORT INTEGRATED OUTLINE
25 Nov 70 DVN

```
      WKE  hdwe        >!      ddd D   wwwwwwrrwww                  2f1
      CHI  soft        >!      ddD     wwwwwwrrwww                  2f2
      WSD core NLS+    >!      ddd     wwwwwwrrwww                  2f3
      JCN jou+B-line   >!      dddD    wwwwwwrrwww                  2f4
      WLB  HLP         >!      ddd D   wwwwwwrrwww                  2f5
   Sec.5 XDS940 +PDP10 !                                            2g
      WKE  hdwe        >!          ddD       wwwwwwrrwww           2g1
      JTM monitor      >!          ddD       wwwwwwrrwww           2g2
      CHI NLS/TODAS    >!          ddD       wwwwwwrrwww           2g3
      DIA+WHP compr    >!          ddd       wwwwwwrrwww           2g4
   Sec.1 Abstract      !                                            2h
      DVN              >!          ddd       wwwwwwrrwww           2h1
   Sec.4 ARPA Network  !                                            2i
      WKE connection   >!        dddD      wwwwwwrrwww             2i1
      WHP Net fac use  >!        ddd       wwwwwwrrwww             2i2
      JCN  NIC         >!        dddD      wwwwwwrrwww             2i3
   Sec.3 Summary       !                                            2j
      DCE              >!                ddd  wwwwwwrrwww         2j1
      DVN              >!                ddd  wwwwwwrrwww         2j2
   Sec.7 Future plans  !                                            2k
      DCE              >!                ddd  wwwwwwrrwww         2k1
   Sec.8 Glosssary     !                                            2l
      DVN              >!             wwwrw                       2l1
```

ARC RADC REPORT INTEGRATED OUTLINE
25 Nov 70 DVN

```
   Sec.9 References     !                                                2m
       JCN             >!                  wwwrrwww                     2m1
       DVN             >!                  wwwrrwww                     2m2
   Sec10 Bibilo         !                                                2n
       JCN             >!                  wwwrrwww                     2n1
       DVN             >!                  wwwrrwww                     2n2
   DCE approval         !                        xxx                    2o
   SRI edit+approval    !                           xxxxxx              2p
   Mail good draft      !                           Feb     4th>2q
          Day           ! 07 14 21 28 05 12 19 26 02 09 16 23 30    02r
          Month         ! NO NO NO NO DE DE DE DE JA JA JA JA JA     F2s
   Patterns:   /"CHI" OR"Sec"/;                                        2t
Links to inprocess report sections:                                    3
   Sec.1 Summary          (nouhuys,summary:zxbnZD)                     3a
   etc etc                                                             3b
```

':523|', |2/0|/70 08|8:38 JGN ; .DPR=|; ':RPLAN', ||/25/70 |056:59 DVN ;
.DPR=0;

HIGHER LEVEL PROCESSES -- TABLE OF CONTENTS

WLB 12/02/70
TABLE OF CONTENTS

HIGHER LEVEL PROCESSES -- TABLE OF CONTENTS

TABLE OF CONTENTS 1

I. CONTENT ANALYZER 2

A. Introduction 2

B. Pattern-Specification Language 2

C. Procedure for Using Content Analyzer 11

II. ANALYSER-FORMATTER 14

A. Introduction 14

B. Brief Notes about Analyser-Formatter Syntax . . . 14

C. Compiling and Executing User-Written Programs . . 18

D. Examples 19

III. COLLECTOR-SORTER 22

A. Introduction 22

B. Procedures for Using the Collector-Sorter 22

C. Collector-Sorter Commands 22

IV. NEW FEATURES IN EXECUTABLE TEXT 25

A. New Syntactic Elements 25

B. Imbedded Commands 25

C. New TODAS Commands 27

D. New Analyser/Formatter Features 29

E. Using the New Features 29

I. CONTENT ANALYZER

7

A. Introduction 8

The content analyzer feature of NLS/TODAS permits the user to write a string of text which specifies (in a special language) a pattern of content. After the pattern has been compiled, when the content analyzer is turned on (by the VIEWSPEC parameter "i"), only statements which meet the content specification will be displayed by NLS, printed out by "Print" commands, outputt by "Output Device" commands, or affected by "Substitute" commands. 8a

The pattern specified may be a simple one -- e.g., it may specify a string of characters that must appear somewhere in each statement tested; or it may be complex -- e.g., it may specify a string, to be followed within a given number of words by another specified string, in statements which were created after a certain date by a certain author, and not containing some third specified string. 8a1

The language for specifying content patterns is simple and easy to use for simple cases, but more exacting for complex cases. 8a2

B. Pattern-Specification Language 9

a. The Process of Searching a Statement 9a

When the content analyzer is turned on, each statement in the file is searched, character by character, for the content specified in the pattern. Normally, the search begins with the first character, but it is possible to cause the search to proceed backwards. 9a1

The analyzer uses a pointer to keep track of the search. The pointer always indicates which character is to be examined NEXT, unless something in the pattern causes the pointer to be moved first. 9a2

At any given moment in the search process, the analyzer is searching for one of four types of content entity: 9a3

A literal string of characters, such as "abcd" or "13-x" or "ed Mat" or "memory." 9a3a

A string of "character-class variables"; these are explained in detail further on. A string of character-class variables might specify "three digits, one after another," or "two letters, followed by any number of spaces, followed by three to five letters or digits." 9a3b

The date associated with the statement. (This is not normally printed, but every statement bears the date on which it was created or most recently modified.) 9a3c

The initials associated with the statement. (This is not normally printed, but every statement bears the initials of the user by whom it was created or most recently modified.) 9a3d

All of the more complex analysis is achieved by moving the pointer according to the logic of the pattern specification. 9a4

For example, if the analyzer is to start at a given point and find either String A or String B, it first looks for string A; if String A is not found, the pointer is returned to the starting point, and a search is made for String B. 9a4a

b. Basic Elements 9b

Every pattern ends with a semicolon. If the pattern is used as part of a link, it must also begin with a semi-colon. 9b1

Every pattern is made up of one or more of the basic entities listed above, combined by operators. 9b2

If the element being searched for (or some part of it) is to be found anywhere after the point in the statement where the search begins, the corresponding pattern element is enclosed in square brackets; otherwise it must be the first thing found. 9b3

A string of characters specified as content is enclosed in quotation marks. For convenience, if the string consists of only one character, it may be preceded by an apostrophe and the quotation marks omitted. 9b4

Examples 9b4a

;/"memory"/; This pattern will select only those statements containing the word "memory" at any point. 9b4a1

;"inside"; This pattern will select only those statements beginning with the word "inside". 9b4a2

;/'3/; This pattern will select only those statements containing the character "3" at any point. 9b4a3

Patterns like those shown in the examples above may be strung together; the significance of this is that one item is to be found after the one specified ahead of it. 9b5

Examples 9b5a

;/"abc""def"/; This pattern specifies that the string abc immediately followed by the string def must appear somewhere in the statement. The pattern ;/"abcdef"/; is exactly equivalent. 9b5a1

;/"abc"//"def"/; This pattern specifies that the string abc is to be found anywhere in the statement, and anywhere after the "c" the string def is to be found. 9b5a2

c. Character-Class Variables 9c

The character-class variables are as follows: 9c1

L means any letter 9c1a

D means any digit 9c1b

LD means any letter or digit 9c1c

PT means any printing character (any character except space, tab, and carriage return) 9c1d

SP means a space 9c1e

TAB means a tab 9c1f

CR means a carriage return 9c1g

NP means any nonprinting character (space, tab, or carriage return) 9c1h

CH means any character at all. 9c1i

Note that these must always be capitalized. 9c1j

Examples 9c2

;/'.LLL'=D';/; This pattern will select only those statements containing (anywhere) the following content: a period immediately followed by three letters, immediately followed by an equals sign, immediately followed by a digit, immediately followed by a semicolon. 9c2a

;"abcd"SPL D; This will select only those statements beginning with the following content: the string abcd immediately followed by a space, immediately followed by any letter, immediately followed by any digit. 9c2b

Note that a space is necessary between the L and the D because of a possible ambiguity: The pattern ;"abcd"SPLD; would mean "the string abcd immediately followed by a space, immediately followed by any letter or digit," because LD means any letter or digit. 9c2b1

d. The Dollar Sign (Arbitrary-Number Construct) 9d

The arbitrary-number construct, in its most general form, is m$n. The meaning is "any number from m to n of occurrences of the following entity." 9d1

This pattern is executed by scanning through ALL characters which are of the appropriate class and then comparing the number found with the specified limits -- in other words this matching process is always deterministic, and the limits m and n are absolute. 9d2

Example 9d2a

The pattern ;5$11LD; specifies that each statement tested must begin with five to eleven letters and/or digits. 9d2a1

A statement beginning with more than eleven or fewer than five letters and/or digits would be rejected by this pattern as would a statement beginning with any character other than a letter or digit. 9d2a1a

The m or the n, or both, may be omitted; their assumed values in this case are m=0, n=1000. For all practical purposes, then, the default value of n is "any arbitrary number," since it is very unlikely that any entity will occur 1000 times consecutively. 9d3

Examples 9d3a

The pattern ;/7$D1$12L$5NP/; specifies that each statement tested must contain the following: seven or more digits immediately followed by one to twelve letters, immediately followed by zero to five nonprinting characters. 9d3a1

The pattern ;2$/"abc"/; specifies that each statement tested must contain two or more occurrences of the string abc. 9d3a2

e. Grouping by Parentheses 9e

Parentheses may be used as they are in algebra to group elements. The specifications found within the parentheses are then treated as a single entity for logical purposes. 9e1

Example 9e1a

;/3$4(DSPL)1$2NP/; This pattern specifies that each statement tested must contain the following: three or four occurrences of the string (digit space letter), immediately followed by one or two nonprinting characters. 9e1a1

If the parentheses were not used, the 3$4 construct would apply only to the D. 9e1a1a

The square brackets have the same grouping effect as parentheses; however, they are not interchangeable with parentheses because they also mean that the enclosed pattern may be found anywhere after the starting point. 9e2

f. Operators 9f

The operators used for combining entities are as follows, in order of decreasing precedence (see note on precedence, below): 9f1

- (minus sign): This indicates negation. Thus -LD means a character which is not a letter or a digit. 9f1a

Example: ;/"abc"-SP/; This pattern specifies that each statement tested must contain the string abc immediately followed by some character which is not a space. 9f1a1

(space): This indicates concatenation. Thus ;"abc" "xyz"; specifies that the string abc must occur and must be immediately followed by the string xyz. 9f1b

The space may be omitted unless it is necessary to prevent ambiguity. Thus ;"abc" "xyz"; could also be written ;"abc""xyz"; 9f1b1

/ (slash): This indicates alternation. Thus SP/TAB means a character that may be either a space or a tab. 9f1c

Example: ;1$SP/2$3PT; This pattern specifies that each statement tested must begin with either one or more spaces, or two or three printing characters. 9f1c1

NOT: This indicates negation, and is the same as the minus sign except for lower precedence. 9f1d

AND: This is logical intersection. 9f1e

The action of the AND is to return the pointer to the beginning of the search that has just been completed. 9f1e1

Example: The pattern ;[("abc"]AND[("xyz"]; causes each statement to be searched first (from the beginning) for the string abc; then, if it is found, the statement is searched again from the beginning for the string xyz. Each statement passing the test will contain both strings, but the order in which they occur will be irrelevant. 9f1e1a

Note that this is different from the pattern ;[("abc"][("xyz"]; because if the AND is not used, the second search is not made from the beginning but from the point just after the end of the first search. Each statement that passes the test will then contain both strings, but the string xyz must be somewhere after the string abc. When the AND is used, this restriction will not apply. 9f1e1a1

Note also that the pattern ;[("abc"AND"xyz"]; is meaningless: it specifies a string that is both "abc" and "xyz". 9f1e1a2

OR: This is the same as the slash sign except for the lower precedence. 9f1f

Note that NOT, AND, and OR must always be capitalized. 9f1g

Note on Precedence of Operators: As used here, "high precedence" means that when the pattern is parsed, the higher-precedence operators are used first in grouping the elements of the pattern. Thus a high-precedence operator has low "binding power." 9f2

Example: Consider a pattern of the form ;a AND b OR c/-d AND NOT e f; where a, b, c, d, e, and f stand for pattern elements such as quoted strings or character-class variables. 9f2a

This is grouped as follows: 9f2b

The minus sign has the highest precedence, so that we have ;a AND b OR c/(-d) AND NOT e f; 9f2b1

Next is concatenation, so we have ;a AND b OR c/(-d) AND NOT (e f); 9f2b2

Next is the slash, so we have ;a AND b OR (c/(-d)) AND NOT (e f); 9f2b3

Next the NOT, giving ;a AND b OR (c/(-d)) AND (NOT (e f)); 9f2b4

Finally, the AND gives ;(a AND b) OR ((c/(-d)) AND (NOT (e f)));. 9f2b5

g. Dates and Initials 9g

The dates and initials associated with each statement may be tested with the constructs .SINCE, .BEFORE, .INITIALS=, and .INITIALS#. (The symbol # is used to mean "not equal.") 9g1

The .INITIALS construct requires the following format: 9g2

.INITIALS=ABC where the string ABC is a user's initials (three initials must be given). 9g2a

The .SINCE and .BEFORE constructs require the following format: 9g3

.SINCE (68/10/12 13:14) where 68 is the year, 10 is the month, 12 is the day, 13 is the hour, and 14 is the minute. The time may be eliminated by using 0:0. 9g3a

Examples 9g3b

;.BEFORE (67/3/22 15:15) AND .SINCE (67/1/12 12:00); This pattern will select only those statements bearing dates between noon of 12 January 1967 and 3:15 PM of 22 March 1967. 9g3b1

;.SINCE (68/10/10 0:0) AND .INITIALS#DGC; This pattern will select only those statements bearing dates later than 10 October 1968 and not bearing the initials DGC. 9g3b2

h. Special Control of Search 9h

The position of the search pointer can be stored and set, and the direction of search can be controlled, in order to achieve complex effects. These effects also involve the use of the IF construct (described further on), and the possibilities have been explored only superficially at present. It should be possible to create pattern expressions of great complexity which would resemble sophisticated data-processing or information-retrieval programs, but at present the techniques have not been worked out. 9h1

The position of the pointer may be stored in any one of nine buffers, P1 ... P9. This is done by writing ↑Pn, where n is some digit from 1 to 9. 9h1a

The stored value in the buffer can then be decremented by writing ←Pn. The reason for doing this is that when the analyzer has found some entity, the pointer is moved to the next character position; in order to store the value of the last character actually searched, then, it is necessary to write ↑Pn←Pn. 9h1a1

The search pointer can then be set to the value in a buffer by writing Pn. 9h1b

The search pointer can also be set to the beginning or end of a statement by writing SF(Pn) for the beginning and SE(Pn) for the end. 9h1c

Note that SF and SE are functions which require a buffer value as argument; buffer values are not reinitialized after a statement has been scanned but continue to indicate the same character in the statement they were originally set to. Thus it is possible for a search to cover more than one statement. 9h1c1

The normal direction of scanning may be reversed by writing a less-than sign (<) and returned to the forward direction by writing a greater-than sign (>). 9h1d

The left-arrow (←) used for decrementing a buffer value will increment it instead if the current scan direction is backward. Thus the effect will always be the same -- the buffer value will indicate the character just scanned. 9h1d1

Example 9h1d1a

;↑P1 SE(P1) < $NP -'.; This pattern causes statements to be searched backwards from the end. Only statements whose last printing character is not a period will pass the test. 9h1d1a1

The construct "↑P1" at the beginning of the pattern causes the current pointer position (which indicates the beginning of the statement) to be stored. This is simply for the purpose of having an argument for the "SE(P1)" construct, which causes the pointer to be positioned to the end of the statement. The less-than sign then causes the scan to proceed backwards; any number of nonprinting characters will be permitted, and then a character which is not a period is specified. 9h1d1a1a

i. The BETWEEN Construct 9i

The BETWEEN construct permits the user to specify that search for certain elements be restricted to the text between two elements already located. 9i1

The syntax is: 9i2

"BETWEEN" Pn Pm '(<pattern> ') 9i2a

Example: 9i3

;[['*] ↑P1 ['*] ↑P2 ←P2 ←P2 BETWEEN P1 P2 ("Author" [" Jones "])]; 9i3a

This pattern matches statements such as: 9i3b

*Title More Computer Gobbledygook *Authors Brown Jones Smith *Date November 1970 * 9i3b1

Note that if the pattern had been written without the outer set of square brackeds, e.g.: 9i3c

;['*] ↑P1 ['*] ↑P2 ←P2 ←P2 BETWEEN P1 P2 ("Author" [" Jones "]); 9i3c1

it would only match statements in which the string "Authors" appeared after the first '* in the statement. 9i3d

C. Procedure for Using Content Analyzer 10

TODAS 10a

There are three stages to using a pattern: (1) writing the pattern, (2) causing the pattern to be "compiled," as explained below, and (3) putting the pattern into effect by turning the content analyzer ON with the VIEWSPEC "i". 10a1

Patterns are written in two ways: as part of a link or as part of a command. In either case, the pattern is written as if it were a VIEWSPEC -- i.e., it goes in the VIEWSPEC portion of a link or is typed in when a command is expecting VIEWSPECs. 10a2

If the pattern is written as part of a link, it is "compiled" (i.e., understood by TODAS) when the link is executed by means of an indirect address (see Appendix A). If it is typed in the VIEWSPEC portion of a command, it is compiled as soon as the final CA is given and the command begins executing. 10a3

Only one pattern may be compiled at a time -- i.e., when a new pattern is compiled the previous one is forgotten. 10a3a

If there is a syntax error in the pattern (i.e. if it is not a legal pattern) the message ERROR will be typed and the command will be aborted. The last few characters read by the compiler will be output to the teletype simulation buffer, with the character at which the error was detected being the last character output. 10a3b

Errors are frequently caused by inadvertent omission of some character such as a quotation mark. Another common cause for a syntax error (or a legal pattern that does not work as expected) is an error in the way that parts of the pattern are grouped. In the latter case, the problem may often be solved by insertion of parentheses. 10a3c

When the pattern has been compiled, it will not go into effect unless the VIEWSPEC "i" is placed in effect. When this has been done, TODAS will ignore all statements which do not fit the pattern. 10a4

To summarize, a pattern may be compiled whenever you use a command that allows you to set VIEWSPECs or whenever a link is executed. Once the pattern is compiled and the VIEWSPEC "i" is set, the content analyzer will affect all commands that are affected by VIEWSPECs (Print, Output Device, Substitute) until the VIEWSPEC "j" is set or a new pattern is compiled. It affects these commands by testing statements against the pattern; statements that pass the test are handled by the command; those that fail the test are ignored. 10a5

Testing of statements begins with the current statement (the one pointed to by the CSP); other statements are then tested in the order in which they would appear "normally," i.e. with the analyzer off. Any other view specifications which are in effect continue to work; thus if only first- and second-level statements are being printed, output to a device, or substituted, then only first- and second-level statements will be tested by the analyzer. 10a6

Testing continues until all the statements in the part of the file that was specified in the command have been tested. If no statements are found that fit the pattern, the message "EMPTY" is typed. 10a7

NLS 10b

A pattern may be written as text anywhere in a file. A file may thus contain any number of patterns; however, only one pattern may be compiled at a time -- i.e., when a new pattern is compiled the code created by the previous one is lost. 10b1

To compile a pattern, the command Execute Content Analyzer is used. The syntax is 10b2

e c /c/ CA 10b2a

where /c/ means that a character is selected either with the mouse or by means of a pointer call, and CA means that a Command Accept key is struck. 10b3

The character selected must be either the first character of the pattern or a nonprinting character preceding the pattern, with no printing characters intervening. 10b3a

Note that the last part of a pattern may thus be used as a separate pattern, if it is meaningful. 10b3b

When the pattern has been compiled the command feedback line will change from "Content Analyzer" to something else. If the pattern has an error in it which prevents it from compiling, the screen will go momentarily blank with the message "error." The last few characters read by the compiler will be output to the teletype simulation buffer, with the character at which the error was detected being the last character output. 10b4

Syntax errors are frequently caused by inadvertent omission of some character such as a quotation mark. Another common cause for a syntax error or a compiled pattern that does not work as expected is an error in the way that parts of the pattern are grouped. In the latter case, the problem may often be solved by insertion of parentheses. 10b4a

When the pattern has been compiled, it will not go into effect until the view-control parameter "i" is placed in effect. When this has been done, the system will display only statements which fit the pattern. 10b5

Testing of statements begins with the statement currently designated as the display start; other statements are then tested in the order in which they would appear "normally," i.e. with the analyzer off. Any other view specifications which are in effect continue to work; thus if only first- and second-level statements are being displayed, only first- and second-level statements will be tested by the analyzer. 10b6

Statements are tested until the display screen has been filled. If no statements are found that fit the pattern, the screen goes blank with the message "empty" and remains so until the analyzer is turned off or until changed view-control parameters or a new display start make it possible to find a statement that fits the pattern. 10b7

Whenever the display is recreated, the testing process is repeated. Thus if a statement is edited, and the editing changes it so that it no longer fits the pattern, it will disappear from the screen. 10b8

II. ANALYSER-FORMATTER

11

A. Introduction 12

The following is by no means a complete description of the programmed editing (Analyser-Formatter) feature -- no such description has yet been written. 12a

In principle, one can write arbitrarily complex SPL (Special-Purpose Language) programs, compile them with the "Execute Analyzer Compiler" command (see below), and run them with the VIEWSPECs i and 0 to edit statements automatically. Here we merely give a few examples of very simple SPL programs, to indicate the general possibilities. 12a1

In explaining the examples, we assume that the reader is well-versed in the content-analyzer language, which turns out to be a subset of one of the SPL's. 12a2

B. Brief Notes about Analyser-Formatter Syntax 13

(1) A program begins with the word PROGRAM and ends with the word FINISH. 13a

(2) Comments are enclosed between percent-signs. Anything enclosed in percent-signs is ignored in compilation. 13b

(3) Blanks are insignificant except where they are used to prevent ambiguity (as in the content-analyzer language). Statement breaks are insignificant. 13c

(4) The body of the program consists of one or more procedures. A procedure begins with a name in parentheses followed by the word PROCEDURE followed by a semicolon, and ends with the words RETURN END followed by a period. 13d

(5) The bodies of all the procedures consist of "constructions" and other statements. 13e

(6) A construction begins with ":C" and ends with a colon. Its body may contain the following: 13f

(a) Content-analyzer patterns used for testing a statement and setting pointers in it. 13f1

The pattern scan begins at the "current statement position" and goes forward or backward, under program control. 13f1a

The current statement position may be changed by "mentioning" a pointer name -- e.g. :C P2 : -- or by using the SF/SE constructions -- e.g. :C SF(P1) : which sets the CSP to the first character in the statement into which P1 points. 13f1b

Direction of scan may be set to forwards using the pattern element > and to backwards using the pattern element <. 13f1c

At the beginning of execution of each pattern, the system identifier "flag" is set to true. It remains true unless some pattern element is found which cannot be matched in the statement. The value of "flag" may be tested in conditional statements (see below) and may be set arbitrarily using "flag ← 0" or "flag ← 1" (for false and true respectively) -- note that these statements are not in the construction SPL and must not appear within the :C : delimeters (which tell the compiler to switch from one language to another). 13f1d

When any pattern element fails to match, the scan is aborted and the rest of the pattern is ignored -- this can cause strange things to happen if programs are not written very carefully. E.g. consider the pattern
:C ↑P1 SE(P1) < /'*/ ↑P2 ←P2 > :
The intent of this pattern is to make P2 point to the last '* in the statement. Note that if there is no '* in the statement, the part of the pattern to the right of /'*/ will be ignored: since this part of the pattern contains the element which restores the scan direction to forwards, the scan direction will remain backwards until another > is encountered elsewhere in the program. 13f1e

If such a pattern is the last thing in the construction, the terminating semicolon normally used with a pattern is omitted and it is terminated by the same colon that terminates the pattern. 13f1f

(b) An instruction to create a new statement and replace the old statement with it. Such an instruction has the following elements: 13f2

(1) Identification of the statement by reference to a pointer that has been set in it with a content-analyzer pattern, e.g. "ST P1" means "the statement that has Pointer 1 in it." 13f2a

(2) A left-arrow (←), signifying "is to be replaced by." 13f2b

(3) One or more pairlists, separated by commas. Each pairlist specifies a string of text, and the new statement is constructed by taking the pairlists in order and stringing together the corresponding strings of text. A pairlist may be one of three things: 13f2c

(i) A pair of pointers in the old statement, such as "P4 P5". The corresponding text string is the string delimited by these pointers in the old statement. 13f2c1

Instead of pointers, one may use the notations "SF(Pn)" and "SE(Pn)", meaning the "statement front" and "statement end," respectively, of the statement containing pointer n. Thus a pairlist could be "SF(P1) P3", meaning the text from the front of the statement to pointer 3. 13f2c1a

(ii) A literal string of text enclosed in quotation marks ("<string>"). Such a string is simply copied into the new statement. 13f2c2

(iii) One or more of the notations SP for "space," TAB for "tab,", and "CR" for "carriage return." Thus, CR CR TAB is a valid pairlist, and the corresponding string would consist of two carriage returns followed by a tab. 13f2c3

(7) Other statements of general utility include: 13g

(a) RETURN Statements 13g1

Syntax: RETURN 13g1a

Semantics: Causes a return to the calling procedure. 13g1b

(b) Assignment Statements 13g2

Syntax (example): flag ← 1 13g2a

Semantics: Sets the value of the identifier "flag". 13g2b

(c) Conditional statements 13g3

Syntax: IF flag THEN <block-> ELSE <block> ENDF 13g3a

Semantics: The ELSE part is optional. If flag=1, then <block-> is executed; otherwise, <block> is executed. <block> consists of one or more statements separated by semicolons; if a statement ends with a colon or an ENDF, then the semicolon is omitted. <block-> is the same as <block> except that none of the statements may be conditionals. 13g3b

(d) SEND statements 13g4

Syntax: SEND 13g4a

Semantics: Normally, each time the sequence generator calls an analyser-formatter program, it passes a single statement to the program as input and expects a single statement back as output. The SEND statement allows a program to generate more than one statement as output. 13g4b

Whenever a SEND statement is executed, the current version of the statement passed by the sequence generator is sent back as output. This version is either the original (if no reconstruction has been done) or the version most recently constructed by the program. After execution of the SEND, additional analysis and construction can be performed. 13g4b1

If the sequence generator is generating statements for output (e.g. to the printer) or for an Execute Merge operation, all versions SENT by the program will be used; if it is merely processing a file "internally" (i.e. with viewspecs iO), then only the final version of each statement SENT will actually remain in the file. 13g4b2

NOTE: Whenever a reconstruction is performed, all pointers in the reconstructed statement still point to that statement, but their character position values are meaningless. This means that if several reconstructions are to be done (with SENDs between each one), it might be desirable to use a copy of the statement for the analysis parts so that pointer values can be preserved. This can be accomplished by copying the original statement into the origin statement of the file, taking account of the fact that when NLS/TODAS is started, all pointers initially point to the file origin statement. 13g4b3

Note that pointers are never reinitialized to the origin, so that once a pointer has been set to point to another statement (either by the program or by internal NLS/TODAS routines), it can not be used for this purpose. Pointers P1-P8 are used by NLS/TODAS editing routines and can be assumed to point to the origin only if no editing has been done prior to use of the analyser-compiler program. Pointers P9-P30, however, are not so used, and consequently are "safe" in this respect (P10-P30 are currently (11/25/70) available only in the experimental system). 13g4b3a

C. Compiling and Executing User-Written Programs 14

(ea) Execute Analyzer Compiler 14a

Syntax: e a CA 14a1

Semantics: This command assumes that a valid SPL program is in the file currently loaded, with the PROGRAM statement as the display start in NLS or designated by the current statement pointer in TODAS. The program is compiled and can then be put into effect by use of the "i" VIEWSPEC. If the program fails to compile for any reason, an error message is displayed. If the user then gives an "Execute Quit" command to go back to the Exec, he will see more detailed error information. 14a2

NOTE: If you have used the Output Processor before trying to compile, you may be flashed an error message with no comments present at the Exec. This is a bug in NLS/TODAS; to get your program compiled you must go to the Exec, do a RESET, and then fire up NLS/TODAS again. 14a2a

(eb) Execute Big Analyzer Compiler 14b

Syntax: e b CA 14b1

Semantics: This command is the same as Execute Analyser Compiler except that a larger buffer is provided for the compiled code, permitting larger programs to be compiled. 14b2

D. Examples 15

1. Delete Leading Blanks from Statements 15a

PROGRAM %deletes leading blanks from statements% 15a1

```
(dls) PROCEDURE; :C 1$NP ↑P1: IF flag THEN :C ST P1 ← P1
SE(P1): ENDF RETURN END.                                    15a1a

FINISH                                                      15a1b
```

Explanation 15a2

The body of procedure "dls" begins with a construction which contains a content-analysis pattern. This pattern looks for leading blanks; if it finds at least one, it sets pointer 1 to the first nonblank character and sets the flag TRUE; otherwise it simply sets the flag FALSE. 15a2a

Next comes a conditional clause, which will be excuted only if the flag is TRUE. The conditional contains a single construction with no ELSE part. 15a2b

This construction causes the statement in which Pointer 1 is set ("ST P1") to be replaced ("←") with a new statement made up of the string specified in the single pairlist "P1 SE(P1)". This string is found in the old statement, from Pointer P1 (the first non-blank character) to the end of the statement. 15a2b1

2. Append a Message to Each Statement 15b

PROGRAM %appends a message to each statement% 15b1

```
(addmess) PROCEDURE; :C ↑P1; ST P1 ← SF(P1) SE(P1), "
This is an appended message": RETURN END.                   15b1a

FINISH                                                      15b1b
```

Explanation 15b2

The body of procedure addmess contains a single construction. This construction contains a content-analysis pattern (terminated with a semicolon) and an instruction for making a new statement. 15b2a

The content-analysis pattern does nothing except place a pointer in the statement. 15b2a1

The instruction has two pairlists. The first is "SF(P1) SE(P1)", and specifies the string running from the front to the end of the old statement. The second pairlist is the text in quotation marks. Note that the two pairlists are separated by a comma. 15b2a2

3. Retrieve Author and Date from a Bibliographic Reference 15c

PROGRAM %retrieves author and date from a bibliographic reference% 15c1

```
(restruc) PROCEDURE; :C /',/ ↑P1←P1 SE(P1) < /DDDD/
↑P2←P2 > /DDDD/ ↑P3←P3: IF flag THEN :C ST P1 ← SF(P1)
P1, SP, P2 P3: ENDF RETURN END.
```
15c1a

```
FINISH
```
15c1b

Sample Statement Before Processing 15c2

Fansome, A. O., "Omphaloskepsis in Modern Systems Analysis" (Golem Press, Bethesda, Md. 1969). 15c2a

Sample Statement After Processing 15c3

Fansome, 1969 15c3a

Assumptions 15c4

The author's name is assumed to be the first thing in the statement and to have a comma at the end. The date is assumed to be the last string of four digits in the statement. 15c4a

Explanation 15c5

The body of procedure restruc begins with a content-analysis construction. 15c5a

The content-analysis pattern searches for the first comma in the statement and fixes Pointer 1 on it. It then goes to the end of the statement and scans backwards to find the last string of four digits, fixing Pointer 2 on the front of this string. Next it scans forward to the end of the string of four digits and fixes Pointer 3 there. If either the comma or the string of four digits is not found, the flag is set FALSE; otherwise it is TRUE. 15c5a1

Next comes a conditional, containing a single construct which contains an instruction for making a new statement; this will be executed only if the flag is TRUE. 15c5b

The instruction has three pairlists. 15c5b1

The first is "SF(P1) P1", and specifies the string running from the front of the old statement to Pointer 1 (i.e., to the first comma). This string is assumed to be the author's name. 15c5b1a

The second is "SP", meaning a space. 15c5b1b

The third pairlist is "P2 P3", meaning the string from Pointer 2 to Pointer 3 (i.e., the string of four digits). This string is assumed to be the date. 15c5b1c

III. COLLECTOR-SORTER

16

A. Introduction 17

The collector-sorter (CS) is a program which operates on one or more NLS/TODAS files to extract statements passing some content analysis test, possibly reformatting them in the process, and possibly sorting the collected statements with respect to a specified "key". The collected statements are placed in one or more files named *1, *2, *3, ..., where * stands for a user specified string. The source files may be structured, but the collected (output) files are always single leveled. 17a

B. Procedures for Using the Collector-Sorter 18

Start by preparing a "control file" (optional) which may contain an analyser-formatter program and/or a statement consisting of a list of colon file names (with user names if necessary). 18a

If you wish to sort, the AF program should place the "keys" -- the string(s) on which you wish to sort -- at the front of each statement, enclosed within @'s (with individual keys, if there are more than one, separated by @'s). 18a1

The file names in the list statement are seperated by spaces, the colons are optional. 18a2

Compile the analyser-formatter program. 18b

Type "E X CA" for Execute Colsort. (It will respond by asking for "Device" as in Todas, then will type a "-" indicating it is ready for commands.) 18c

You may then give any of the commands described below. The order in which you give them -- up to the "Go" command -- is unimportant. 18d

C. Collector-Sorter Commands 19

File list 19a

'F <statement number>/(SPACE <file list>) CA 19a1

Specifies list of files to be processed by Colsort either in text of statement in control file or by typing in text string. 19a2

Output prefix 19b

'O <statement number>/(SPACE <string>) CA 19b1

Specifies prefix to be used in naming output files. 19b2

Sort 19c

'S ('Y/CA/'N) 19c1

Sets or resets flag which determines whether the collected file is to be sorted. 19c2

Delete keys 19d

'D ('Y/CA/'N) 19d1

Sets or resets flag which determines whether sort keys are to be deleted from the collected file during the output phase. 19d2

Length 19e

'L ('Y/CA/'N) 19e1

Sets or resets flag which determines whether alphanumeric sort keys will be sorted by arithmetic order. If flag is set, sorting will be like (1,2,3, ... ,9,10,11, ,,,); otherwise, it will be like (1,10,11, ... , 2,21,22, ...). 19e2

Viewspecs 19f

'V <string> CA 19f1

Specifies viewspecs to be used during the collection phase. If Analyser-Formatter is being used, this is the time to turn on viewspecs i and O. 19f2

Go 19g

'G CA 19g1

Causes Colsort execution to commence. When processing is completed, you should be careful to turn viewspec i off before returning to NLS/TODAS in order to avoid crapping up the file you have so carefully constructed. 19g2

Execute Quit 19h

'E 'Q CA 19h1

Returns user to NLS/TODAS. 19h2

if Colsort has successfully produced one or more output files, control is returned to NLS/TODAS with first output file loaded for inspecton. 19h3

IV. NEW FEATURES IN EXECUTABLE TEXT

20

A. New Syntactic Elements 21

TEXTADDR ::= STMTAD/T-STRING 21a

T-STRING ::= '(PTR PTR ') 21b

PTR ::= 'P NUMBER 21c

B. Imbedded Commands 22

These are commands which will be recognised by the input routines, rather than the Todas (or COLSORT) command parser. 22a

This meas that they may appear anywhere in an executeable text program, and do not affect the specification of the command currently being processed by Todas. 22a1

It is additionally true that any of the characters used for the imbedded commands must be preceded by an ! if they are to be used as literal input during the execution of an executable text program. 22a2

Subroutine Call 22b

Syntax: '+ TEXTADDR CA 22b1

This causes the location in the Executable Text program currently being executed to be saved (on a stack), and the text indicated by the TEXTADDR to be executed. 22b1a

When the subroutine program has finished (i.e. when it has issued a return command or the end of the text is encountered), the calling executable text program is continued at the next instruction. 22b1b

Since there is only one buffer available for storing text which is not being executed directly from the statement, a restriction is placed on the subroutine call, which requires that subroutines be executed from a statement, and never from the buffer. 22b1c

It is concievable that this restriction is unreasonable, in which case it would be possible to provide another subroutine call which would allow the executing of subrouines from the buffer, and maybe leave the inherent danger to be resolved by the user. 22b1c1

Return 22c

Syntax: '\ / ENDCHR 22c1

This causes a return to be made from a subrouutine. If the command is executed from a program which is not a subroutine, then it terminates the execute text sequence. 22c1a

Go To text program 22d

Syntax: '← TEXTADDR CA 22d1

This is in essence a branch to another execute text program. 22d1a

No return is possible, and it is assumed that the new executable text program is to be executed in the same mode as the current one, that is, if the current executable text program is being executed from the statement, then the new one will be, and if the current is being executed from the buffer, then the new will be executed from the buffer. 22d1a1

Feedback control 22e

Syntax: '&/'$ 22e1

The '$ command cuses the feedback parameter to be set to 50, while the '& command causes it to be set to 0 (which is the initial setting). 22e1a

Switch to User for input 22f

Syntax: '↑ (DIGIT/CHARACTER) 22f1

Ths allows thhe executable text machinery to request that input be read from the user's terminal. 22f1a

If the '↑ is followed by a digit, then that number of characters is read. 22f1b

If then '↑ is followed by anyother character (including the digit '0), the input is read from the user unitl that character is encountered. 22f1c

The character which serves as a terminator is discarded. 22f1c1

C. New TODAS Commands 23

Execute Text 23a

Syntax: 'E 'T TEXTADDR CA 23a1

This is identical to the current execute text command, except that a TXTADDR may be used in place of a STMTAD. 23a2

Execute Statement 23b

Syntax: 'E 'S TEXTADDR CA 23b1

This command is like execute text, except that the text is not loaded into the executable text buffer before execution. 23b2

This means that the limit on the length of an executable text statement is the maximum length of a statment. 23b3

It also means, however, that the integrity of the file from which the text is being executed must be maintained. 23b4

Execute Status 23c

Syntax: 'E 'Z CA 23c1

The syntax of this command is changed to allow for the Execute Statement command. 23c2

Execute Pointer Set -- Not done yet (11/24/70) 23d

Syntax: 'E 'P NUMBER (SF/SE) '(STMTAD ') CA 23d1

This command allows the user to identify a T-Pointer with a statement front or end at the command level (without entering the conan). 23d2

The NUMBER is th number of the pointer being set (it may range from 0 to 30). 23d3

The T-pointer may then subsequently be accessed by conan programs. 23d4

Execute Define Error or CD Targets 23e

Syntax: 'E 'D ('E/'C) TEXTADDR CA 23e1

This command allows the user to identify an executable text string to be executed upon the occurrence of a CD or an error in the execution of a command. 23e2

There probably needs to be some indication of where the error/CD occurred, but it is not immediately obvious what this should be, so lets this defer this question a bit. 23e3

Format 23f

Syntax: 'F ('S/'B/'P/'G) ADDRESS CA 23f1

This command causes the indicated structural entity to be examined and formatted by the Analyser/Formatter. 23f2

Breakpoint -- Not done yet (11/24/70) 23g

Syntax: 'H CA 23g1

This command causes the executable text program currently in execution to pause, and return control to the user. 23g2

The user may continue the executable text program after the pause by typing a centerdot (??) 23g3

Execute Step Mode -- Not done yet (11/24/70) 23h

Syntax: 'E 'H NUMBER CA 23h1

This command is intended as a debugging tool. 23h2

It causes the equivalent of a breakpoint after the number of commands indicated by NUMBER are executed. 23h3

D. New Analyser/Formatter Features 24

A pair of pointers, PTS ans PTE will be reserved, and will always point to the text currently being executed WHEN IN THE EXECUTE STATEMENT MODE. -- Not done yet (11/24/70) 24a

These pointers may be read and set by a AF program, however it is up to the user to ascertain that the is doing a reasonable thing when they are set. 24a1

PTS will always point to the next caaracter to be read from the executable text string, and PTE will point to the end of that string. 24a2

Subroutine call from AF program 24b

Syntax: "<etsubc>(" '$ PTR ', '$ PTR ') 24b1

This is, in effect, a subroutine call to the executable text identified by the two pointers. 24b2

Note that this does not cause the control to be immediately transferred to the subroutine callled, but rather it causes the subroutine to be executed in the next place where input would be normally requested to the executable text. 24b3

E. Using the New Features 25

The features described above are available only in the experimental version of TODAS (as of 11/25/70). 25a

To enter this experimental version, give the Exec-level command: XTODAS. 25b

:5232, 12/28/70 1403:37 JCN ; ["."HED"]; :HIGHER LEVEL PROCS, 12/02/70 1351:32 WLB ;

1

The query system (called qs in this guide) can be found under KDF as (BOSCH)QUERY or (X1FILES)BQUERY. 1a

To use it, go into TODAS, load the file and execute statement "start" (i.e. type: "es:start" followed by ca.). Now look under (1)

1b

(1) qs types "New inputfiles?? ". 1b1

If you do not want to specify a new list of inputfiles, type "n@" and proceed to (3), otherwise type "y@" and go to (2).

1b1a

(2) qs types "Type new list: ". 1b2

You must now type a list of inputfiles, separated by spaces. Proceed to (3).

1b2a

(3) qs types "Field: ". 1b3

Specify a field by typing the fieldname followed by '@ .You may specify 'any field' starting with a particular letter by just typing that letter, followed by '@ . 1b3a

For example you may type: "a1@" to look in field a1 alone or "a@" to look in field a1 through a5. 1b3b

Proceed to (4).

1b3c

(4) qs types "Subfield?? ". 1b4

If you want to specify a subfield type "y@" and go to (5) else type "n@" and go to (6). 1b4a

If you asked to look in any field of a particular type, you'd better answer this question with no, since qs is not smart enough to look in all the subfields concerned.

1b4b

(5) qs types "Subfieldnumber: ". 1b5

Specify a subfield by typing the subfield number followed by '@ . Now go to (7).

1b5a

(6) qs types "Any fieldgroup?? ". 1b6

If you typed in a single letter under (3), you can type "y@" to say that you want to look in all fields starting with that letter; if you type "n@" qs will just look in the fields starting with that letter until it finds a '#
. 1b6a

If you typed in a full fieldname under (3), you can type "y@" to say that you want to look in all subfields of that field; if you type "n@" qs will just look in the first subfield of the field (i.e. until it finds a '#). 1b6b

Now go to (7).

1b6c

(7) qs types "Text: ". 1b7

Specify the text you want to look for by typing it between quotes. You may specify several textitems separated by '/ , "OR", "AND" or "NOT". 1b7a

The meaning of the separators is the same as for the NLS contentanalyser. 1b7b

The total length of the string you type is limited however, by the size of the buffer used by the TODAS 'substitute statement' command. Goto (8).

1b7c

(8) qs types '- . 1b8

You may now type "a@", "o@" or "f@". 1b8a

We call each string (or set of strings, separated by '/ etc.) of text you want to look for in a particular field or subfield a 'criterion' on which you select citations. 1b8b

Qs allows you to specify several criteria and to use

AND and OR operations. The AND operation has a higher precedence than the OR operation. 1b8c

Examples: we call the criteria c1,c2,... . 1b8d

You may select on basis of (c1 OR c2), (c1 AND c2 AND c3), ((c1 OR c2) AND c3), ((c1 OR c2 OR c3)AND(c4 OR c5)) etc.. 1b8d1

If you type "o@", this invokes an OR between the previous criterion and the next one. Proceed to (9) if the last thing you looked in was a subfield, to (3) otherwise. 1b8e

If you type "a@", this invokes an AND between the previous criterion and the next one. Proceed to (9) if the last thing you looked in was a subfield, to (3) otherwise. 1b8f

If you type "f@", this means you are through and want to perform the actual selection. Go on to (10). 1b8g

(9) qs types "Another subfield in the same field?? ". 1b9

If you want to look in another subfield of the same field, type "y@" and goto (5), otherwise type "n@" and go to (3). 1b9a

(10) qs types "Outputprefix: ". 1b10

Type the desired outputprefix for the Collector Sorter. Proceed to (11). 1b10a

(11) qs types "Files searched: ". 1b11

This time you don't have to do anything, qs will type the files it has searched through and will then print the outputfile (one level, one line). Goto (12). 1b11a

(12) qs types "Done ". 1b12

Qs is done. Goto (13). 1b12a

(13) qs types "Continue?? ". 1b13

If you want to do another round type "y@" and go to (1) else type "n@" to get out of qs and back in TODAS. 1b13a

Since qs searches the inputfiles sequentially, you have to give the fields and subfields in the order in which they occur in the inputfiles. 1c

The excutable text does not provide possebilities for error checking, so be careful, qs won't blow up on you (I hope) but will give a wrong result. 1d

If you made a typing error during the command specification, you can abort qs by hitting two rubouts. In order to assure proper working afterwards you have to leave TODAS, do a reset and start all over again. If you don't do this the collector sorter will probably blow up. 1e

':5233', 12/02/70 1644:31 JCN ; .DPR=1; :JOURNAL, 11/30/70 1131:53 VDB ;
.DPR=0;

Summary of major ARC developments from 1 August 1969 to 1 August 1970

1

User System

2

2a

Developed a typewriter-oriented documentation system, TODAS, allowing access to and manipulation of NLS files from typewriter terminals.

2b

2c

Implemented many new NLS user features including:

2d

(1) Executable text allowing user to write NLS commands as text in a statement

2d1

(2) Substitute command allowing replacement of text strings through selected portions of a file

2d2

(3) New editing commands and extensions to old commands to give more flexible selection of entities

2d3

(4) Improved calculator package for operating on numbers in the text

2d4

(5) File merging capabilities

2d5

(6) Improved output processor for hard copy with expanded directives and formatting capabilities.

2d6

3

Management System

4

4a

Investigated on-line management information-handling techniques on an experimental basis under actual operating conditions within the Center, including development of on-line cost records, estimates, working forecasts, and purchase-order processing records closely integrated with other on-line files.

4b

4c

Conducted activity and task planning using NLS and TODAS with development of various files for task descriptive material, resource allocation, and work status.

4d

4e

Summary of major ARC developments from 1 August 1969 to 1 August 1970

Studied specialized organization and operating techniques needed by the ARC on-line community, with the aim of developing new models for the operation of such groups, recognizing their particular needs and making use of their capabilities. 4f

5

Service System 6

6a

Hardware 6b

(1) Integrated an external core system for refreshing displays 6b1

(2) Replaced existing drums with faster drums giving significant improvement in system capacity 6b2

(3) Installed new line printer with excellent quality upper and lower case print 6b3

(4) Studied ways of increasing system capacity, resulting in decision to lease a PDP-10. Other alternatives were: 6b4

Standard Computer SC-9000 6b4a

Berkeley Computer BCC-1 6b4b

Use of small computers with the 940 for interactive front end to NLS 6b4c

(5) Placed orders for a PDP-10 system to be delivered in November 6b5

6b6

Software 6c

(1) Developed new Tree Meta producing binary output 6c1

(2) Improved MOL and SPL compilers making use of new Tree Meta features 6c2

(3) Reorganized NLS using new MOL and SPLs allowing expansion and greater flexibility in evolution 6c3

(4) Wrote simulation of 940 system and studied factors affecting response to users 6c4

(5) Modified Tree Meta to produce binaries loadable on the PDP-10 6c5

(6) Used this Tree Meta to write a compiler which will replace MOL with the move to the PDP-10. 6c6

7

Network 8

8a

Completed hardware and software interfaces to the Network. Implemented preliminary protocol allowing log-in to our system over the Network. 8b

8c

Participated in Network Working Group to develop operating protocol 8d

8e

Used NetWork to load and checkout programs in the PDP-10 at University of Utah 8f

8g

Developed plans for use of disc file storage at UC Santa Barbara over the Network 8h

8i

Network Information Center (NIC) 9

9a

Established liaison system for personal contact with other Network sites 9b

9c

Established preliminary NIC collection consisting of Network Working Group documents and significant documentation for Network sites 9d

9e

Developed cataloging and indexing procedures and formulated preliminary retrieval techniques 9f

9g

Summary of major ARC developments from 1 August 1969 to 1 August 1970

Studied the use of microform for NIC documentation and evaluated various techniques and equipment 9h

9i

Surveyed data-base management systems that might be available over the Network for NIC use. 9j

9k

Papers and presentations 10

10a

On-line presentation to American Society for Information Sciences in San Francisco, October 1969 10b

10c

Paper on Augmenting the Software Engineer at the COINS Conference in Miami, December 1969 10d

10e

Published comprehensive reports for RADC and NASA covering the 2 years ending February 1970 10f

```
':5234', 12/29/70 1027:08 JCN ; .DPR=1; :2PRPT, 07/22/70 1145:17 JCN ;
;.NSW=0;      .DPR=0;
```

I/O Bus Control Multiplexor Specifications

1

Functional Description. 1a

This unit will interface several peripheral devices to the PDP-10 IO Bus. These devices require no data to be transmitted over the interface; only command signals, status bits, and interrupts will be processed. 1a1

Interrupts 1b

Provision must be made for two interrupt channels to the PDP-10 central processor. Under program control any of twelve device interrupts may be assigned to either priority channel, or may be masked to inhibit any CPU interrupt. The priority level of each of the two CPU interrupt channels must also be dynamically controlled by the program. 1b1

Interrupt pulses from the peripheral devices will set one of twelve flip-flops in a flag register. These flag register bits will be used by the masking and priority selection logic to generate a CPU interrupt. Selected flag bits can be cleared under program control, and except for a general reset of the entire unit, no other mechanism shall exist for clearing an interrupt flag. A technique must also be provided to allow the program to set selected flag bits for maintenance purposes, and as a convenience in certain types of system software organization. 1b2

Three separate CONO instructions shall be used to (a) set or clear selected flag bits, (b) set or clear selected bits in mask register A and set priority level on interrupt channel A, and (c) similarly manipulate mask register B and priority channel B. 1b3

Data Available to Program. 1c

The current values of the two twelve-bit mask registers and the corresponding three-bit priority channel numbers shall be furnished on DATAI instructions issued by the PDP-10 processor. 1c1

The current values of the twelve-bit flag register, and the flag bits "showing through" the two mask rgisters (i.e., the flag bits actually causing a current interrupt on each channel) shall be available in the right half-word on three separate CONI instructions. 1c2

At least 18 bits of peripheral device status information shall be made available on a CONI instruction. This data is a direct sampling of level status lines provided by the various devices; no buffering registers are needed. 1c3

Command Signals to Peripherals. 1d

One CONO instruction will cause pulses to be generated on various command lines to the peripheral units. 1d1

Provision must be made for at least four distinct commands to each of at least twelve devices. 1d2

The command may be a coded binary number, but the sub-device selection should be on individual bit positions in the instruction word, to allow the same command to be sent to several devices simultaneously. 1d3

PDP-10 IO Bus Interface. 1e

This unit will appear as four separate IO device addresses. The two low-order address bits will be decoded to obtain this four-way selection. The five high-order address bits assigned to the unit will be specified when the overall system configuration is finalized. 1e1

While the unit is addressed as four different devices, only one load and/or one driver should be placed on any control or data line on the IO bus. 1e2

The instruction set is arranged so that only the low-order 18 bits of the data bus are used; drivers and receivers for the left half-word are not needed. 1e3

No unique DATAO instructions are used, and the control signals for DATAO should be treated as identical to CONO signals, allowing the software a choice of immediate-operand or memory-operand modes for all control instructions. 1e4

Other than the ignored half-word on the data bus, the interface must conform to DEC specifications for the IO bus, including use of standard cables and connectors and driving and receiving circuits. 1e5

IO bus cables and margin check cable will be furnished by SRI. 1e6

I/O Bus Control Multiplexor Specifications

Peripheral Device Interfaces. 1f

Six devices will initially be connected through the control multiplexor: the Bryant disk system, two display controllers, the input device controller (for keyboard input), the ARPA network message processor, and a line printer. 1f1

Connectors will be provided in the control multiplexor for signal cables from each device. The number of signal lines for each device varies, but the maximum (for the disk system) is eleven. All other current devices have at most six signal paths. These signals are generated by standard positive-logic DTL or TTL integrated circuits. 1f2

Control Panel. 1g

Lamps shall be provided to display the flag register, the two mask registers, the two interrupt channel priority level numbers, the interrupt request level for the two channels, and the 18 device status lines. 1g1

Switches shall be provided for at least the following functions: 1g2

Reset flags 1g2a

Inhibit interrupts on channel A 1g2b

Inhibit interrupts on channel B 1g2c

Power 1h

+10 and -15 volt power supplies should be standard DEC units; +5 volt power should use either DEC or Lambda units. 1h1

Power-down crowbar circuits shall be provided to prevent transients on the interface lines, and power-up shall cause an automatic general reset of all control logic; including clearing the flag register and priority channel number registers. 1h2

All power supplies and power controls are to be supplied by contractor. 1h3

Cabinets and fans will be furnished by SRI. 1h4

Documentation. 1i

A detailed description of the internal operation of the unit and all maintenance features, detailed logic and timing diagrams, a wire list, signal dictionary, and programmer's reference manual must be provided. 1i1

As an option, PDP-10 diagnostic software may also be supplied for maintenance of the interface. 1i2

Expansion Capability. 1j

The design must provide for twelve peripheral devices, twelve interrupt flags and 18 status bits. While only six devices will be connected initially, using eight interrupt flags and at most ten status lines, the expanded capacity must be prewired and tested where feasible and appropriate mounting slots left open for future additions. All twelve bits of register flip-flops and bus gating for all 18 status lines should be supplied for the initial installation. 1j1

Instruction Formats 1k

CONO A - Set Mask and Interrupt Level for Channel A 1k1

bits 18-29-mask bits 1k1a

bit 30 - clear selected mask bits (allow interrupts) 1k1b

bit 31 - set selected mask bits (inhibit interrupts) 1k1c

bit 32 - set interrupt level from bits 33-35 1k1d

bits 33-35-interrupt priority level (as a binary number) 1k1e

CONO B - Set Mask and Interrupt Level for Channel B 1k2

(bit configuration same as for CONO A) 1k2a

CONO C - Set or Clear Flags 1k3

bits 18-29-flag bits 1k3a

bit 30 - clear selected flags (clear interrupt condition) 1k3b

bit 31 - set selected flags (force interrupt) 1k3c

bit 32 - generate master reset for entire interface 1k3d

CONO D - Generate Sub-device Commands 1k4

bits 18-20 - (not used) 1k4a

bits 21-32 - sub-device select (one bit position per peripheral device) 1k4b

bits 33-35 - device order code (as a binary number) 1k4c

DATAO Instructions (Identical functions as for corresponding CONO instructions) 1k5

CONI Instructions 1k6

CONI A - Read Flags Selected by Mask A 1k6a

bits 18-29 - flag bits which show through "holes" in the mask (i.e., flags which are currently causing an interrupt request on Channel A) 1k6a1

bits 30-35 - zero (not used) 1k6a2

CONI B - Read Flags Selected by Mask B 1k6b

(format same as for CONI A) 1k6b1

CONI C - Read Flag Register 1k6c

(format same as for CONI A, but gives all flags without regard to masking) 1k6c1

CONI D - Read Status Lines 1k6d

bits 18-35 - direct sampling of 18 device status lines 1k6d1

DATAI Instructions 1k7

DATAI A - Read Mask and Level on Channel A 1k7a

bits 18-29 current setting of Mask A 1k7a1

bits 30-32 - zero (not used) 1k7a2

bits 33-35 - current setting of Channel A priority level 1k7a3

I/O Bus Control Multiplexor Specifications

DATAI B - Read MasK and Level on Channel B	1k7b
(format same as for DATAI A)	1k7b1
DATAI C and DATAI D	1k7c
(not used - will return all zero data)	1k7c1

```
':5235', 12/02/70 1749:57 JCN ; .DPR=1; :IOMUX, 07/28/70 1649:16 WKE ;
.DPR=0;
```

Ideal Head Up

Current outline for February ROMAC Report (11/17) 1

I Abstract (1page) (DVN) 1a

II Prefactory Credits (1page) (DVN) 1b

III Summary 1c

History and Philosphy (2 pages; brief rough = <1 page) (DvN) 1c1

Highlights of the contrat year (6 pages; brief rough= 1 page) (DvN) 1c2

Future plans (1 page; page; brief rough =<1page) (DCE) 1c3

IV Working in the ARPA Net. 1d

NIC (13 pages; brief rough = 1 page) (JCN) 1d1

Goals, strategy, and philosophy 1d2

Establish first contacts 1d3

Build and maintain collection 1d4

Establish Network Dialogue 1d5

Activity support surveys 1d6

Stimulate dialogue 1d7

Use of network facilities (3 pages; brief rough = 1 page) 1d8

In change to the PDP 10 1d8a

Use of PDP 10's on the net to bootstrap the compiler (WHP) 1d8a1

Running TODAS on another PDP 10 on the net (WHP) 1d8a2

Connection to the network (3 pages; brief rough = 1page) (WKE) 1d9

Hardware Connection 1d9a

Software 1d9b

V Changing from XDS 940 to PDP-10 (37 pages) 1e

Hardware (WKE) (10 pages;brief rough =2 pages) 1e1

Reasons for the chanege 1e1a

The PDP-10 facility 1e1b

Considerations for Design of the facility 1e1c

Adapting non-DEC Equipment 1e1d

Addition of the BB&N Paging Box 1e1e

Monitor + Exec. (JTM)(5 pages; brief rough = 1page) 1e2

Related to Hardware 1e2a

Not Related to Hardware 1e2b

Compiler (DIA+WHP) (7 pages; brief rough = 1 page) 1e3

Convert compiler to produce PDP-10 code on 940 1e3a

Use of Network to bootstrap compiler 1e3b

ReWrite NLS to new compiler language 1e3c

Paging modifications 1e3d

NLS/TODAS (CHI) (10 pages; CHI has some files that will serve as brief rough) 1e4

Getting a version runningon the network 1e4a

Augmentation system adaptation to 10 1e4b

Making the Tenex monitor run with our hardware 1e4c

VI New Tools 1f

Hard ware (WKE) (13 pages; brief rough = 2 pages) 1f1

Univac drums 1f1a

Remote terminals of various kinds 1f1b

Imlac 1f1c

Higher level processes(WLB) (5 pages; brief rough=1page) 1f2

Content Analyser 1f2a

Analyser formatter 1f2b

Collector - Sorter 1f2c

New features in executable text 1f2d

Core NLS---design philosophy (WSD) (3pages; brief rough=<1page) 1f3

New Features for Users 1f4

Journal (JCN) (3 pages; brief rough = 1 page) 1f4a

Concept 1f4a1

What We Did 1f4a2

Next Steps 1f4a3

Relations to Network Dialogue Subsystems 1f4a4

Comments 1f4a5

Mail (WSD) (3pages; brief rough = 1 page) 1f4b

New NLS features (CHI) (7 pages; CHI has a file that will serve for a brief rough) 1f4c

Calculator 1f4c1

Several others (see CHI's file) 1f4c2

Design Team Planning 1f5

Central Planning File 1f5a

Individual files for each task, plan, design, schedule 1f5b

Automatic collecton and integration of schedules 1f5c

When many people plan 1f5d

Updating 1f5e

Remoter terminal experiments (WKE) (2 pages; brief rough =<1page) 1f6

Plans for the future (DCE) (15 pages; rough brief =3 pages) 1g

Glossary (DvN) (2 pages) 1h

References JCN) (2 pages) 1i

Bibliography (JCN) (2pages) 1j

Emphasis in this report should be in this order: first on what we need second on what the people who will use NIC need, third on what NET experimentos need, fourth on what other developers of interactive systems can use 1k

Hence, for example, descriptions of a given use of the NET in converting to the new machine should be most developed under section 6 rather than in connection with the NET. 1k1

Page assignments are tentative. Please find contradictions, redundancies etc and tell me about them. 1l

Report schedule 2

Month >! NO NO NO NO DE DE DE DE JA JA JA JA JA FE FE FE FE 2a

Day >! 07 14 21 28 05 12 19 26 02 09 16 23 30 06 13 20 27 2b

d=draft w=write r=review x=other activity CAP MEANS DONE 2c

Planning and start !xxx 2d

Sec.2 Pref credits ! 2e

DVN >! www 2e1

Sec.6 New features ! 2f

WKE hdwe >! ddd D wwwwwwrrwww 2f1

CHI soft >! ddD wwwwwwrrwww 2f2

WSD core NLS+	>!	ddd	wwwwwwrrwww	2f3
JCN jou+B-line	>!	dddD	wwwwwwrrwww	2f4
WLB HLP	>!	ddd D	wwwwwwrrwww	2f5
Sec.5 XDS940 ←PDP10	!			2g
WKE hdwe	>!	ddD	wwwwwwrrwww	2g1
JTM monitor	>!	ddD	wwwwwwrrwww	2g2
CHI NLS/TODAS	>!	ddD	wwwwwwrrwww	2g3
DIA+WHP compr	>!	ddd	wwwwwwrrwww	2g4
Sec.1 Abstract	!			2h
DVN	>!	ddd	wwwwwwrrwww	2h1
Sec.4 ARPA Network	!			2i
WKE connection	>!	dddD	wwwwwwrrwww	2i1
WHP Net fac use	>!	ddd	wwwwwwrrwww	2i2
JCN NIC	>!	dddD	wwwwwwrrwww	2i3
Sec.3 Summary	!			2j
DCE	>!	ddd	wwwwwwrrwww	2j1
DVN	>!	ddd	wwwwwwrrwww	2j2
Sec.7 Future plans	!			2k
DCE	>!	ddd	wwwwwwrrwww	2k1
Sec.8 Glosssary	!			2l
DVN	>!	wwwrw		2l1
Sec.9 References	!			2m
JCN	>!	wwwrrwww		2m1
DVN	>!	wwwrrwww		2m2

Ideal Head Up

```
   Sec10 Bibilo          !                                                  2n
       JCN             >!                    wwwrrwww                     2n1
       DVN             >!                    wwwrrwww                     2n2
   DCE approval          !                           xxx                   2o
   SRI edit+approval     !                               xxxxxx            2p
   Mail good draft       !                               Feb
   4th>x                                                                   2q
            Day          ! 07 1h 21 28 05 12 19 26 02 09 16 23 30
   06 13 20 27                                                             2r
            Month        ! NO NO NO NO DE DE DE DE JA JA JA JA JA
   FE FE FE FE                                                             2s
   Patterns:   /"CHI" OR"Sec"/;                                            2t
Links to inprocess report sections:                                        3
   Sec.1 Summary        (nouhuys,summary;zxbnZD)                           3a
   etc etc                                                                 3b
```

':5236', 12/06/70 1249:47 JGN ; .DPR=1; ':JRNL1', 12/04/70 1412:30 DVN ;
.DPR=0;

- 07DEC70 DCE 5237

Log: Call from Steve Crocker

In a previous phone discussion, he had expressed an interest in coming to ARC and spending enough time to learn how to use our tools. We had welcomed that, and were to set a time today. 1

He finds that he can't spare time for a long visit this month. 2

I had raised another issue with him -- the possibility of he and I spending time together jointly developing plans for stimulating more activity in the Network -- activity wherein NIC can be more helpful than now in getting things buzzing. About this issue -- could we meet here, or there, for a shorter period to discuss the "NET Dialogue Activity" issue? 3

If here, he gets more value 3a

If there, might include some liaison visiting. (RAND, SDC, UCLA) 3b

I'll look into it, and call back about preferred arrangement from our point of view. Call within a few days. 3c

He'd like a written response to his letter of Nov 23 (5435,). O.K. 4

```
':5237', 12/08/70 1919:33 JCN ; .DPR=1; :JRNLA,. 12/07/70 1700:22 DCE ;
.HED="                                                    07DEC70 DCE 5237

Log:  Call from Steve Crocker ";
|    .    |    .    2    .    3    .    4    .    5    .    6    .    7
.SNF=72; .MCH=65; .SNB=0; .DLS=1; .SCR=2; .RTJ=0; .PGN=0; .COD[21B]=114B;
.DIR=0;       .DPR=0;
```

TABLE OF CONTENTS

TABLE OF CONTENTS 1

INDEX TO DIRECTIVES 2

A. PRINCIPAL CHARACTER DIRECTIVES 7

B. PRINCIPAL LINE FORMAT DIRECTIVES 9

C. PRINCIPAL PAGE LAYOUT DIRECTIVES 13

D. PRINCIPAL STATEMENT INTERPRETATION DIRECTIVES 18

E. OTHER IMPORTANT DIRECTIVES 23

F. OUTPUT PROCESSOR DIRECTIVES (OLD, NEW, AND SOME FUTURE) . 24

G. COMMENTS 47

INDEX TO DIRECTIVES

A. PRINCIPAL CHARACTER DIRECTIVES 7

BSP=n: code for a BackSPace
CAS: CASe array
CMD=n: force all alphabetic characters to specified Case
COD: output character CODe
GCR; Generate a Carriage Return
TAB=n: TABs -- what to do with them
TST: TabSTop array

B. PRINCIPAL LINE FORMAT DIRECTIVES 9

CEN=1/0: CENter,
CRL: Output a CaRrriage Return and Line feed
DLS=1/0: Delete Leading Spaces
DSN=1/0: Delete Statement Numbers
GCR; Generate a Carriage Return
HJB=n: Horizontal Justification of lines in Body area
IND=1/0: INDentation option
INS=n: INdent n spaces per Statement Level
LMS=n: Left Margin Setting
LSP=n: Leading SPaces
MCH=n: Maximum number of printing CHaracters to a line
MIN=n: Maximum number of spaces to INdent
NCH: Number of CHaracters in current line
NIN: Number of INdentation blanks for current line
PEL: Paginate at End of Line
TAB=n: TABs -- what to do with them
TST: TabSTop array

C. PRINCIPAL PAGE LAYOUT DIRECTIVES 13

GRB=n: GRaB
HJH=n: Horizontal Positioning of the Header lines
HJP=n: Horizontal Positioning of the Page number lines
HLN=n: number of blank LiNes to follow the Header
HSW=1/0: Header SWitch
LMS=n: Left Margin Setting
MLN=n: Maximum number of LiNes to bottom of body area
NLN: Number of LiNes in current page
NTP=n: Number of lines from ToP of page to printing area
PEL: Paginate at End of Line
PES: Paginate at End of Statement
PGN=n: current PaGe Number
PGP=n: Verticle Position of the PaGe number
PLN=n: number of LiNes to a Page
PLO=n: Paginate for each Level n statement
PNO=n: Page Numbering Option
PSH=n: Page Show
PST=1/0: Paginate when STatement won't all fit on page on/off
PSW=1/0: Pagination SWitch on/off
RES: page REStore here
WLN=n: Widow LiNes

D. PRINCIPAL STATEMENT INTERPRETATION DIRECTIVES 18

DSN=1/0: Delete Statement Numbers
IBR: Ignore BRanch
IND=1/0: INDentation option
INS=n: INdent n spaces per Statement Level
IRS: Ignore Rest of Statement
IST: Ignore this STatement
LCP=n: Level CLipping
PES: Paginate at End of Statement
PST=1/0: Paginate when STatement won't all fit on page on/off
SCR=n: number of Carriage Returns to separate Statements
SGF=n: SiGnature Format
SNA=1/0: Statement NAmes print on/off
SNB=1/0: Statement Numbers print on/off
SNF=n: Statement Number Format
TLN=n: Truncate to n LiNes

E. OTHER IMPORTANT DIRECTIVES 23

DIR=1/0: DIRective print/not print
IGD=1/0: IGnore Directives
PSH=n: Page Show
RES: page REStore here
SKP=1/0: SKiP on/off

F. OUTPUT PROCESSOR DIRECTIVES (OLD, NEW, AND SOME FUTURE) . 24

BSP=n: code for a BackSPace
CAS: CASe array
CEN=1/0: CENter,
CMD=n: force all alphabetic characters to specified Case
COD: CODe
CRL: Output a CaRrriage Return and Line feed
DIR=1/0: DIRective print/not print
DLS=1/0: Delete Leading Spaces
DMX: MaXimum value a Directive may assume array
DNM: Directive Name array
DOV=1/0: Delete OVerBars
DPN=1/0: Don't Print statement Names
DPR=1/0: Directive PRint
DPV=1/0: Don't Produce Vector output
DSH=n: code to be used for character to do NDH
DSN=1/0: Delete Statement Numbers
DTS=1/0: Delete Trailing Spaces
DUB=1/0: Delete Underbars
FDS=n: output code for a DaSh
FLN=1/0: Format LiNes
FNC=1/0: Case of the Roman page Numbers
GCD: Generate Current time and Date NOT IMPLEMENTED
GCR; Generate a Carriage Return
GRB=n: GRaB
GTB; Generate a TaB
HED: used to set the content of the running HEaD
HJB=n: Horizontal Justification of lines in the Body area
HJH=n: Horizontal Positioning of the Header lines
HJL=n: Horizontal Justification of Line NOT IMPLEMENTED
HJP=n: Horizontal Positioning of the Page number lines
HJS=n: Horizontal Justification of Statement NOT IMP.
HLN=n: number of blank LiNes to follow the Header
HLT: HaLT
HSW=1/0: Header SWitch
IBR: Ignore BRanch
ICR=n; Input code for a Carriage Return
IGD=1/0: IGnore Directives
IND=1/0: INDentation option
INS=n: INdent n spaces per Statement Level
IOV=n; Input code for an OverBar
IPN=n: Increment Page Number NOT IMPLEMENTED
IRS: Ignore Rest of Statement
ISP=n; Input code for a SPace
IST: Ignore this STatement
ITB=n; Input code for a TaB
IUB=n; Input code for an UnderBar
LCP=n: Level CLipping
LMS=n: Left Margin Setting
LSP=n: Leading SPaces

MCH=n: Maximum number of printing CHaracters to a line
MIN=n: Maximum number of spaces to INdent
MLN=n: Maximum number of LiNes to bottom of body area
MSP=n: Maximum number of SPaces for right justification
NBL=n: NumBer of Lines per output line (n-spacing)
NCH: Number of CHaracters in current line
NDH=n: Number of DasHes at end of page
NIN: Number of INdentation blanks for current line
NLN: Number of LiNes in current page
NPX=n: Number PleX NOT IMPLEMENTED
NSW=n: page Numbering SWitch
NTP=n: Number of lines from ToP of page to printed area
NUL: NULl directive
OSW:
OVB=1/0: OVerBar print on/off
OVD=n: OVerbar description for Device
PBI=n: Paragraph Body Indent NOT IMPLEMENTED
PEL: Paginate at End of Line
PES: Paginate at End of Statement
PGN=n: current PaGe Number
PGP=n: Verticle Position of the PaGe number
PIC=1/0: PICture print on/off
PIN=n: Paragraph INdent NOT IMPLEMENTED
PLN=n: number of LiNes to a Page
PLO=n: Paginate for each Level n statement
PNO=n: Page Numbering Option
POV:
PSH=n: Page Show
PST=1/0: Paginate when STatement will not fit on current page
PSW=1/0: Pagination SWitch on/off
RDD: Restore Default-Default values NOT IMPLEMENTED
REL=1: page Restore at End of Line
RES: page REStore here
ROM=1/0: ROMan page numbering, and
RTJ=1/0: RighT Justification on/off
SCR=n: number of Carriage Returns to separate Statements
SGF=n: SiGnature Format
SHU=n: output code for a SHift to Upper case
SHD=n: output code for a SHift to Lower case
SKP=1/0: SKiP on/off
SNA=1/0: Statement NAmes print on/off
SNB=1/0: Statement NumBers print on/off
SNF=n: Statement Number Format
SOV:
SSW=1/0: Stop code SWitch
STP=n: output code for a SToP code
TAB: Output a TAB
TAB=n: TABs -- what to do with them
TAL=n: Tab ALgorithm,
TBD=n: TaB description for Device
TLN=n: Truncate to n LiNes

TMA=n: TeMporary A
TMB=n: TeMporary B
TMC=n: TeMporary C
TMD=n: TeMporary D
TSP=n: Tab SPace, and
TST: TabSTop array
TSW=1/0: Tab SWitch on/off
TYP=1/0: TYPe switch on/off
UBD=n: UnderBar description for Device
UBR=1/0: UnderbaR print on/off
UPR:
USP:
USW:
WLN=n: Widow LiNes

A. PRINCIPAL CHARACTER DIRECTIVES

BSP=n: code for a BackSPace

Do not use -- replaces FBS

CAS: CASe array

The following cases are now possible:
= 0: the character will print in any case
= 1: lower case only
= 2: upper case only
= 3: special film case only
The default settings of this array depend on the device. The array is not used for Devices Printer and Teletype so don't worry about changing this array when you use the COD directive if you are outputting to either of these devices.

CMD=n: force all alphabetic characters to specified Case

n now has the following meanings:
= 0: don't change
= 1: force lower case
= 2: force upper case
The default setting is still zero.

COD: output charater CODe

By means of this directive and the directive giving the character case (see the description of CAS), it is possible to change the output code for any character in the input.
For example: to change the output code for the number 1 from a verticle bar to a lower case l, use the following directive: COD[21B]=114B (the input code for a one is 21B and the output code for an l is 114B), If the device is Dura or Film, then one has to worry about the case too.

GCR; Generate a Carriage Return

Replaces CRL.

TAB=n: TABs -- what to do with them

This replaces parts of the old directives TAL (TAb Algorithm), TSP (Tab SPace), and TSW (Tab SWitch) and straighten them out. The default setting is 1.
The three possible settings will be:
- = 0: delete tabs
- = 1: keep tabs
- = 2: replace tabs by a single space

TST: TabSTop array

An array directive which is used to determine where the tab stop settings are.
This is a bit array stored in six words (144 bits). The ith bit corresponds to the ith column. The first bit in the array is consedered to be number zero. The first word in the array is also number zero.
A one bit indicates a tab stop and setting a position to 0 will clear a tab stop.
An example: TST[0]=040000000B and TST[2]=00002000B
will set tabstops in the 3rd and 61st columns; clear any previously existing tabstops in columns 1, 2, 4 thru 23 inclusive, 48 thru 60 inclusive, and 62 thru 71 inclusive; and leave in their previous state columns 24 thru 47 inclusive and 72 thru 143 inclusive.
Upon activation of the Output Processor, the array TST is initialized according to the tab stops set in the NLS Viewchange Parameters.

B. PRINCIPAL LINE FORMAT DIRECTIVES

CEN=1/0: CENter

Superseded by HJB (Horizontal Justification of Body):
CEN=1 is now HJB=3 CEN=0 is now HJB=1

CRL: Output a CaRrriage Return and Line feed

Name changed to GCR (Generate a Carriage Return)

DLS=1/0: Delete Leading Spaces

DLS is effective for each output line in the body area (but the LSP spaces won't be deleted).
If the first character(s) of a statement are blanks then they are affected by DLS. Because of the OP's line break algorithm, the only other time leading spaces will occur is when there are spaces following a carriage return in an input statement. If one is using leading spaces to produce tabular output, then be sure DLS is zero.
The default setting is zero -- leave the spaces alone.

DSN=1/0: Delete Statement Numbers

Replaced by SNB=0/1 (Statement NumBer print off/on).

GCR; Generate a Carriage Return

Replaces CRL.

HJB=n: Horizontal Justification of the lines in the Body area

The default setting is 1.
How do you want the lines of the body area formatted -- let me count the ways:
= 0: don't format the lines
i.e., don't bother backing up to last invisible to make a line break, but maybe make a line break in the middle of a word
This replaces the old directive FLN=0 (don't Format LiNes)
= 1: set lines flush left
This replaces the old directive FLN=1 (Format LiNes) when CEN=0 (CENtering off) and RTJ=0 (RighT Justification off)
= 2: set lines flush right
= 3: set lines centered with respect to left margin setting
i.e., centered between the left and right margins
This replaces the old directive CEN=1 (CENtering on)
= 4: set lines centered with respect to page
i.e., centered as if LMS=0
= 5: set lines centered with respect to indentation for the statement
i.e., indent according to LMS and the statement's level and then center between that point and the right margin
= 6: set odd/even numbered pages lines flush right/left
= 7: set odd/even numbered pages lines flush left/right
= 8: set lines "right justified"
if can't: set lines flush left
"can't" means that it would take more than MSP spaces to do it. Also the last output line of every statement is set according to the "can't" option.
This replaces the old directive RTJ=1 (RighT Justification on)
= 9: set lines "right justified"
if can't: set lines flush right
= 10: set lines "right justified"
if can't: set odd/even pages lines flush right/left
= 11: set lines "right justified"
if can't: set odd/even pages lines flush left/right
If there is a tab in a line then the line is set flush left.

IND=1/0: INDentation option

If IND=1 then indent according to the statement's level (see INS) will be performed. This directive has no effect on LMS (Left Margin Setting).

Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec.

INS=n: INdent n spaces per Statement Level

Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec.

LMS=n: Left Margin Setting

This sets the left margin of the page to n columns to the right of the standard (on all devices it's to the right of the edge of the page to begin with). Thus except for lines that are "centered with respect to the page", all lines will be indented at least n columns.
The default setting is zero.
LMS applies equally to the body, header, and page number areas.

LSP=n: Leading SPaces

If SNB=0 (don't print Statement NumBers), then print n blanks before printing the first character of the statement text. Note that the n blanks are in addition to the blanks required for the LMS (Left Margin Setting) and statement indentation (IND and INS) directives. This directive is effective for the first output line of the statement only -- not subsequent ones.
The default setting is 0.

MCH=n: Maximum number of printing CHaracters to a line

Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec. MCH is set to the number of NLS/TODAS columns minus one (unless the device is Teletype, in which case MCH gets set to 64 -- to allow room for SNF=72 on narrow teletype paper). MCH is set to the number of columns minus one because Create Display has a different line break algorithm than that of the Output Processor. This way the Output Processor will almost always make the same line breaks as Create Display did. MCH is equally applicable to the body, running head, and page number "areas".

MIN=n: Maximum number of spaces to INdent

LMS is included when enforcing MIN.
The default setting is 48.

NCH: Number of CHaracters in current line

Only the Output Processor programs can change the value of NCH. The user can only query their current value, e.g. in an IF clause of another directive.

NIN: Number of INdentation blanks for current line

Only the Output Processor programs can change the value of NIN. The user can only query their current value, e.g. in an IF clause of another directive. The value of NIN now includes LMS.

PEL: Paginate at End of Line

This directive replaces REL. The old form was REL=1 -- the new form is PEL.

TAB=n: TABs -- what to do with them

This replaces parts of the old directives TAL (TAb Algorithm), TSP (Tab SPace), and TSW (Tab SWitch) and straighten them out.
The default setting is 1.
The three possible settings will be:
- = 0: delete tabs
- = 1: keep tabs
- = 2: replace tabs by a single space

TST: TabSTop array

An array directive which is used to determine where the tab stop settings are.
This is a bit array stored in six words (144 bits). The ith bit corresponds to the ith column. The first bit in the array is consedered to be number zero. The first word in the array is also number zero.
A one bit indicates a tab stop and setting a position to 0 will clear a tab stop.
An example: TST[0]=04000000B and TST[2]=00002000B
will set tabstops in the 3rd and 61st columns; clear any previously existing tabstops in columns 1, 2, 4 thru 23 inclusive, 48 thru 60 inclusive, and 62 thru 71 inclusive; and leave in their previous state columns 24 thru 47 inclusive and 72 thru 143 inclusive.
Upon activation of the Output Processor, the array TST is initialized according to the tab stops set in the NLS Viewchange Parameters.

C. PRINCIPAL PAGE LAYOUT DIRECTIVES

GRB=n: GRaB

Paginate immediately before the next statement if first line of next statement would be within n+1 lines of the bottom of the page.
Usual use is for "heading widows". One doesn't want a chapter or subsection to be the very last thing on a page, but would like one or more line of the section itself to appear on the same page as the heading.
The way the directive works means that the GRB should appear in the statement immediately before the statement that is the heading.

HJH=n: Horizontal Positioning of the Header lines

Same options as with HPB except that centered with respect to indentation doesn't make any sense.
The default setting is 1 -- left flush. Maybe it ought to be changed to centered.

HJP=n: Horizontal Positioning of the Page number lines

Same options as with HPB except no "right justification" and centered with respect to indentation doesn't make any sense.
A subset of its options replace NSW = 1 or 2 (page numbers centered or flush right on odd pages and flush left on even pages).
Default setting is 3 -- center between right and left margins (taking LMS into account).

HJP=3 replaces NSW=1 (center page numbers)
HJP=6 replaces NSW=2 (put odd page numbers flush right and even page numbers flush left)

HLN=n: number of blank LiNes to follow the Header

Effective only if HSW=1 and there has been a HED directive.
The default setting is 3.

HSW=1/0: Header SWitch

If HSW=0 then no header will be output at the top of each page.
The default setting is 1.

LMS=n: Left Margin Setting

This sets the left margin of the page to n columns to the right of the standard (on all devices it's to the right of the edge of the page to begin with). Thus except for lines that are "centered with respect to the page", all lines will be indented at least n columns.
The default setting is zero.
LMS applies equally to the body, header, and page number areas.

MLN=n: Maximum number of LiNes to the bottom of the body area

This means that the last line of the body area will not fall below the nth line. Note that some of the n lines may be taken up by NTP, the running head, and HLN.
Actually the last line of the body may be printed as far down as the MLN + 2nd line. If all three of the last line of statement text on the page, the SNF statement number, and the SGF signature overlap each other and the last line of a statement's text falls on line MLN, then the statement number will be on line MLN + 1 and the signature will be on line MLN + 2.
The default setting is 56.

NLN: Number of LiNes in current page

Only the Output Processor programs can change the value of NLN. The user can only query their current value, e.g. in an IF clause of another directive.

NTP=n: Number of lines down from ToP of page to begin printing

The default setting is 3.

PEL: Paginate at End of Line

This directive replaces REL. The old form was REL=1 -- the new form is PEL.

PES: Paginate at End of Statement

When the entire statement (including statement number, signature, and/or picture) has been output, a new page is begun.
It is suggested that this directive be used in almost all places where the RES directive is now being used.
If you are using SNF and/or SGF then you will probably want the statement number and/or signature to be printed on the same page as their statements. If SNF and SGF are not being used and the RES is the last thing in its statement, there will be SOR blank lines at the top of the body area of the next page. Thus it would seem that the only time someone would want to use a RES would be to paginate in the middle of a statement or to get a blank page by having a RES immediately precede a PES at the end of a statement.

PGN=n: current PaGe Number

The page number that would appear on the current output page. The default setting is such that the first output page would be number 1.

PGP=n: Verticle Position of the PaGe number

Meaning of n changed. n used to be the number of lines up from the page bottom to put the page number, but is now the number of blank lines to insert between the bottom of text body area and the line that is to contain the page number. Thus the page number will be printed in line MLN + PGP + 1 of the page,
This will allow the changing of the text body size (MLN) without having to also change PGP.
The default setting is still 5. New pages will look like old pages.

PLN=n: number of LiNes to a Page

Includes header, body, and page number areas.
The default setting is 66.

PLO=n: Paginate for each Level n statement

PLO can now be set to any number n -- which means that all statements of level n or higher will cause a page break to occur if the statement is not the head of its sublist (which I think is what is wanted).
The default setting is still zero.

PNO=n: Page Numbering Option

This combines the old option NSW=0 (no page number) and the directives ROM (Roman numeral page numbers or not) and FNC (upper or lower case for Roman numeral page numbers).
Default setting is 1 -- arabic page numbers.
The four possible settings are:

- 0: no page number
  replaces NSW=0
- 1: arabic numeral page numbers
  replaces ROM=0
- 3: lower case Roman numeral page numbers
  replaces ROM=1 and FNC=3 or 5
- 4: upper case Roman numeral page numbers
  replaces ROM=1 and FNC=1 or 4

PSH=n: Page Show

Only produce output for page n, but format and scan all the other pages for directives.
The default setting is zero, which means print all pages.
This would be nearly equivalent to beginning the file with a TYP=0 and having a TYP=1 immediately before page n and a TYP=0 immediately after.
Note that there can be several PSH's in a file and if put in the right places, one could get any number of single pages as output.

PST=1/0: Paginate when STatement won't all fit on page on/off

The algorithm for estimating the number of output lines a statement will take up has been changed and (hopefully) is now much more accurate.
The default setting is still 0.

PSW=1/0: Pagination Switch on/off

If PSW=1 then the directives involved with page numbering (PGP, PNO, and HJP), dashes at the end of a page (NDH and DSH), stop code at the end of a page (SSW and STP), verticle size of the page (PLN), getting to the top of the next page, spacing down from the top of the next page (NTP), and the running head (HSW, HED, HJH, and HLN) will be executed.
The default setting is 1.

RES: page REStore here

Causes a page restore (new page) at the point the directive occurs.
It is suggested that the new directive PES (Paginate at End of Statement) will do what you really want done instead of using RES. See the description of that directive.

WLN=n: Widow LiNes

Number of lines of a statement guaranteed to be output on the next page if the statement would not all fit on the current page. The "guarantee" is like many guarantees these days. The algorithm for estimating the number of output lines a statement will take up has been changed and (hopefully) is now much more accurate.
The default setting is still 2.

D. PRINCIPAL STATEMENT INTERPRETATION DIRECTIVES

DSN=1/0: Delete Statement Numbers

Replaced by SNB=0/1 (Statement NumBer print off/on).

IBR: Ignore BRanch

At the point this directive is encountered, the statement containing it is treated the same way as if an IST had occurred. In addition all subseqent statements are ignored (without any scanning at all) until a statement is seen that is of a level less than or equal to that of the statement in which the IBR occurred.

IND=1/0: INDentation option

If IND=1 then indent according to the statement's level (see INS) will be performed. This directive has no effect on LMS (Left Margin Setting).
Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec.

INS=n: INdent n spaces per Statement Level

Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec.

IRS: Ignore Rest of Statement

At the point this directive is encountered the same thing happens as if the directive were the last thing in its statement.

IST: Ignore this STatement

Normally the Output Processor will behave just as if a statement containing an IST were not there. It will not get confused if the next statement it sees is of a lower or higher level.
Any directives occuring in the same statement but before this one are recognized and executed. Thus a good way to hide directives on output might be to make up a statement consisting entirely of directives, the last of which is IST. Then you won't even get a blank line output for the statement.
If IST would occur in the ith output (printed not input) line of a statement, then the first i-1 lines of that statement will be printed -- there is no backup beyond the current line -- so be sure to put the IST early enough in the statement.

LCP=n: Level CLipping

This will work similarly to the NLS L Viewspec.
The "default setting" is the NLS L Viewspec at the time the file is output thru the Output Processor.
If 1 is the setting of the L Viewspec when the file is output thru the Output Processor, the Output Processor only sees the first 1 levels of statements. So having n > 1 just won't do anything.

PES: Paginate at End of Statement

When the entire statement (including statement number, signature, and/or picture) has been output, a new page is begun.
It is suggested that this directive be used in almost all places where the RES directive is now being used.
If you are using SNF and/or SGF then you will probably want the statement number and/or signature to be printed on the same page as their statements. If SNF and SGF are not being used and the RES is the last thing in its statement, there will be SCR blank lines at the top of the body area of the next page. Thus it would seem that the only time someone would want to use a RES would be to paginate in the middle of a statement or to get a blank page by having a RES immediately precede a PES at the end of a statement.

PST=1/0: Paginate when STatement won't all fit on page on/off

The algorithm for estimating the number of output lines a statement will take up has been changed and (hopefully) is now much more accurate.
The default setting is still 0.

SCR=n: number of Carriage Returns to separate Statements

After printing the last line of a statement, the OP will output SCR*NBL carriage returns.
The "default setting" is determined by the NLS blank line Viewspec. If blank lines are on, then SCR is initialized to two. Otherwise it is initialized to one.
Watch out for this initialization. It is the only one that under normal conditions will result in something different from the old PASSA.
Setting SCR to zero will no longer work correctly.

SGF=n: SiGnature Format

Its setting has a similiar meaning to that of SNF, i.e., if n > 0, print each statement's signature (date, time, and initials of the person when the statement was created or last alterred) right justified to column n after the last of the text of the statement has been printed.
The "default setting" is determined by the NLS Viewspecs in force at the time the file is output thru the Output Processor. If signatures are on and blank lines are on, then SGF is set to 60; otherwise it is set to zero -- this is the same convention as in NLS.
If SCR*NBL = 1, the Output Processor will attempt to put the signature in the last line of the statement. If the signature would "overlap" the text of the statement or the statement number, then it will put the signature in a blank line following the statement. A blank line will be forced, if necessary, to accomodate the signature (before the statement number was not printed if SCR*NBL=1 and the statement number overlapped the last line of text).
A convention will be followed that if SCR > 1, then the signature will be forced onto a blank line following the last line of its statement -- it will not go on the same line as the last line of its statement even if it wouldn't "overlap".
If both SNF and SGF are set and they "overlap" each other, then the statement number has precedence (the signature will be printed on the next line).

Two things "overlap" if there is not at least one space between the ends of the things. There are 20 characters in a signature.

The signature and statement number will be printed no matter what SGR and NBL are. However the lines occupied by SGF and/or SNF are subtracted from SCR*NBL -- there won't be SCR*NBL blank lines following the signature and statement number unless they are both printed on the same line as the last line of the text of their statement.

If the signature is printed on a line following the statement, the directive LMS (Left Margin Set) will not be effective for that line so that it will be possible to get the signature printed in the left margin. The amount of indentation for a statement has no affect on the placement of the signature. This is a different convention than was used before with SNF.

If 0 < n <= 20, the signature will be printed flush against the left edge of the page (there are 20 characters in a signature).

The signature (and statement number) will always go on the same page as the last line of its statement (unless there is a RES in the statement).

The bugs that occured before with SNF when the line containing the statement number was supposed to be centered or the line contained nothing or nothing but but blanks will not occur.

SGF may be used in conjunction with the directive MCH, which sets the right margin for the body of the printout. SGF is not constrained by the setting of MCH -- it can be larger.

SNA=1/0: Statement NAmes print on/off

Replaces old directive DPN=0/1 (Don't Print statement Names)
Default setting is 0 --'t print statement names.

SNB=1/0: Statement NumBers print on/off

Replaces old directive DSN=0/1 (Delete Statement Numbers)
Default setting is 0 -- don't print statement numbers.

Note that SNB is entirely independent of SNF.

SNF=n: Statement Number Format

This works the same as it did before except that a few bugs and shortcomings will no longer happen:

if n > 0, print each statement's statement number right justified to column n after the last of the text of the statement has been printed.

The default setting is zero, except for Device Teletype where it is 72.

The Output Processor will attempt to put the statement number in the last line of the statement. If the statement number would "overlap" the text of the statement, then it will put the number in a blank line following the statement. A blank line will be forced, if necessary, to accomodate the statement number (before the number was not printed if SCR*NBL=1 and the number "overlapped" the last line of text).
If both SNF and SGF are set and they "overlap" each other, then the statement number has precedence (the signature will be printed on the next line).

Two things "overlap" if there is not at least one space between the ends of the things.

The statement number and signature will be printed no matter what SCR and NBL are. However the lines occupied by SGF and/or SNF are subtracted from SCR*NBL -- there won't be SCR*NBL blank lines following the statement number and signature unless they are both printed on the same line as the last line of the text of their statement.
The statement number (and signature) will always go on the same page as the last line of its statement (unless there is a RES as the last thing in the statement).
The bugs that occured when the line containing the statement number was supposed to be centered or the line contained nothing or nothing but but blanks will not occur.
If the statement number is printed on a line following the statement, the directive LMS (Left Margin Set) will not be effective for that line so that it will be possible to get the statement number printed in the left margin. The amount of indentation for a statement has no affect on the placement of the number. This is a different convention than was used before.

If n = 1, the statement number will be printed flush against the left edge of the page.

SNF may be used in conjunction with the directive MCH, which sets the right margin for the body of the printout. SNF is not constrained by the setting of MCH -- it can be larger.

TLN=n: Truncate to n LiNes

Will work the same as the NLS T Viewspec.
The "default setting" is the NLS T Viewspec at the time the file is output thru the Output Processor.

E. OTHER IMPORTANT DIRECTIVES

DIR=1/0: DIRective print/not print

Replaces old directive DPR (Directive PRint)
Default setting is still 0 -- don't print directives.

IGD=1/0: IGnore Directives

Any directives encountered between IGD=1 to IGD=0 will be ignored except that directives will be recognized in order to effect the directive DIR (DIRective print on/off).

PSH=n: Page Show

Only produce output for page n, but format and scan all the other pages for directives.
The default setting is zero, which means print all pages.
This would be nearly equivalent to beginning the file with a TYP=0 and having a TYP=1 immediately before page n and a TYP=0 immediately after.
Note that there can be several PSH's in a file and if put in the right places, one could get any number of single pages as output.

RES: page REStore here

Causes a page restore (new page) at the point the directive occurs.
It is suggested that the new directive PES (Paginate at End of Statement) will do what you really want done instead of using RES. See the description of that directive.

SKP=1/0: SKiP on/off

Now while SKP is on, directives (except SKP) won't be executed (they used to be).
The default setting is still 0.

```
.HED="                                          08DEC70 WLB5238NEWO
(OLD,NEW,ANDSOME
FUT
```

F. OUTPUT PROCESSOR DIRECTIVES (OLD, NEW, AND SOME FUTURE)

BSP=n: code for a BackSPace

Do not use -- replaces FBS

CAS: CASe array

The following cases are now possible:
= 0: the character will print in any case
= 1: lower case only
= 2: upper case only
= 3: special film case only
The default settings of this array depend on the device. The array is not used for Devices Printer and Teletype so don't worry about changing this array when you use the COD directive if you are outputting to either of these devices.

CEN=1/0: CENter

Superseded by HJB (Horizontal Justification of Body):
CEN=1 is now HJB=3 CEN=0 is now HJB=1

CMD=n: force all alphabetic characters to specified Case

n now has the following meanings:
= 0: don't change
= 1: force lower case
= 2: force upper case
The default setting is still zero.

COD: CODe

By means of this directive and the directive giving the character case (see the description of CAS), it is possible to change the output code for any character in the input.
For example: to change the output code for the number 1 from a verticle bar to a lower case l, use the following directive:
COD[21B]=114B (the input code for a one is 21B and the output code for an 1 is 114B), If the device is Dura or Film, then one has to worry about the case too.

CRL: Output a CaRrriage Return and Line feed

Name changed to GCR (Generate a Carriage Return)

DIR=1/0: DIRective print/not print

Replaces old directive DPR (Directive PRint)
Default setting is still 0 -- don't print directives.

DLS=1/0: Delete Leading Spaces

DLS is effective for each output line in the body area (but the LSP spaces won't be deleted).
If the first character(s) of a statement are blanks then they are affected by DLS. Because of the OP's line break algorithm, the only other time leading spaces will occur is when there are spaces following a carriage return in an input statement. If one is using leading spaces to produce tabular output, then be sure DLS is zero.
The default setting is zero -- leave the spaces alone.

DMX: MaXimum value a Directive may assume array

The ith entry in the array contains the maximum value to which the ith directive may be set. The order of the directives in the arrays has been changed. Thus if you used this directive, now it will change the wrong maximum value.

DNM: Directive Name array

The ith entry in the array contains the name, i.e., the 3 letter mnemonic, of the ith directive. The order of the directives in the arrays has been changed. Thus if you used this directive, now it will change the wrong name.

DOV=1/0: Delete OVerBars

Replaced by OVB=0/1 (OVerBar print off/on). The new directive applies only to 8-bit overbars.

DPN=1/0: Don't Print statement Names

Replaced by SNA=0/1 (Statement NAmes print off/on).

DPR=1/0: Directive PRint

Name changed to DIR=1/0 (DIRective print on/off)

DPV=1/0: Don't Produce Vector output

Replaced by PIC=0/1 (PICture print off/on).

DSH=n: code to be used for character to do NDH

Replaces FDS (the code for the character to be printed when NDH (Number of Dashes at end of each page) is greater than zero). Note that you can print a row of Q's at the bottom of every page if you'd like.
The default setting is 15B -- a dash (minus sign).

DSN=1/0: Delete Statement Numbers

Replaced by SNB=0/1 (Statement NumBer print off/on).

DTS=1/0: Delete Trailing Spaces

DTS is effective for each output line in the body area. Because of the OP's line break algorithm, the only time this directive has any effect is when lines are being centered, set right flush, or "right justified". Any trailing spaces will then cause their lines to be positioned differently than if the trailing spaces were not there.
The default setting is one -- delete the spaces.

DUB=1/0: Delete Underbars

Replaced by UBR=0/1 (UnderbaR print off/on). The new directive applies only to 8-bit underbars.

FDS=n: output code for a DaSh

Name changed to DSH (output code for a DaSH -- the character that will go out when NDH (Number of DasHs to output at end of page) is greater than zero)

FLN=1/0: Format LiNes

Superseded by HJB (Horizontal Justification of Body):
FLN=0 is now HJB=0 FLN=1 is now HJB=1

FNC=1/0: Case of the Roman page Numbers

The old directives NSW, ROM, and FNC have been combined into options of the new directives PNO (Page Numbering Option) and HJP (Horizontal Justification of Page Number).

GCD: Generate Current time and Date NOT IMPLEMENTED

GCR; Generate a Carriage Return

Replaces CRL.

GRB=n: GRaB

Paginate immediately before the next statement if first line of next statement would be within n+1 lines of the bottom of the page.
Usual use is for "heading widows". One doesn't want a chapter or subsection to be the very last thing on a page, but would like one or more line of the section itself to appear on the same page as the heading.
The way the directive works means that the GRB should appear in the statement immediately before the statement that is the heading.

GTB; Generate a TaB

Replaces TAB.
There are going to be a lot more directives of the "Generate" kind -- see Stage II. This is an effort to make their names consistent -- the directive names will all begin with a G.

HED: used to set the content of the running HEaD

For example: HED="HEADING" will set the OP to output "HEADING" at the top of each page (if "HSW" is set on).

HJB=n: Horizontal Justification of the lines in the Body area

The default setting is 1.
How do you want the lines of the body area formatted -- let me count the ways:
= 0: don't format the lines
i.e., don't bother backing up to last invisible to make a line break, but maybe make a line break in the middle of a word
This replaces the old directive FLN=0 (don't Format LiNes)
= 1: set lines flush left
This replaces the old directive FLN=1 (Format LiNes) when CEN=0 (CENtering off) and RTJ=0 (RighT Justification off)
= 2: set lines flush right
= 3: set lines centered with respect to left margin setting
i.e., centered between the left and right margins
This replaces the old directive CEN=1 (CENtering on)
= 4: set lines centered with respect to page
i.e., centered as if LMS=0
= 5: set lines centered with respect to indentation for the statement
i.e., indent according to LMS and the statement's level and then center between that point and the right margin
= 6: set odd/even numbered pages lines flush right/left
= 7: set odd/even numbered pages lines flush left/right
= 8: set lines "right justified"
if can't: set lines flush left
"can't" means that it would take more than MSP spaces to do it. Also the last output line of every statement is set according to the "can't" option.
This replaces the old directive RTJ=1 (RighT Justification on)
= 9: set lines "right justified"
if can't: set lines flush right
= 10: set lines "right justified"
if can't: set odd/even pages lines flush right/left
= 11: set lines "right justified"
if can't: set odd/even pages lines flush left/right
If there is a tab in a line then the line is set flush left.

HJH=n: Horizontal Positioning of the Header lines

Same options as with HPB except that centered with respect to indentation doesn't make any sense.
The default setting is 1 -- left flush. Maybe it ought to be changed to centered.

HJL=n: Horizontal Justification of Line NOT IMPLEMENTED

Effective for just the output line in which it occurs (user doesn't have to say CEN=1--CEN=0 or HJB=3--HJB=1).
Would have same options as HJB.

HJP=n: Horizontal Positioning of the Page number lines

Same options as with HPB except no "right justification" and centered with respect to indentation doesn't make any sense.
A subset of its options replace NSW = 1 or 2 (page numbers centered or flush right on odd pages and flush left on even pages).
Default setting is 3 -- center between right and left margins (taking LMS into account).
HJP=3 replaces NSW=1 (center page numbers)
HJP=6 replaces NSW=2 (put odd page numbers flush right and even page numbers flush left)

HJS=n: Horizontal Justification of Statement NOT IMPLEMENTED

Effective for only the current output line and the remaining lines of the statement in which it occurs (user doesn't have to say HJB=3--HJB=1).
Would have same options as HJB.

HLN=n: number of blank LiNes to follow the Header

Effective only if HSW=1 and there has been a HED directive.
The default setting is 3.

HLT: HaLT

At the point this directive is encountered the same thing happens as if the directive were the last thing in its statement and the statement were the last in the file. The output file is closed normally and everything up to that point gets printed.
(existed previously, but now it'll do it)

HSW=1/0: Header SWitch

If HSW=0 then no header will be output at the top of each page.
The default setting is 1.

IBR: Ignore BRanch

At the point this directive is encountered, the statement containing it is treated the same way as if an IST had occurred. In addition all subseqent statements are ignored (without any scanning at all) until a statement is seen that is of a level less than or equal to that of the statement in which the IBR occurred.

ICR=n: Input code for a Carriage Return

Do not use -- replaces FCR

IGD=1/0: IGnore Directives

Any directives encountered between IGD=1 to IGD=0 will be ignored except that directives will be recognized in order to effect the directive DIR (DIRective print on/off).

IND=1/0: INDentation option

If IND=1 then indent according to the statement's level (see INS) will be performed. This directive has no effect on LMS (Left Margin Setting).
Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec.

INS=n: INdent n spaces per Statement Level

Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec.

IOV=n: Input code for an OverBar

Do not use -- replaces FOV

IPN=n: Increment Page Number NOT IMPLEMENTED

A slightly nicer way to increment the page number than the present (equivalent) method: PGN=PGN+n.

IRS: Ignore Rest of Statement

At the point this directive is encountered the same thing happens as if the directive were the last thing in its statement.

ISP=n: Input code for a SPace

Do not use -- replaces FSP

IST: Ignore this STatement

Normally the Output Processor will behave just as if a statement containing an IST were not there. It will not get confused if the next statement it sees is of a lower or higher level.
Any directives occuring in the same statement but before this one are recognized and executed. Thus a good way to hide directives on output might be to make up a statement consisting entirely of directives, the last of which is IST. Then you won't even get a blank line output for the statement.
If IST would occur in the ith output (printed not input) line of a statement, then the first i-1 lines of that statement will be printed -- there is no backup beyond the current line -- so be sure to put the IST early enough in the statement.

ITB=n: Input code for a TaB

Do not use -- replaces FTB

IUB=n: Input code for an UnderBar

Do not use -- replaces FUB

LCP=n: Level CLipping

This will work similarly to the NLS L Viewspec.
The "default setting" is the NLS L Viewspec at the time the file is output thru the Output Processor.
If 1 is the setting of the L Viewspec when the file is output thru the Output Processor, the Output Processor only sees the first 1 levels of statements. So having n > 1 just won't do anything.

LMS=n: Left Margin Setting

This sets the left margin of the page to n columns to the right of the standard (on all devices it's to the right of the edge of the page to begin with). Thus except for lines that are "centered with respect to the page", all lines will be indented at least n columns.
The default setting is zero.
LMS applies equally to the body, header, and page number areas.

LSP=n: Leading SPaces

If SNB=0 (don't print Statement NumBers), then print n blanks before printing the first character of the statement text. Note that the n blanks are in addition to the blanks required for the LMS (Left Margin Setting) and statement indentation (IND and INS) directives. This directive is effective for the first output line of the statement only -- not subsequent ones.
The default setting is 0.

MCH=n: Maximum number of printing CHaracters to a line

Upon activation of the Output Processor, the default value of n is set according to the corresponding NLS/TODAS Viewspec. MCH is set to the number of NLS/TODAS columns minus one (unless the device is Teletype, in which case MCH gets set to 64 -- to allow room for SNF=72 on narrow teletype paper). MCH is set to the number of columns minus one because Create Display has a different line break algorithm than that of the Output Processor. This way the Output Processor will almost always make the same line breaks as Create Display did.
MCH is equally applicable to the body, running head, and page number "areas".

MIN=n: Maximum number of spaces to INdent

LMS is included when enforcing MIN.
The default setting is 48.

MLN=n: Maximum number of LiNes to bottom of body area

This means that the last line of the body area will not fall below the nth line. Note that some of the n lines may be taken up by NTP, the running head, and HLN.
Actually the last line of the body may be printed as far down as the MLN + 2nd line. If all three of the last line of statement text on the page, the SNF statement number, and the SGF signature overlap each other and the last line of a statement's text falls on line MLN, then the statement number will be on line MLN + 1 and the signature will be on line MLN + 2.
The default setting is 56.

MSP=n: Maximum number of SPaces for right justification

MSP is set to the maximum number of spaces which may be inseted into a line when doing right justification. If more than MSP spaces would be required, the line is set according to the "can't" option. See description of the new directive HJB.
The default setting is 15.

NBL=n: NumBer of Lines per generated output line (n-spacing)

The OP makes up an output line, prints it and then outputs NBL carraige returns. The default setting is one, so if you want "double-spacing" (like when you ask a typist to double space), then set NBL to 2.
NBL is effective for the body area only. The running head gets printed as if NBL were one.

NCH: Number of CHaracters in current line

Only the Output Processor programs can change the value of NCH. The user can only query their current value, e.g. in an IF clause of another directive.

NDH=n: Number of DasHes at end of page

The character output for the "dash" may be changed by means of the directive DSH.
Not meaningful for printer or film output.
Default setting for Teletype is 9. Its 0 for the DURA.

NIN: Number of INdentation blanks for current line

Only the Output Processor programs can change the value of NIN. The user can only query their current value, e.g. in an IF clause of another directive. The value of NIN now includes LMS.

NLN: Number of LiNes in current page

Only the Output Processor programs can change the value of NLN. The user can only query their current value, e.g. in an IF clause of another directive.

NPX=n: Number PleX NOT IMPLEMENTED

The sublist (not the whole plex) that is one level below the statement in which the NPX occurs will be numbered. This number will go before the statement number (SNB) or leading spaces (LSP) at the beginning of the first output line of each of the statements in the sublist. The statements will be numbered consecutively according to the following options (setting of n):

- = 0: no numbering
- = 1: Arabic numerals
- = 2: Roman upper case numerals
- = 3: Roman lower case numerals
- = 4: lower case alphabetic characters
- = 5: upper case alphabetic characters
- = 6: statement number type
- = 7: outline type

NSW=n: page Numbering SWitch

The old directives NSW, ROM, and FNC have been combined into options of the new directives PNO (Page Numbering Option) and HJP (Horizontal Justification of Page Number).

NTP=n: Number of lines down from ToP of page to begin printing

The default setting is 3.

NUL: NULl directive

NUL does nothing.

OSW:

Replaced by OVD=n (describes how the Device hardware handles OVerbar characters). Don't fool with OVD.

OVB=1/0: OVerBar print on/off

Replaces old directive DOV=0/1 (Delete OVerBars). The new directive applies only to 8-bit overbars.
Default setting is 0 -- delete overbars -- for all devices.

OVD=n: OVerbar description for Device

Replaces OSW, POV, and SOV, and describes how the device hardware handles overbar characters.
Don't fool with OVD; it will disappear soon anyway.

PBI=n: Paragraph Body Indent NOT IMPLEMENTED

Guarantees exactly n spaces (after lms and indentation) before all lines execpt first line of each statement.

PEL: Paginate at End of Line

This directive replaces REL. The old form was REL=1 -- the new form is PEL.

PES: Paginate at End of Statement

When the entire statement (including statement number, signature, and/or picture) has been output, a new page is begun.
It is suggested that this directive be used in almost all places where the RES directive is now being used.
If you are using SNF and/or SGF then you will probably want the statement number and/or signature to be printed on the same page as their statements. If SNF and SGF are not being used and the RES is the last thing in its statement, there will be SCR blank lines at the top of the body area of the next page. Thus it would seem that the only time someone would want to use a RES would be to paginate in the middle of a statement or to get a blank page by having a RES immediately precede a PES at the end of a statement.

PGN=n: current PaGe Number

The page number that would appear on the current output page. The default setting is such that the first output page would be number 1.

PGP=n: Verticle Position of the PaGe number

Meaning of n changed. n used to be the number of lines up from the page bottom to put the page number, but is now the number of blank lines to insert between the bottom of text body area and the line that is to contain the page number. Thus the page number will be printed in line MLN + PGP + 1 of the page,
This will allow the changing of the text body size (MLN) without having to also change PGP.
The default setting is still 5. New pages will look like old pages.

PIC=1/0: PICture print on/off

Replaces old directive DPV=0/1 (Don't Produce Vector output). Default setting is 1 -- print pictures -- for the printer and 0 -- don't print pictures -- for all the other devices.

PIN=n: Paragraph INdent NOT IMPLEMENTED

Guarantees exactly n spaces (after lms and indentation) before first line of each statement (DLS=1, LSP=n doesn't work, because DLS deletes spaces following a CR).

PLN=n: number of LiNes to a Page

Includes header, body, and page number areas.
The default setting is 66.

PLO=n: Paginate for each Level n statement

PLO can now be set to any number n -- which means that all statements of level n or higher will cause a page break to occur if the statement is not the head of its sublist (which I think is what is wanted).
The default setting is still zero.

PNO=n: Page Numbering Option

This combines the old option NSW=0 (no page number) and the directives ROM (Roman numeral page numbers or not) and FNC (upper or lower case for Roman numeral page numbers).
Default setting is 1 -- arabic page numbers.
The four possible settings are:

- = 0: no page number
  replaces NSW=0
- = 1: arabic numeral page numbers
  replaces ROM=0
- = 3: lower case Roman numeral page numbers
  replaces ROM=1 and FNC=3 or 5
- = 4: upper case Roman numeral page numbers
  replaces ROM=1 and FNC=1 or 4

POV:

Replaced by OVD=n (describes how the Device hardware handles OVerbar characters). Don't fool with OVD.

PSH=n: Page Show

Only produce output for page n, but format and scan all the other pages for directives.
The default setting is zero, which means print all pages.
This would be nearly equivalent to beginning the file with a TYP=0 and having a TYP=1 immediately before page n and a TYP=0 immediately after.
Note that there can be several PSH's in a file and if put in the right places, one could get any number of single pages as output.

PST=1/0: Paginate when STatement will not fit on current page

The algorithm for estimating the number of output lines a statement will take up has been changed and (hopefully) is now much more accurate. (The OP uses the same estimate of the statement's output length as for WLN, so it may not always work.)
The default setting is 0 -- off.

PSW=1/0: Pagination SWitch on/off

If PSW=1 then the directives involved with page numbering (PGP, PNO, and HJP), dashes at the end of a page (NDH and DSH), stop code at the end of a page (SSW and STP), verticle size of the page (PLN), getting to the top of the next page, spacing down from the top of the next page (NTP), and the running head (HSW, HED, HJH, and HLN) will be executed.
The default setting is 1.

RDD: Restore Default-Default values NOT IMPLEMENTED

Restore built-in directive default values -- i.e., ignore NLS/TODAS viewspec paramenters.

REL=1: page Restore at End of Line

Name changed to PEL (Paginate at End of Line). The old form was REL=1, the new form is PEL.

RES: page REStore here

Causes a page restore (new page) at the point the directive occurs.
It is suggested that the new directive PES (Paginate at End of Statement) will do what you really want done instead of using RES. See the description of that directive.

ROM=1/0: ROMan page numbering, and

The old directives NSW, ROM, and FNC have been combined into options of the new directives PNO (Page Numbering Option) and HJP (Horizontal Justification of Page Number).

RTJ=1/0: RighT Justification on/off

Superseded by HJB (Horizontal Justification of Body):
RTJ=1 is now HJB=8 RTJ=0 is now HJB=1

SCR=n: number of Carriage Returns to separate Statements

After printing the last line of a statement, the OP will output SCR*NBL carriage returns.
The "default setting" is determined by the NLS blank line Viewspec. If blank lines are on, then SCR is initialized to two. Otherwise it is initialized to one.
Watch out for this initialization. It is the only one that under normal conditions will result in something different from the old PASS4.
Setting SCR to zero will no longer work correctly.

SGF=n: SiGnature Format

Its setting has a similiar meaning to that of SNF, i.e., if n > 0, print each statement's signature (date, time, and initials of the person when the statement was created or last alterred) right justified to column n after the last of the text of the statement has been printed.
The "default setting" is determined by the NLS Viewspecs in force at the time the file is output thru the Output Processor. If signatures are on and blank lines are on, then SGF is set to 60; otherwise it is set to zero -- this is the same convention as in NLS.

If SCR*NBL = 1, the Output Processor will attempt to put the signature in the last line of the statement. If the signature would "overlap" the text of the statement or the statement number, then it will put the signature in a blank line following the statement. A blank line will be forced, if necessary, to accomodate the signature (before the statement number was not printed if SCR*NBL=1 and the statement number overlapped the last line of text).
A convention will be followed that if SCR > 1, then the signature will be forced onto a blank line following the last line of its statement -- it will not go on the same line as the last line of its statement even if it wouldn't "overlap".
If both SNF and SGF are set and they "overlap" each other, then the statement number has precedence (the signature will be printed on the next line).

Two things "overlap" if there is not at least one space between the ends of the things. There are 20 characters in a signature.

The signature and statement number will be printed no matter what SCR and NBL are. However the lines occupied by SGF and/or SNF are subtracted from SCR*NBL -- there won't be SCR*NBL blank lines following the signature and statement number unless they are both printed on the same line as the last line of the text of their statement.
If the signature is printed on a line following the statement, the directive LMS (Left Margin Set) will not be effective for that line so that it will be possible to get the signature printed in the left margin. The amount of indentation for a statement has no affect on the placement of the signature. This is a different convention than was used before with SNF.

If 0 < n <= 20, the signature will be printed flush against the left edge of the page (there are 20 characters in a signature).

The signature (and statement number) will always go on the same page as the last line of its statement (unless there is a RES in the statement).
The bugs that occured before with SNF when the line containing the statement number was supposed to be centered or the line contained nothing or nothing but but blanks will not occur.
SGF may be used in conjunction with the directive MCH, which sets the right margin for the body of the printout. SGF is not constrained by the setting of MCH -- it can be larger.

SHU=n: output code for a SHift to Upper case

Do not use

SHD=n: output code for a SHift to Lower case

Do not use -- replaces FSD

SKP=1/0: SKiP on/off

Now while SKP is on, directives (except SKP) won't be executed (they used to be).
The default setting is still 0.

SNA=1/0: Statement NAmes print on/off

Replaces old directive DPN=0/1 (Don't Print statement Names)
Default setting is 0 --'t print statement names.

SNB=1/0: Statement NumBers print on/off

Replaces old directive DSN=0/1 (Delete Statement Numbers)
Default setting is 0 -- don't print statement numbers.
Note that SNB is entirely independent of SNF.

SNF=n: Statement Number Format

This works the same as it did before except that a few bugs and shortcomings will no longer happen:

if n > 0, print each statement's statement number right justified to column n after the last of the text of the statement has been printed.

The default setting is zero, except for Device Teletype where it is 72.

The Output Processor will attempt to put the statement number in the last line of the statement. If the statement number would "overlap" the text of the statement, then it will put the number in a blank line following the statement. A blank line will be forced, if necessary, to accomodate the statement number (before the number was not printed if SCR*NBL=1 and the number "overlapped" the last line of text).

If both SNF and SGF are set and they "overlap" each other, then the statement number has precedence (the signature will be printed on the next line).

Two things "overlap" if there is not at least one space between the ends of the things.

The statement number and signature will be printed no matter what SCR and NBL are. However the lines occupied by SGF and/or SNF are subtracted from SCR*NBL -- there won't be SCR*NBL blank lines following the statement number and signature unless they are both printed on the same line as the last line of the text of their statement.

The statement number (and signature) will always go on the same page as the last line of its statement (unless there is a RES as the last thing in the statement).

The bugs that occured when the line containing the statement number was supposed to be centered or the line contained nothing or nothing but but blanks will not occur.

If the statement number is printed on a line following the statement, the directive LMS (Left Margin Set) will not be effective for that line so that it will be possible to get the statement number printed in the left margin. The amount of indentation for a statement has no affect on the placement of the number. This is a different convention than was used before.

If n = 1, the statement number will be printed flush against the left edge of the page.

SNF may be used in conjunction with the directive MCH, which sets the right margin for the body of the printout. SNF is not constrained by the setting of MCH -- it can be larger.

SOV:

Replaced by OVD=n (describes how the Device hardware handles OVerbar characters). Don't fool with OVD.

SSW=1/0: Stop code SWitch

You can get a stop code inserted at the end of each page (for mats - normally only for flex).
The default setting is 0 -- don't do it.

STP=n: output code for a STOP code

Do not use -- replaces FSC

TAB: Output a TAB

Name changed to GTB (Generate a TaB)

TAB=n: TABs -- what to do with them

This replaces parts of the old directives TAL (TAb Algorithm), TSP (Tab SPace), and TSW (Tab SWitch) and straighten them out.
The default setting is 1.
The three possible settings will be:
- = 0: delete tabs
- = 1: keep tabs
- = 2: replace tabs by a single space

TAL=n: Tab ALgorithm,

Superceded by TAB directive.

TBD=n: TaB description for Device

Describes how the device hardware handles tab characters, and replaces the rest of TSW, TAL, and TSP. Don't fool with TBD. It will disappear soon anyway.

TLN=n: Truncate to n LiNes

Will work the same as the NLS T Viewspec.
The "default setting" is the NLS T Viewspec at the time the file is output thru the Output Processor.

TMA=n: TeMporary A

TMA is not used by the Output Processor. It for use by user -- for instance in IF clauses.

TMB=n: TeMporary B

(same as for TMA)

TMC=n: TeMporary C

(same as for TMA)

TMD=n: TeMporary D

(same as for TMA)

TSP=n: Tab SPace, and

Superceded by TAB directive.

TST: TabSTop array

An array directive which is used to determine where the tab stop settings are.
This is a bit array stored in six words (144 bits). The ith bit corresponds to the ith column. The first bit in the array is consedered to be number zero. The first word in the array is also number zero.
A one bit indicates a tab stop and setting a position to 0 will clear a tab stop.
An example: TST[0]=040000000B and TST[2]=00002000B
will set tabstops in the 3rd and 61st columns; clear any previously existing tabstops in columns 1, 2, 4 thru 23 inclusive, 48 thru 60 inclusive, and 62 thru 71 inclusive; and leave in their previous state columns 24 thru 47 inclusive and 72 thru 143 inclusive.
Upon activation of the Output Processor, the array TST is initialized according to the tab stops set in the NLS Viewchange Parameters.

TSW=1/0: Tab SWitch on/off

Superceded by TAB directive.

TYP=1/0: TYPe switch on/off

Do not output lines from the line which contains TYP=0 up to the line which contains TYP=1, but continue doing directive recognition and formatting.
The OP only recognizes the directive after a "line" has been formatted and is ready for output, so both TYP=0 and TYP=1 become effective at the beginning of the output line in which they would fall. So watch out.
The default setting is 1.

UBD=n: UnderBar description for Device

Replaces USW, UPR, and USP, and describes how the device hardware handles underbar characters. Don't fool with UBD; it will disappear soon anyway.

UBR=1/0: UnderbaR print on/off

Replaces old directive DUB=0/1 (Delete Underbars). The new directive applies only to 8-bit underbars.

Default setting is 0 -- delete underbars -- for the printer and teletype and 1 -- print underbars -- for all the other devices.

UPR:

Replaced by UBD=n (describes how the Device hardware handles UnderBar characters). Don't fool with UBD.

USP:

Replaced by UBD=n (describes how the Device hardware handles UnderBar characters). Don't fool with UBD.

USW:

Replaced by UBD=n (describes how the Device hardware handles UnderBar characters). Don't fool with UBD.

WLN=n: Widow LiNes

Number of lines of a statement guaranteed to be output on the next page if the statement would not all fit on the current page. The "guarantee" is like many guarantees these days. The algorithm for estimating the number of output lines a statement will take up has been changed and (hopefully) is now much more accurate.
The default setting is still 2.

G. COMMENTS

Directives can now appear in the string given in the HED directive. They will be executed each time the running header is printed.
It is no longer possible to define new directives. This feature will reappear in the next version.

Setting SCR to zero will no longer work properly -- it never worked very well anyway. This feature may reappear in the next version.

The "non-explicit pagination" thing

A "non-explicit pagination" occurs when:

1) the body area is full -- line MLN has been printed
2) because of the WLN (Widow LiNe) directive
3) because of the PST (Paginate when current STatement won't all go on current page) directive

Whenever a non-explicit pagination occurs, the Output Processor will throw away all immediately following lines that consist of only a carriage return (and are to go in the body area). Also, an "explicit pagination", i.e. due to the directives RES (REStore), PEL (Paginate at End of LINE), PES (Paginate at End of Statement), GRB (GRaB), or PLO (Paginate for each Level n statement), will be ignored if executing them would cause a blank page immediately following a non-explicit pagination or with only (thrown away) blank lines intervening.

If there are two explicit paginations (if the user really does want a blank page) following a non-explicit pagination, then the second one will be executed.

Blank lines following an explicit pagination are not thrown away.

Bugs that won't happen anymore:

It is possible to have an unlimited number of HED directives and now each new one will indeed change the running head.

Tabs on the Dura didn't work correctly.

Centering didn't always work correctly.

Page numbers weren't centered correctly.

A bug that's still there:

The Output Processor and Quickprint and TODAS and Create Display do different things with tabs. Create Display apparently has a bug. TODAS apparently uses a slightly different algorithm. The Output Processor and Quickprint do what they they think they should (I think they do the same thing). All of this may be straightened out soon.

```
':5238', 12/08/70 1951:29 JCN ; .DPR=1; ':JRNLNOPUG', 12/08/70 1400:20 WLB
;
|   .   |   .   2   .   3   .   4   .   5   .   6   .   7
                WLB 12/04/70
          (('S/'s) "witch"); .DPR=0;
```

Proposal for Baseline for IMLAC development

On 940. 1

On the 940, most of the development of IMLAC software will be oriented towards familiarisation with the IMLAC, development of bootstrap tools for current and later use (compilers, loaders, utilities, etc.) and utilising the power of the IMLAC on the 940 as much as possible (without an undue amount of work which is not transferable to the 10). 1a

The software which is currently being pursued towards this end is mainly in the form of a display/line driver which will allow use of the mouse andd keyset, with a possible future provision for decoding and transmitting bug selections in a simple manner which might be useful by TODAS. 1b

On PDP10. 2

Work towards a fully implemented NLS on the IMLAC seems to be a realistic and worthwhile goal. 2a

Initially, the IMLAC would allow the 10 to perform all or most of the command parsing and execution. 2b

Following the initial system, logic should be included in the IMLAC to allow it to assume a growing part of the command parsing and control, and perhaps even some of the simple command execution (e.g. local text editing). 2c

Work in this direction is justifiable in that the IMLAC could serve as a 'testbed' for developing techniques for distributing NLS to sites over the ARPA network. 2c1

':5239', 12/08/70 1957:59 JCN ; .DPR=1; :BASEIM, 12/08/70 1209:36 WSD ; To
CHI WKE WHP .DPR=0;

Conceptual Specification of PDP10 Mail System

The mail system on the 10 will difer from that on the 940 in both the method of distribution of messages, and the formats of the files used for distribution. 1

As on the 940, messages going through the mail will be automatically entered into the journal. 2

There will be a special file containing a copy of each message, which will become the journal entry for those messages. 2a

This file may be publicly read, but not written. 2b

Instead of being sored in special files, however, the messages will be stored in NLS files. 3

Each user will have a mail file. 3a

When a user is sent a message, a copy of that message, with a header added as on the 940, will be inserted as statement 1 in his mail file. 3b

Thus the most recent message in the file is statement, the next most recent is 2, etc. 3c

When a user enters NLS, his mail file is automatically loaded instead of the current "Dummy" file. 3d

This may well be the most controversial part of this plan. 3d1

It is based on the following assumptions and thoughts. 3d2

A good reset file command must be afailable. 3d2a

The "dummy" file tells us very little in the way of information, and it is difficult to envision losing a great deal by replacing it with the mail file upon NLS entry. 3d2b

It might be argued that not having a dummy file requires the user to execute a reset file which would otherwise be unecessary, to which I offer the following alternatives: 3d2b1

Have the display be that of the mail file, but the file loaded be a dummy file. Any attempt at editing would cause the display to bbe recreated with the dummy file. 3d2b1a

This means that a user would have to explicitly load his mail file in order to edit it. 3d2b1a1

Change it so that the mail file is loaded only when there is new mail. 3d2b1b

This would tend to reduce the utilty of being able to edit the file, set reminders to oneself and have them be present, etc. (i.e. it makes it so that one must go through the mail system to use the mail file, which I think is more of a restriction thatn we want) 3d2b1b1

Require that a special command (e.g. Save Mail) must be typed as the first command to NLS in order to prevent having the mail file replaced by a dummy file upon the first editing command. 3d2b1c

I think I would favor this for a solution. 3d2b1c1

This scheme will hopefully allow the user greater freedom in editing and disposing messages addressed to him, in addition to allowing nim to use the mail system as a 'prompter'. 4

One possibility which this proposed mailsystem allows is that the user maintain his personal file directory in his mail file 5

This would allow him immediate access to it upon entering NLS. 5a

It is anticipated that the implementation of the mail system on the 10 will be done at a high level, with some suitable nigh level NLS programming language. 6

':5240', 12/08/70 2004:19 JGN ; .DPR=1; :IOMAIL, 12/08/70 1020:13 WSD ;
.DPR=0;

Proposal For Automatic Journal On PDP10

Appearance to user. 1

Thee Auto entry system for the Journal will be called from NLS/TODAS either by a sppecific command, or as a super processor, the main requirement being that it be convenient, and the user need not leave NLS. 1a

Syntax of entry: 'Submit (FILNAME/'Deferred) To Journal # (NUMBER/EMPTY) (CA/CENTERDOT ATTRIBUTE LIST) 1b

ATTRIBUTE LIST = $(ATTRIBUTE (CA/CDOT)) 1b1

ATTRIBUTE = ('Distribute to NAME-LIST/ 1b2

'Author = NAME-LIST/ 1b2a

'Title = "STRING '"/ 1b2b

'Comments = "STRING '"/ 1b2c

'Expedite) 1b2d

IDENT = INIT/NAME/GROUP NAME/OTHER 1b3

NAME-LIST = IDENT $(', IDENT) 1b4

Semantics: 1c

FILENAME/Deferred 1c1

FILENAME identifies the file to be submitted to the journal. 1c1a

It is assumed to be the equivalent of a colon file on the 940. 1c1b

There are instances in which a user wishes a number for a document which is to be entered into the journal at a future time. 1c1c

This need is filled by the Deferred option, which allows the user to not specify a file name, but get a number all the same. 1c1d

(NUMBER/EMPTY) 1c2

This specifies the journal number which the entry will have. 1c2a

If the number is explicitly typed, then it is assumed that it is a number which has been previously assigned with he Deferred option. 1c2b

If it has not bpeen so assigned, then the number is illegal, and an error occurrs. 1c2b1

Otherwise, a number is supplied to he user for the document. 1c2c

This number will be the permanent accession number of the document. 1c2d

ATTRIBUTES 1c3

The attribute secton allows a user to specify certain information which is relevent to the entry being made to the journal. 1c3a

This section has no meaning if the Deferred option is being used. 1c3b

Although if we think about it for a while, we might be able to find a use. 1c3b1

The information specified in the attribute section will be permanently reflected in the header of the entry being made into the journal. 1c3c

Distribute. 1c3d

This allows the submiitter to specify a distribution list for hard copies of the document being entered into the Journal when it is published. 1c3d1

The information will be reflected in the header by text of the form: 1c3d2

"Send To:" DIST LIST 1c3d2a

Author 1c3e

This allows the submitter to specify the authors of the document being submitted. 1c3e1

It may b reasonable to assue that the submitter is th author unless otherwise stated. 1c3e1a

The information will be represented in the header by the text "Author:"NAME-LIST lc3e2

Title. lc3f

This allows a user to specify a title which he wishes printed at the start of each pag in the hard copy version of the journal entry. lc3f1

This command overrides any previous title statements (.HED directives) which may have been in the document. lc3f2

The imformation is represented in the header by an appropriate .HED directive. lc3f3

Comment. lc3g

This allows the submitter to attatch a commnt to th header of the candidate file. lc3g1

The comment appears in the header statement in the form: lc3g2

"Comment: STRING lc3g2a

Expedite. lc3h

This allows the submitter to specify that the hard copy distribution of the document is to be expedited. lc3h1

This information is represented in the header statement by the word "EXPEDITED". lc3h2

When All of the submission information hs been specified, and a CA typed, a carraige return is typed, and the work to journalise the document is initiated. 1d

When the docunment has been journalised,, "Done" is typed, and the user is left in a position to submit another document or exit. 1e

Implementation outline. 2

FILENAME 2a

Read filename and open file, to make sure it is there. 2a1

Get user name and set up text string with full file name somewhere 2a2

If Deferred, then set flag 2a3

NUMBER 2b

If literal number is entered, load number file and check that the number has been pre-assigned to the subbmitter 2b1

What about when someone gets a number, and then asks somwone else to submit the document for him?? 2b1a

The submitter needs some way to pretend to be someone else 2b1b

Otherwwise, load number fle and remove number from available list, and put it into in use list 2b2

If this is not a deferred eentry, set up author name (if doing it automatically) 2b3

A note on number file 2b4

As curretly envisioned, the number file could simply be an NLS file, with four statements or (possibly) brnches. 2b4a

(1) Available list 2b4a1

Contains a list of te numbers which are available for use for te journal (or other documents, if we want a universal number allocator). 2b4a1a

(2) In-use list 2b4a2

Numbers which have been taken, but the transaction to which they belong has not yet been completed. This allows a method of recovery of numbers which have been lost in crashes. 2b4a2a

(3) Assigned list. 2b4a3

A list of all of the numbers which have been assigned 2b4a3a

(4) Deferred list 2b4a4

Contains list of numbers which have been

'Pre-assigned' with the identification of the persons (or groups, etc.) to which they have been pre-assigned. 2b4a4a

ATTRIBUTES. 2c

If not deferred, then: 2c1

Parse attribute statements, and set up text areas (could be named statements) per: 2c2

Distribute 2c2a

(distribution) Send To $(INIT/NAME/GROUP) 2c2a1

Author 2c2b

(Author) Author: $(INIT/NAME/GROUP) 2c2b1

Comment 2c2c

(Comment) Comment: STRING 2c2c1

Title 2c2d

(Title) 2c2d1

Before copying ths to the header statement, a

':5241', 12/08/70 2012:03 JCN ; .DPR=1; :1OJOURNAL, 12/08/70 1016:33 WSD ;
TO DCE, JCN .DPR=0;

-08DEC70 BLP 5242

Partial Description of the 'Universal Output Machine'

By "Universal Output Machine" is meant the one, non-real device to which the Portrayal Generator outputs. 1

By "Partial Description" it is meant that the following is certainly not complete -- particularly as regards pictures, updating portions of a "display", and the map between the input and output that enables such things as bug selection -- and it is subject to change. 2

The Portrayal Generator is the code that will perform all of the functions now done by Create Display, the Output Processor (PASS4), Quickprint, and the TODAS Print Command. 3

The Universal Machine is realized by a collection of software called the Post Processor. The division of coding between TSS, NLS, and remote sites on the Network for any particular, actual hardware device is left unspecified at this time. 4

I assume that for each new device, usually most of the code would at first lie within NLS. Then some portion could be moved to TSS if deemed desirable. 4a

I assume that the Network is seen by the Post Processor as at most two devices -- a storage-display-like thing, i.e., a thing that can be told to update a portion of an "area", and a teletype-like thing. Hopefully the Network will be seen as only one device. 5

The way the Portrayal Generator works in order to run only one device is a whole nother story. It looks like the only device dependence that it must contain is: 6

information about the horizontal space each character will take up 6a

an initial setting of some variables -- most of which are currently included in NLS View Parameters and/or Output Processor directives 6b

Anyway -- the Universal Output Machine will have the following characteristics: 7

64K x 64K addressable points in a "display"/"page" 7a

the point 0,0 is in the upper left corner 7a1

it accepts 7-bit ASCII character codes 7b

character codes for a tab, carriage return, line feed, multiple space, or multiple carriage return are NOT included 7b1

256 "fonts" 7c

(256 different results from the same character code) 7c1

256 character sizes 7d

32 levels of intensity for the piece(s) that make up a character and/or a vector 7e

(the strokes, dots, whatever) 7e1

32 "widths" for the piece(s) that make up a character and/or a vector 7f

7g

The above numbers were chosen because no device known to me has even half that number. The things were chosen because I think they could completely characterize all the capabilities of all the devices I know of. 7h

7i

The Post Processor has at least the following capabilities: 8

"display" a "line segment" 8a

a line segment consists of the following: 8a1

n 7-bit ASCII character codes 8a1a

x coordinate of first character 8a1b

relative to 0,0 point of the "area" where it is displayed 8a1b1

y coordinate of the line segment 8a1c

relative to 0,0 point of the "area" where it is displayed 8a1c1

font 8a1d

character size 8a1e

horizontal character spacing 8a1f

either a constant or a table 8a1f1

intensity 8a1g

"width" 8a1h

"display" a "picture" 8b

a picture consists of the following: 8b1

x and y coordinates of bottom left corner of the picture 8b1a

relative to 0,0 point of the "area" where it is displayed 8b1a1

n vectors 8b1b

a vector consists of the following: 8b1b1

x and y cooridinates of both endpoints 8b1b1a

relative to bottom left corner of the picture 8b1b1a1

maybe things like rectangles and arcs 8b1c

n labels 8b1d

a label is the same as a line segment except that its coordinates are relative to the bottom left corner of the picture 8b1d1

do control stuff 8c

initialize the device 8c1

close the device 8c2

get to top of next "page"	8c3
stop code	8c4
define an "area"	8d
replace a specified line segment	8e

':5242', 12/09/70 1348:39 JCN ; .DPR=1; ':JRNL1', 12/08/70 2214:51 BLP ;
Copies to CHI, WKE, WHP, Duvall, Melvin, and KEV. .DPR=0;

. 09DEC70 BLP 5243

Partial Description of the 'Universal Output Machine'

By "Universal Output Machine" is meant the one, non-real device to which the Portrayal Generator outputs. 1

By "Partial Description" it is meant that the following is certainly not complete -- particularly as regards pictures, updating portions of a "display", and the map between the input and output that enables such things as bug selection -- and it is subject to change. 2

The Portrayal Generator is the code that will perform all of the functions now done by Create Display, the Output Processor (PASS4), Quickprint, and the TODAS Print Command. 3

The Universal Machine is realized by a collection of software called the Post Processor. The division of coding between TSS, NLS, and remote sites on the Network for any particular, actual hardware device is left unspecified at this time. 4

> I assume that for each new device, usually most of the code would at first lie within NLS. Then some portion could be moved to TSS if deemed desirable. 4a

I assume that the Network is seen by the Post Processor as at most two devices -- a storage-display-like thing, i.e., a thing that can be told to update a portion of an "area", and a teletype-like thing. Hopefully the Network will be seen as only one device. 5

The way the Portrayal Generator works in order to run only one device is a whole nother story. It looks like the only device dependence that it must contain is: 6

> information about the horizontal space each character will take up 6a
>
> an initial setting of some variables -- most of which are currently included in NLS View Parameters and/or Output Processor directives 6b

Anyway -- the Universal Output Machine will have the following characteristics: 7

64K x 64K addressable points in a "display"/"page" 7a

the point 0,0 is in the upper left corner; positive x direction is rightward, positive y direction is downward 7a1

it accepts 7-bit ASCII character codes 7b

character codes for a tab, carriage return, line feed, multiple space, or multiple carriage return are NOT included 7b1

512 "fonts" 7c

(256 different results from the same character code) 7c1

256 character sizes 7d

the ability to draw lines 7e

32 levels of intensity for the piece(s) that make up a vector and/or a character 7f

(the strokes, dots, whatever) 7f1

32 "widths" for the piece(s) that make up a vector and/or a character 7g

7h

The above numbers were chosen because no device known to me has even half that number. The things were chosen because I think they could completely characterize all the capabilities of all the devices I know of. Other capabilities that some machines have are: true italic, true boldface, true lightface, extended, condensed, oblique, and rotated characters. All of these could indicated by the font number. Note that it could be possible to fake boldface and lightface by intensity and/or "width" control. 7i

Partial Description of the 'Universal Output Machine'

The Post Processor has at least the following capabilities: 8

"display" a "line segment" 8b
- a line segment consists of the following: 8b1
  - n 7-bit ASCII character codes 8b1a
  - x coordinate of first character 8b1b
    - relative to 0,0 point of the "area" where it is displayed 8b1b1
  - y coordinate of the line segment 8b1c
    - relative to 0,0 point of the "area" where it is displayed 8b1c1
  - font 8b1d
  - character size 8b1e
  - horizontal character spacing 8b1f
    - either a constant or a table 8b1f1
  - intensity 8b1g
  - "width" 8b1h
    - 8b1h1

"display" a "picture" 8c
- a picture consists of the following: 8c1
  - x and y coordinates of bottom left corner of the picture 8c1a
    - relative to 0,0 point of the "area" where it is displayed 8c1a1
  - n vectors 8c1b
    - a vector consists of the following: 8c1b1
      - x and y coordinates of both endpoints 8c1b1a
        - relative to bottom left corner of the picture; positive x direction is still rightward, but now positive y direction is upward 8c1b1a1
      - whether the vector is to be drawn as a solid, dashed, or dotted line and information specifying the spacing in the two latter cases 8c1b1b
  - maybe things like rectangles and arcs 8c1c
  - n labels 8c1d
    - a label is the same as a line segment except that its coordinates are relative to the bottom left corner of the picture 8c1d1
      - 8c1d1a

do control stuff 8d
- initialize the device 8d1
- close the device 8d2
- get to top of next "page" 8d3
  - 8d3a

probably several functions like: 8e
- define an "area" 8e1
- replace a specified line segment 8e2

Partial Description of the 'Universal Output Machine'

':5243', 12/09/70 1736:07 JCN ; .DPR=1; ':JRNL1', 12/09/70 1630:27 BLP ;
Copies to CHI, WKE, WHP, Duvall, Melvin, and KEV. .DPR=0;

NOTES ON CHANGES TO THE NLS SYSTEM DURING THE TRANSFER TO THE TEN: Distribute to WSD

ROUGH DRAFT: TRANSFRE OF NLS TO TEN 1

Internal NLS/TODAS Modifications 1a

Reorganization of NLS/TODAS 1a1

NLS/TODAS was reorganized to allow the user access to the command parser for NLS or TODAS and the parameter specification and executor for each command--this also makes possible the separation of NLS/TODAS command specification from the (core-nls) file manipulative capabilities, with perhaps a network in between. 1a1a

New Capabilities 1a2

File System 1a2a

The file system implimented in the TEN NLS system is functionally similar to that of the 940 NLS system, but allows more core space for file blocks, applies aging to those file blocks, and allows for more than one file. 1a2a1

In addition, the working copy of the 940 system has been replaced by a 'partial copy' which contains only the blocks of the original file which have actually been changed by the user. The partial copy is obviously associated with the source file and is not reused when the user modifies another file. 1a2a1a

Also, only one user may have modification access to a file source at one time (new monitor calls's). The partial copies are retained until the user does an output file or explicity deletes the partial copy. 1a2a1a1

Those partial copy blocks which have changed since the second to the last checkpoint are automatically copied to the oldest of two checkpoint files, which are also associated with the source file. 1a2a1b

Display Areas 1a2b

Unlike the 940 NLS system, the TEN NLS system allows the user to subdivide the text area of his screen

into rectangular, non-overlaping display areas. The user is provided with commands to split extant display areas into two display areas, move the boundries of display areas, and erase the display from a display area. The user may display portions of several files in his display areas and may freely edit across the display area boundries. The user may also have a list of frozen statements (from any currently open file) associated with each display area. 1a2b1

The position of the cursor will be used to resolve ambiguities related to the use of display areas. 1a2b2

For example, if the user has just fully specified a 'Jump to Item' command, the new display will appear in the display area which contained the cursor when the final CA was entered. The viewspecs of this display area will then be those of the display area which contained the cursor when the first selection was made plus any modifiers typed by the user. 1a2b2a

If a marker is used the file which is being displayed in the display area which contained the cursor when the right mouse button was released is searched for the marker, then the rest of the files are searched in the order in which they were opened (this latter is still being debated). 1a2b2b

Similarly, if a statement name is used, the position of the cursor when the final CA is entered will be used to determine which file is to be used. 1a2b2c

If the view specs are changed using the 'Viewset' command (or shift-3 of the keyset), the viewspecs of the display area which contains the cursor when the 'V is typed (or the mouse buttons depressed) are displayed in the viewspec area and are used as the initial values. When the user types the final CA (or releases the mouse buttons), the viewspecs associated with the display area which containes the cursor when this final action takes place will be updated by these new viewspecs. 1a2b2d

NOTE: In addition, in specifying each command,

the viewspec area display will be updated to reflect those of the display area which contained the first selection. 1a2b2d1

NOTE: One may see the viewspec display of any display area by simply depressing and releaseing the shift-3 mouse buttons while the cursoer is in the desired display area. Note also, that if instead of immediately releasing the buttons, the user moves the cursor to another display area first, the new display area will receive the viewspecs of the old. 1a2b2d2

If a file is not being displayed anywhere on the screen, NLS will close the file. 1a2b3

New A-string Routines 1a2c

Additional string manipulation routines. 1a2c1

Modified or Deleted Capabilities 1a3

Structure manipulation 1a3a

Modified to allow for cross-file editing. 1a3a1

Statement destruction 1a3b

Statement destruction routines were modified to combine free space in the statement data blocks to allow for better method of using this free space by the statement construction routines. 1a3b1

Statement construction 1a3c

Statement construction routines were modified to make better use of the free space in a statement data block, to make use of the field operations, and to allow for string construction in a-strings as well as statements. 1a3c1

Text edit eupport 1a3d

Modified to allow for editing of strings as well as statements. 1a3d1

Literal feedback 1a3e

Aplit and associated routines completely rewriten. 1a3e1

Input feedback support 1a3f

modified to make use of field operations, and to make them more consistent. 1a3f1

NLS input routines 1a3g

Character input routines were reorganized, with the more basic routines modified to account for the tennex system. 1a3g1

Markers 1a3h

Markers were called pointers on the 940 system. The marker lookup routines were modified to allow for multiple file use of markers, in that a reference to a marker is not restricted to the markers in the current file, but rather to the markers of all of the files which are currently open. 1a3h1

Calculator system 1a3i

Modified to make use of the double-word arithmetic instructions of the ten. 1a3i1

Substitute 1a3j

Semantically the same but completely rewritten and reorganized. 1a3j1

Output processor 1a3k

Similar to the output Processor (pass4) now available on the 940 with the addition of new directives and a TREE META generated directive recognizer.
See(parsley, opug) and (parsley, oplan, stage II). 1a3k1

Insert sequential 1a3l

Expanded to incorporate insert QED function for the MAIL system, unless a better solution is found. 1a3l1

Output quickprint 1a3m

As now available on the 940 with a more informative

page header, giving the file name, the date and time of printing, the initials of the printer, and the value of the L and T view specs, if they are not "all". 1a3m1

Content analyzer-analyzer compiler 1a3n

The analyzer compiler is replaced by the L10 compiler, which now includes the capabilities of the old SPL analyzer compiler. The content analyzer will make use of the L10 compiler also. 1a3n1

File compactor 1a3o

Completely rewritten to make use of the multiple file capabilities of NLS/TODAS. 1a3o1

File input/output 1a3p

Load file, Output File, Load (most recent, oldest) Checkpoint, Output Checkpoint commands are either new or completely rewritten to account for the new file system. In addition, an automatic checpointing feature has been added. 1a3p1

Initialization 1a3q

Almost entirely rewriten to account for the tennex system. 1a3q1

Jumping 1a3r

Same with the addition of 'jump to word', a 'use the old value' capability for jump to name, word, and content. and the use of 'J to indicate 'jump to Item' to the jump command parser--this inconsistency can be easily justified by an experienced user. 1a3r1

Parameter specification 1a3s

Almost completely rewritten to take advantage of the added capabilities of L10. 1a3s1

Sequence generator 1a3t

Partially rewritten to account for desireable changes in the sequence generator work area. 1a3t1

Frozen statements 1a3u

The same as on the 940, except that there may be frozen statements associated with each display area and that the frozen statement lists may contain statements from any of the files which are currently open. 1a3u1

Verify (cleanup) 1a3v

File verify replaces file cleanup and is a fast read-only inspection of a user's file. 1a3v1

Bug selection 1a3w

Modified to allow for multiple display area capabilities. 1a3w1

Create display 1a3x

Largely rewritten and reorganized to allow for monitor/network/imlac control of the display, multiple display areas, and eventual replacement by the output processor. 1a3x1

Display support 1a3y

Modified to allow for monitor/network/imlac control of the display. 1a3y1

Message display 1a3z

Modified to allow for addition of messages to extant messages on the screen. 1a3z1

A-string routines 1a3a

Rewritten to make use of the ten's byte accessing capability. 1a3a1

Text editing 1a3aa

Rewritten to be compatible with the intersticial text pointer scheme and with the L10 language. 1a3aa1

TODAS alter 1a3ab

Expanded to include a few new control characters currently available in QED. la3ab1

TODAS input la3ac

The most basic routines were rewritten to be compatible with the tennex system. la3ac1

TODAS control code la3ad

Partially rewritten and reorganized to allow for changes and reorganization of the support routines and to be more (structurally) similar to the NLS control code. la3ad1

File manipulation la3ae

Ring and data block manipulation, core page status table routines, etc were extensively rewritten to implement a more powerful file system. la3ae1

Character readout la3af

Modified to make use of expanded capabilities of the L10 language, the TEN's byte manipulation instructions, and to allow for character read out from strings as well as statements. la3af1

Text pointers la3ag

The use and implimetation of the text pointers was changed to allow pointers to point to the gap between characters (intersticial) rather than to one of the characters. la3ag1

NLS control code la3ah

NLS main control code--command language code was rewritten to account for the replacement of the SPL language by L10 and was reorganized to allow the user access to the parameter specification segment of each command. la3ah1

Data la3ai

Almost completely new--due in large part to the use

of local variables, and the renameing of unclear global variables. 1a3ail

Errors and aborts 1a3aj

The old error routines were slightly rewritten and error and abort routines were added which await the input of a GA from the user. 1a3ajl

Insert QED 1a3ak

Replaced by Insert Sequential. 1a3akl

Keyword system 1a3al

To be replaced later by a more powerful capability. 1a3all

Trails system 1a3am

To be replaced later by a more powerful capability. 1a3aml

Tree display 1a3an

Eliminated because of lack of use. 1a3anl

Capabilities Not Yet Transferred 1a4

Mail system 1a4a

The same or a similar system will be provided. Subsequent development will incorporate it into NLS/TODAS. 1a4al

Merge file (or equivalent) 1a4b

A capability similar to that available now on the 940 will be provided in a cleaner and safer manner. 1a4bl

Dont modify working-copy 1a4c

A capability similar to that available now on the 940 will be provided in a cleaner and safer manner. 1a4cl

Execute status 1a4d

As now available on the 940. 1a4dl

Viewchange 1a4e

As now available on the 940. 1a4e1

Collector -sorter 1a4f

As now available on the 940. 1a4f1

Graphics package 1a4g

A new graphics capability (also available to the calc compiler) which includes 1a4g1

A new data structure, "boxes", "areas", and normal editing of labels. 1a4g1a

Execute TODAS/NLS 1a4h

Commands to allow user to move between NLS and TODAS. 1a4h1

NLS exec text 1a4i

Executable text for NLS, if reasonable. 1a4i1

':5244', 12/10/70 1524:27 JCN ; .DPR=1; :JRNL1, 12/10/70 1032:01 CHI ;
.DPR=0;

Phone Log: Call from Richard S. Brannin

Address: 1

Grumman Data Systems
33 Ogden Ave.
East Williston, N.Y. 11596 1a

office (516) 575-3282; res. (516) 746-1079 (Said, "call me any time of the day or night -- when I get really interested in something, time doesn't make any difference.") 1b

He called after reading several of our publications, to ask for more. Seems very interested in the philosophy of man-effectiveness systems. 2

Concepts such as "levels": materials, energy, information, intelligence, wisdom. He talked at some length about his views of man's developments in control at each level, the effect on a lower level of improvements in a higer one (e.g. more efficient control of lower levels). Hasn't published. Seems quite dedicated. Would like very much to communicate. 2a

Would like ALL OF OUR REPORTS. 3

Now has: 3a

ARC publications sheet (3 pages), up to #19 (July 1970, 7079, NASA) -- i.e. our most up-to-date list. 3a1

They (Grumman) aren't doing anything in this direction yet, although they "have acres of computers." He hasn't been there too long. 4

I forgot to ask him what publications he already had. Should at least send him OSR1, and the last RADC and NASA reports, plus whatever others whose inventory seems adequate. 5

```
':5245', 12/10/70 1538:37 JCN ; .DPR=1; ':JRNLA', 12/10/70 1405:35 DCE ;
.HED="                                                    10DEC70 DCE 5245

Phone Log:  Call from Richard S. Brannin ";
|    .    |    .    2    .    3    .    4    .    5    .    6    .    7
.SNF=72; .MCH=65; .SNB=0; .DLS=1; .SCR=2; .RTJ=0; .PGN=0; .COD[21B]=114B;
.DIR=0;       .DPR=0;
```

Phone Log: Bobrow, Glaser, Barden on NIC NDS Establishment

I placed these phone calls to get final Agent-Liaison assignements. 1

BBN, Dan Bobrow 2

Dan called back while I was out. Talked to WKE. Said that Dan Murphy will be liaison man. Dan will be here today, and can give us name of their Agent. 2a

Jim, Jeanne, Walter, and I should take the opportunity to get acquainted with Murphy, give him some picture of us and of NIC. He'll be here through Friday, apparently. 2b

Case, Ted Glaser 3

John Barden, Room 222, Crawford Hall, Computing and Information Sciences, Case Western Reserve University, (216) 368-4467. 3a

Former lawyer, history professor, technical writer, etc., and also former Dean of University of Cicago, most recently a full professor at Westen Reserve University. 3a1

Just moved into the computer science activity, so isn't technical. Will serve temporarily as both Agent and Liaison; and will administer whatever subsequent re-allocation of tasks that are needed -- like enlisting Agent help, and computer-science liaison. 3a2

John Barden, (216) 368-4467. 4

I called Professor Barden, told him briefly what the NIC and the Network Dialogue System were all about. Said tnat we would forthwith send him his "kit," and for him to pull out and read NIC 4792 when he received it. Gave him his Enterprise number, and told him to call every once in a while to get acquainted. 4a

He is very enthusiastic about participating in all of this -- the Network, the Dialogue, etc. I think that his presence itself will make a very good contribution to the Community. 4b

I gathered that he really has "no technial bckground." He promised tat he would quidkly set up personal liaison with knowledgeable "kids", and would be eager to respond to any communication we sent. 4c

Phone Log: Bobrow, Glaser, Barden on NIC NDS Establishment

If he doesn't call us very often, I think it would be a good practice for us to call him like once a week. Jeanne, might be the best general contact, but Walt, and perhaps Jim, too, might get acquainted.. 4d

```
':5246', |2/10/70 |540:57 JCN ; .DPR=1; ':JRNLB', |2/10/70 |448:0| DCE ;
.HED="                                                    |0DEC70 DCE 5246

Phone Log: Bobrow, Glaser, Barden on NIC NDS Establishment ";
|    .    |    .    2    .    3    .    4    .    5    .    6    .    7
.SNF=72; .MCH=65; .SNB=0; .DLS=1; .SCR=2; .RTJ=0; .PGN=0; .COD[21B]=114B;
.DIR=0;       .DPR=0;
```