BEV KIRK 22-AUG-75 17:27 26324

Documentaion Shopping List for Next NSW Proposal

This is a brief list of ideas for our next proposal. Any comments, suggestions, or selling tactics would be appreciated.

DOCUMENTATION IDEAS FOR NEXT NSW PROPOSAL 1

1. Online Documentation Data Base Maintenance

The Help multi-file data base is an encyclopedic description of NLS tools and NSW procedures designed for quick online how-to reference by the user. To be effective it must be comprehensive and current. Methods to keep the Help data bases up-to-date and keep their extensive cross-referencing accurate are needed. 1a

Index generation and maintenance:
Current experience shows that the automatic creation and maintenance of an alpabetic index in a multifile data base is essential if the online Help is to grow and evolve as planned for NSW tools. This index would be generated from all named statements and perhaps all meaningful words. This needs the automatic link maintenance facility described below. This feature is also necessary for #3, below, "Hardcopy Production from Online Documentation." 1a1

Back links:
It is essential that automatic link maintenance via back links be implemented in order to reduce the tremendous overhead and inherent mistakes in the current procedure for discovering bad links and updating them manually. This should be implemented as a property of a node. See below. Back links have the added benefit of automatic "forward references", a list of back links to what has been written about a document since it was published. 1a2

Comment feature:
The comment feature would allow any arbitrary amount of designated text to dissappear when viewspec capital T is in effect. The text reappears when viewspec capital S is turned on. This feature is important for four separate functions. 1a3

-Making user-invisible comments to other Help writers.
This would do away with the current percent sign convention which requires a special sequence generator or content analyzer pattern since it is not a part of the standard Nls capabilities. 1a3a

-Making output processor directives invisible in Help.
This would do away with the necessity of having two separate directories of files, one with Output Processor (OP) directives and one without. And it would do away with

having to delete directives and update to the second directory whenever a modification is made. 1a3b

-Placing links invisibly next to referenced text.
This is needed to place link syntax in a node to define how that node will be viewed. (See #2, below.) It would also automatically take the user to enriching references. 1a3c

-The comment feature would be one way of implementing the backlink property, where the links to a node are invisibly backlinked from that node. 1a3d

Simple Boolean expression searches:
Once automatic index maintenance is implemented it will provide the added advantage of being the basis for providing more sophisticated and efficient searching capabilities. This will speed searching. It will also allow standard Boolean arguments such as "x AND y OR z AND NOT a" and eventually, natural language typed in (and spoken?) English queries. 1a4

2. Augmenting the Help Command

Several features have emerged as essential in the evolution of the Help command. More effective user control of the information available and a closer integration with other capabilities are two areas that need attention now. 1b

Do feature:
The "do" feature is the capability of having Help execute a command for you or a task consisting of a scenario of commands. Should user specification be necessary in the process, the do feature will tell in English sentences what is expected at every step of the way. This is an active tutorial/example/service which whould make Help much more valuable as a teaching aid and provide a new service as a task doer. It would be implemented by writing commands branches which appear as menu items under Help descriptions. A special symbol placed after the right anglebracket of a link will process the commands branch. Uparrow or ^, indicates that Help is to process the commands in the branch addressed by the link. Backarrow or _, indicates that Help is to process the command(s) between the link delimiters. 1b1

Point with mouse to words and lines:
Help would be far more flexible if the Display user could point with the mouse. With this feature the user would be able to simply point to words and lines of text, and Help would display the descriptions of the words pointed to, or the descriptions pointed to by links contained in the lines. 1b2

Outline and verbose description views:
This feature would allow users to specify whether they would like to see a brief, outline view of the description and its menu, or see the full description. 1b3

Descriptions for two levels of users:
A difficulty with the present Help data bases is that the

descriptions must be tailored to a single, mythical, "average user". The result is some descriptions that are too advanced for the very new user and others that are too simple for the more knowledgeable user. This task would be to rewrite portions of the data base to provide two descriptions for major concepts and commands--one for novices and the other for experts. Some minor additions to Useroptions software could be made so that the user could specify whether she wanted to see the simple or more advanced descriptions. The default would be for the novice user. 1b4

References to additional online information:
A complete Help system points users to information of various sorts besides how to use tools. For example, hardcopy documentation, source files, mail indices, directories of people, and program lists could all be made available via Help. 1b5

3. Production of Hardcopy from Online Documentation

Although online documentation is flexible and immeditely accessible for user reference, hardcopy is preferred by some users, and is helpful for some applications. This endeavor would entail a variety of software and writing tasks to make online documentation such as the Help data bases automatically translatable into readable hardcopy. 1c

Features already cited in this document:
Many of the features cited above would aid in making online documentation into hardcopy, including the "Comment feature", "Backlinks" and the "Index generation" features listed under #1; and "Outline and verbose descriptions views" listed under #2. 1c1

Text that would make hardcopy more readable:
With the comment feature, documentors would be able to include textual passages, such as transitions, headings, and references, in the online document that would only be turned on for hard copy. This would help transform the essentially "reference" online documentation into a form which could be read from front to back. 1c2

4. New Documentation

Documentaion of proposed software or software changes. 1d

5. Documentation Maintenance

Existing documentation must be updated to accurately describe

changes in an experimental system. Some of the major documents that require much effort to maintain are: 1e

Help data bases:

Base	Sendmail
Programs	Calculator
Graphics	Useroptions
Publications	Message
Letter	Core
Helpd	

1e1

Documentaion Shopping List for Next NSW Proposal

User manuals:
  Secretarial Functions Guide
  DEX User's Manual 1e2

Systems documentation:
  Tool Suppliers documentation
  FE System Interface documentation 1e3

BEV KIRK 22-AUG-75 17:27 26324

Documentaion Shopping List for Next NSW Proposal

(J26324) 22-AUG-75 17:27;;;; Title: Author(s): Beverly Boli, Kirk E. Kelley/BEV KIRK; Distribution: /ARC-DEV([ACTION]) ; Sub-Collections: SRI-ARC ARC-DEV; Clerk: BEV; Origin: < BOLI, DOCPROP.NLS;3, >, 21-AUG-75 20:23 BEV ;;;;####;

26324 Distribution

Mary Ann Kellan, Andy Poggio, David L. Retz, Jan A. Cornish, Larry L. Garlick, Delorse M. Brooks, Beverly Boli, James E. (Jim) White, Ann Weinberg, Kenneth E. (Ken) Victor, Dirk H. Van Nouhuys, Jonathan B. Postel, Elizabeth K. Michael, David S. Maynard, Karolyn J. Martin, Harvey G. Lehtman, Kirk E. Kelley, Charles H. Irby, Joseph L. Ehardt, Robert Louis Belleville, Don I. Andrews, Richard W. Watson, Douglas C. Engelbart,

BEV KIRK 22-AUG-75 18:43 26325

Documentation Weekly Report

AFM format manual, AFM formatter debugged, Glossary to SRI printing, Xhelp, Base file , proposal ideas.

Documentation Weekly Report

Week ending 8/22/75 1

Bev 1a

This Week 1a1

Worked on Xhelp,Base file. 1a1a

Went over Format SS (Dirk is writing). 1a1b

Wrote Proposal ideas with Kirk for next NSW proposal. 1a1c

With Dirk, sent Glossary to be printed. 1a1d

Went over milestones with Jon. 1a1e

Next Week 1a2

Try to get Sec. Func. Guide printed, including Format SS and Preface. 1a2a

Continue working on Base Help file. 1a2b

Kirk 1b

Done 1b1

Updated the manual for formatting Airforce Manuals. 1b1a

Debugged the AFM volume formatter and formatted table of contents for volumes III, IV, and V. 1b1b

Worked with Bev on essential Help needs list. 1b1c

Do 1b2

Finish getting Help files in order for bringing up 8.5. 1b2a

Write the step by step DPCS procedures for the Air Force Manual. 1b2b

Continue to add bells, whistles, and necessary changes to AFMFormat. 1b2c

Decide for sure whether or not to punt the Letter Program. 1b2d

Continue transfer to Jan of Help and Class-I user programs. 1b2e

Put the current list of development documentation on line. 1b2f

Documentation Weekly Report

(J26325) 22-AUG-75 18:43;;;; Title: Author(s): Beverly Boli, Kirk E. Kelley/BEV KIRK; Distribution: /DIRT([ACTION]) ARC-DEV([INFO-ONLY]) JHB([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT ARC-DEV; Clerk: BEV;

26325 Distribution

Elizabeth K. Michael, David S. Maynard, Karolyn J. Martin, Harvey G. Lehtman, Kirk E. Kelley, Charles H. Irby, Joseph L. Ehardt, Robert Louis Belleville, Don I. Andrews, Richard W. Watson, Douglas C. Engelbart, James H. Bair,
Jonathan B. Postel, Priscilla A. Wold, Rita Hysmith, Pamela K. Allen, Delorse M. Brooks, Elizabeth F. Finney, Beverly Boli, Lawrence A. Crain, Kirk Sattley, Susan Gail Roetter, Robert N. Lieberman, Ann Weinberg, Kenneth E. (Ken) Victor, Douglas C. Engelbart, James H. Bair, Elizabeth K. Michael, Richard W. Watson, Elizabeth J. Feinler, Harvey G. Lehtman, Kirk E. Kelley, Laura E. Gould, Jeanne M. Beck, Dirk H. Van Nouhuys, James C. Norton, Mary Ann Kellan, Andy Poggio, David L. Retz, Jan A. Cornish, Larry L. Garlick, Delorse M. Brooks, Beverly Boli, James E. (Jim) White, Ann Weinberg, Kenneth E. (Ken) Victor, Dirk H. Van Nouhuys, Jonathan B. Postel

BEV 22-AUG-75 18:59 26326

A Resend of Documentation Shopping List for NSW Proposal

This is a resend of earlier item Documentation Shopping List for NSW Proposal. If you are going to print the file out using Output Printer, this file will work best.Other one is fine for online.

DOCUMENTATION IDEAS FOR NEXT NSW PROPOSAL 1

1. Online Documentation Data Base Maintenance

The Help multi-file data base is an encyclopedic description of NLS tools and NSW procedures designed for quick online how-to reference by the user. To be effective it must be comprehensive and current. Methods to keep the Help data bases up-to-date and keep their extensive cross-referencing accurate are needed. 1a

Index generation and maintenance:
Current experience shows that the automatic creation and maintenance of an alpabetic index in a multifile data base is essential if the online Help is to grow and evolve as planned for NSW tools. This index would be generated from all named statements and perhaps all meaningful words. This needs the automatic link maintenance facility described below. This feature is also necessary for #3, below, "Hardcopy Production from Online Documentation." 1a1

Back links:
It is essential that automatic link maintenance via back links be implemented in order to reduce the tremendous overhead and inherent mistakes in the current procedure for discovering bad links and updating them manually. This should be implemented as a property of a node. See below. Back links have the added benefit of automatic "forward references", a list of back links to what has been written about a document since it was published. 1a2

Comment feature:
The comment feature would allow any arbitrary amount of designated text to dissappear when viewspec capital T is in effect. The text reappears when viewspec capital S is turned on. This feature is important for four separate functions. 1a3

-Making user-invisible comments to other Help writers.
This would do away with the current percent sign convention which requires a special sequence generator or content analyzer pattern since it is not a part of the standard Nls capabilities. 1a3a

-Making output processor directives invisible in Help.
This would do away with the necessity of having two separate directories of files, one with Output Processor (OP) directives and one without. And it would do away with

having to delete directives and update to the second directory whenever a modification is made. 1a3b

-Placing links invisibly next to referenced text.
This is needed to place link syntax in a node to define how that node will be viewed. (See #2, below.) It would also automatically take the user to enriching references. 1a3c

-The comment feature would be one way of implementing the backlink property, where the links to a node are invisibly backlinked from that node. 1a3d

Simple Boolean expression searches:
Once automatic index maintenance is implemented it will provide the added advantage of being the basis for providing more sophisticated and efficient searching capabilities. This will speed searching. It will also allow standard Boolean arguments such as "x AND y OR z AND NOT a" and eventually, natural language typed in (and spoken?) English queries. 1a4

2. Augmenting the Help Command

Several features have emerged as essential in the evolution of the Help command. More effective user control of the information available and a closer integration with other capabilities are two areas that need attention now. 1b

Do feature:
The "do" feature is the capability of having Help execute a command for you or a task consisting of a scenario of commands. Should user specification be necessary in the process, the do feature will tell in English sentences what is expected at every step of the way. This is an active tutorial/example/service which whould make Help much more valuable as a teaching aid and provide a new service as a task doer. It would be implemented by writing commands branches which appear as menu items under Help descriptions. A special symbol placed after the right anglebracket of a link will process the commands branch. Uparrow or ^, indicates that Help is to process the commands in the branch addressed by the link. Backarrow or _, indicates that Help is to process the command(s) between the link delimiters. 1b1

Point with mouse to words and lines:
Help would be far more flexible if the Display user could point with the mouse. With this feature the user would be able to simply point to words and lines of text, and Help would display the descriptions of the words pointed to, or the descriptions pointed to by links contained in the lines. 1b2

Outline and verbose description views:
This feature would allow users to specify whether they would

like to see a brief, outline view of the description and its menu, or see the full description. 1b3

Descriptions for two levels of users:
A difficulty with the present Help data bases is that the descriptions must be tailored to a single, mythical, "average user". The result is some descriptions that are too advanced for the very new user and others that are too simple for the more knowledgeable user. This task would be to rewrite portions of the data base to provide two descriptions for major concepts and commands--one for novices and the other for experts. Some minor additions to Useroptions software could be made so that the user could specify whether she wanted to see the simple or more advanced descriptions. The default would be for the novice user. 1b4

References to additional online information:
A complete Help system points users to information of various sorts besides how to use tools. For example, hardcopy documentation, source files, mail indices, directories of people, and program lists could all be made available via Help. 1b5

3. Production of Hardcopy from Online Documentation

Although online documentation is flexible and immeditely accessible for user reference, hardcopy is preferred by some users, and is helpful for some applications. This endeavor would entail a variety of software and writing tasks to make online documentation such as the Help data bases automatically translatable into readable hardcopy. 1c

Features already cited in this document:
Many of the features cited above would aid in making online documentation into hardcopy, including the "Comment feature", "Backlinks" and the "Index generation" features listed under #1; and "Outline and verbose descriptions views" listed under #2. 1c1

Text that would make hardcopy more readable:
With the comment feature, documentors would be able to include textual passages, such as transitions, headings, and references, in the online document that would only be turned on for hard copy. This would help transform the essentially "reference" online documentation into a form which could be read from front to back. 1c2

4. New Documentation

Documentation of proposed software or software changes. 1d

5. Documentation Maintenance

Existing documentation must be updated to accurately describe changes in an experimental system. Some of the major documents that require much effort to maintain are: 1e

Help data bases:
- Base
- Core
- Sendmail
- Calculator
- Useroptions
- Message
- Programs
- Graphics
- Publications
- Letter
- Helpd 1e1

User manuals:
- Secretarial Functions Guide
- DEX User's Manual 1e2

Systems documentation:
- Tool Suppliers documentation
- FE System Interface documentation 1e3

BEV 22-AUG-75 18:59 26326

A Resend of Documentation Shopping List for NSW Proposal

(J26326) 22-AUG-75 18:59;;;; Title: Author(s): Beverly Boli/BEV; Distribution: /ARC-DEV([INFO-ONLY]) ; Sub-Collections: SRI-ARC ARC-DEV; Clerk: BEV; Origin: < BOLI, DOCPROP.NLS;4, >, 22-AUG-75 18:54 BEV ;;;;####;

26326 Distribution

Mary Ann Kellan, Andy Poggio, David L. Retz, Jan A. Cornish, Larry L. Garlick, Delorse M. Brooks, Beverly Boli, James E. (Jim) White, Ann Weinberg, Kenneth E. (Ken) Victor, Dirk H. Van Nouhuys, Jonathan B. Postel, Elizabeth K. Michael, David S. Maynard, Karolyn J. Martin, Harvey G. Lehtman, Kirk E. Kelley, Charles H. Irby, Joseph L. Ehardt, Robert Louis Belleville, Don I. Andrews, Richard W. Watson, Douglas C. Engelbart,

Graphics demo

!:15 friday is fine (1:15) 1

Graphics demo

(J26327) 25-AUG-75 19:27;;;; Title: Author(s): Robert Louis Belleville/RLB2; Distribution: /JHB([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: RLB2;

26327 Distribution
James H. Bair,

Output (to) Proof

Since it is clear that new generalized output commands are needed for the various output processors, sources, and destinations, I would like to add the "Output (to) Proof" command if it is not already included in the design. This command would be a part of the Base Output command (assuming "Output" remains the same and is not changed to something more intuitive like "Print") and format the currently loaded file and place it on the Tektronix. This could be accomplished by doing an "Output Com File" into the user's directory and then Goto Proof (loading it if necessary), and display the processed file on the Tecktronix thus saving the user from learning and having to do these several steps. 1

KIRK 26-AUG-75 00:54 26328

Output (to) Proof

(J26328) 26-AUG-75 00:54;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /EKM([ACTION]) FEEDBACK([ACTION]) NDM([ACTION]
) HGL([ACTION]) RLB2([ACTION]) SRI-ARC([INFO-ONLY]) ;
Sub-Collections: SRI-ARC FEEDBACK; Clerk: KIRK;

26328 Distribution

Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White, Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews,
Elizabeth K. Michael, Special Jhb Feedback, N. Dean Meyer, Harvey G. Lehtman, Robert Louis Belleville, Mary Ann Kellan, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Joseph L. Ehardt, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Rene C. Ochoa, Ann Weinberg, Joan Hamilton, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael

KIRK 26-AUG-75 23:01 26329

Output (to) Proof

Since it is clear that new generalized output commands are needed for the various output processors, sources, and destinations, I would like to add the "Output (to) Proof" command if it is not already included in the design. This command would be a part of the Base Output command (assuming "Output" remains the same and is not changed to something more intuitive like "Print") and format the currently loaded file and place it on the Tektronix. This could be accomplished by doing an "Output Com File" into the user's directory and then Goto Proof (loading it if necessary), and display the processed file on the Tecktronix thus saving the user from learning and having to do these several steps. 1

Output (to) Proof

(J26329) 26-AUG-75 23:01;;;; Title: Author(s): Kirk E. Kelley/KIRK; Distribution: /DMB([ACTION] dpcs notebook please) &DPCS([INFO-ONLY]) KIRK([INFO-ONLY] rejournalized to add to dpcs subcollection, hint hint hint) ; Sub-Collections: DPCS SRI-ARC; Clerk: DVN;

26329 Distribution
Delorse M. Brooks, Documentation Production and Control System Interest Group , Kirk E. Kelley,

DOCUMENTATION IDEAS FOR NEXT NSW PROPOSAL 1

1. Online Documentation Data Base Maintenance

The Help multi-file data base is an encyclopedic description of NLS tools and NSW procedures designed for quick online how-to reference by the user. To be effective it must be comprehensive and current. Methods to keep the Help data bases up-to-date and keep their extensive cross-referencing accurate are needed. 1a

Index generation and maintenance:
Current experience shows that the automatic creation and maintenance of an alpabetic index in a multifile data base is essential if the online Help is to grow and evolve as planned for NSW tools. This index would be generated from all named statements and perhaps all meaningful words. This needs the automatic link maintenance facility described below. This feature is also necessary for #3, below, "Hardcopy Production from Online Documentation." 1a1

Back links:
It is essential that automatic link maintenance via back links be implemented in order to reduce the tremendous overhead and inherent mistakes in the current procedure for discovering bad links and updating them manually. This should be implemented as a property of a node. See below. Back links have the added benefit of automatic "forward references", a list of back links to what has been written about a document since it was published. 1a2

Comment feature:
The comment feature would allow any arbitrary amount of designated text to dissappear when viewspec capital T is in effect. The text reappears when viewspec capital S is turned on. This feature is important for four separate functions. 1a3

-Making user-invisible comments to other Help writers.
This would do away with the current percent sign convention which requires a special sequence generator or content analyzer pattern since it is not a part of the standard Nls capabilities. 1a3a

-Making output processor directives invisible in Help.
This would do away with the necessity of having two separate directories of files, one with Output Processor (OP) directives and one without. And it would do away with

having to delete directives and update to the second directory whenever a modification is made. 1a3b

-Placing links invisibly next to referenced text.
This is needed to place link syntax in a node to define how that node will be viewed. (See #2, below.) It would also automatically take the user to enriching references. 1a3c

-The comment feature would be one way of implementing the backlink property, where the links to a node are invisibly backlinked from that node. 1a3d

Simple Boolean expression searches:
Once automatic index maintenance is implemented it will provide the added advantage of being the basis for providing more sophisticated and efficient searching capabilities. This will speed searching. It will also allow standard Boolean arguments such as "x AND y OR z AND NOT a" and eventually, natural language typed in (and spoken?) English queries. 1a4

2. Augmenting the Help Command

Several features have emerged as essential in the evolution of the Help command. More effective user control of the information available and a closer integration with other capabilities are two areas that need attention now. 1b

Do feature:
The "do" feature is the capability of having Help execute a command for you or a task consisting of a scenario of commands. Should user specification be necessary in the process, the do feature will tell in English sentences what is expected at every step of the way. This is an active tutorial/example/service which whould make Help much more valuable as a teaching aid and provide a new service as a task doer. It would be implemented by writing commands branches which appear as menu items under Help descriptions. A special symbol placed after the right anglebracket of a link will process the commands branch. Uparrow or ^, indicates that Help is to process the commands in the branch addressed by the link. Backarrow or _, indicates that Help is to process the command(s) between the link delimiters. 1b1

Point with mouse to words and lines:
Help would be far more flexible if the Display user could point with the mouse. With this feature the user would be able to simply point to words and lines of text, and Help would display the descriptions of the words pointed to, or the descriptions pointed to by links contained in the lines. 1b2

Outline and verbose description views:
This feature would allow users to specify whether they would

like to see a brief, outline view of the description and its menu, or see the full description. 1b3

Descriptions for two levels of users:
A difficulty with the present Help data bases is that the descriptions must be tailored to a single, mythical, "average user". The result is some descriptions that are too advanced for the very new user and others that are too simple for the more knowledgeable user. This task would be to rewrite portions of the data base to provide two descriptions for major concepts and commands--one for novices and the other for experts. Some minor additions to Useroptions software could be made so that the user could specify whether she wanted to see the simple or more advanced descriptions. The default would be for the novice user. 1b4

References to additional online information:
A complete Help system points users to information of various sorts besides how to use tools. For example, hardcopy documentation, source files, mail indices, directories of people, and program lists could all be made available via Help. 1b5

3. Production of Hardcopy from Online Documentation

Although online documentation is flexible and immeditely accessible for user reference, hardcopy is preferred by some users, and is helpful for some applications. This endeavor would entail a variety of software and writing tasks to make online documentation such as the Help data bases automatically translatable into readable hardcopy. 1c

Features already cited in this document:
Many of the features cited above would aid in making online documentation into hardcopy, including the "Comment feature", "Backlinks" and the "Index generation" features listed under #1; and "Outline and verbose descriptions views" listed under #2. 1c1

Text that would make hardcopy more readable:
With the comment feature, documentors would be able to include textual passages, such as transitions, headings, and references, in the online document that would only be turned on for hard copy. This would help transform the essentially "reference" online documentation into a form which could be read from front to back. 1c2

4. New Documentation

Documentation of proposed software or software changes. 1d

5. Documentation Maintenance

Existing documentation must be updated to accurately describe changes in an experimental system. Some of the major documents that require much effort to maintain are: 1e

Help data bases:
- Base
- Core
- Sendmail
- Calculator
- Useroptions
- Message
- Programs
- Graphics
- Publications
- Letter
- Helpd 1e1

User manuals:
- Secretarial Functions Guide
- DEX User's Manual 1e2

Systems documentation:
- Tool Suppliers documentation
- FE System Interface documentation 1e3

DOCUMENTATION IDEAS FOR NEXT NSW PROPOSAL

(J26330) 26-AUG-75 23:05;;;; Title: Author(s): Beverly Boli, Kirk E. Kelley/BEV KIRK; Distribution: /DMB([ACTION] dirt and dpcs notebooks please) DLS([INFO-ONLY]) EFF([INFO-ONLY]) PWO([INFO-ONLY]) BEV([INFO-ONLY] rejournalized for dirt and dpcs subcolletions, hint, hint, hint) &DPCS([INFO-ONLY]) &DIRT([INFO-ONLY]) ; Sub-Collections: DPCS DIRT SRI-ARC; Clerk: DVN;

26330 Distribution
Delorse M. Brooks, Duane L. Stone, Elizabeth F. Finney, Pat Whiting O'Keefe, Beverly Boli, Documentation Production and Control System Interest Group , Documentation Instigation and Review Team ,

Printing and the NLS-8 Command Summary

response to 33342

Printing and the NLS-8 Command Summary

The print commands are on page 10.

Printing and the NLS-8 Command Summary

(J26331) 26-AUG-75 23:19;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /SRI-ARC([INFO-ONLY]) DIRT([INFO-ONLY]) FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT FEEDBACK; Clerk: DVN;

26331 Distribution

Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, Jonathan B. Postel, Priscilla A. Wold, Rita Hysmith, Pamela K. Allen, Delorse M. Brooks, Elizabeth F. Finney, Beverly Boli, Lawrence A. Crain, Kirk Sattley, Susan Gail Roetter, Robert N. Lieberman, Ann Weinberg, Kenneth E. (Ken) Victor, Douglas C. Engelbart, James H. Bair, Elizabeth K. Michael, Richard W. Watson, Elizabeth J. Feinler, Harvey G. Lehtman, Kirk E. Kelley, Laura E. Gould, Jeanne M. Beck, Dirk H. Van Nouhuys, James C. Norton, Special Jhb Feedback Mary Ann Kellan, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Joseph L. Ehardt, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Rene C. Ochoa, Ann Weinberg, Joan Hamilton, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White

JMB 27-AUG-75 11:58 26332

Procedure for PR at Gunter to input columnar material into NLS

TABLES PROCEDURE 1

There is a new feature in NLS 8,5 at ISIC that makes it easier to use TABs to line up columns when typing in tables than it is in NLS-8 at Office-1. The Useroptions command, Space (for Tabs), sets up NLS so that the appropriate number of spaces are inserted in your line when you hit TAB to go to the next tabstop you have set. Its other effect is that the first character of your TYPEIN will always appear at the left margin, so that the position of the characters you type in corresponds to where the columns will appear when you print your file. Here is the procedure worked out with Jo Wagner and Cindy Pattillo for using this feature to enter columnar material using NLS at ISIC and getting that text merged into the rest of the document at Office-1. 2

SETTING UP: 3

PR inserts the normal text of their document into the appropriate NLS file at Office-1. When the typist reaches a table of columnar text of more than a couple of lines, she inputs only the title of the table, skips the columnar text, and resumes typing the statement following the table. She should note, on her hardcopy, the statement address (usually the SID) of the last statement which is to precede the table, 3a

When she is ready to input several of the skipped tables, she will logout of Office-1 and close the connection (by typing "@ c <CR>"). 3b

@ o <SP> 244 <CR> [Open a connection to USC-ISIC] 3c

@ LOG <SP> ROETTER <SP> SGR <SP> <CR> 3d

The @'s in this step and those immediately following are not typed by user, they are printed by TENEX as heralds. 3d1

@ TSET <CR> 3e

@ NLS <CR> 3f

INPUTTING YOUR FIRST TABLE: 4

BASE C: Goto Useroptions <CR> 4a

I've capitalized the letters of each command-word that user types. 4a1

USER C: <Sp>Space (for tabs) <CR> 4b

Procedure for PR at Gunter to input columnar material into NLS

USER C: <SP>Printoptions C: Tab (stop settings) T:
<SP> <SP> <SP> <SP> <SP> X <SP> <SP> <SP> X <SP> <SP> <SP> X <CR> 4c

[i.e, type a number of spaces, then a character where you want your first tab stop to be, space over to your next tab stop, type a character, etc. -- to set the tab stops you need for your first table. You have a 72-character line to work with.] 4c1

USER C: Show C: <SP>Printoptions <CR>
...
tabstops: 10,15,38,55 4d

This optional command will show you the numbers of the columns where you set the tabstops in the previous command; this is useful information because you can set the tabstops using the column numbers the next time you have a similar table, without re-spacing through that table. 4d1

USER C: Quit Nls <CR> 4e

@ NLS <CR> 4f

The Useroptions commands you have just given will take effect only in your next NLS session, so quit and recall NLS. 4f1

BASE C: <SP>CReate File T: FILENAME <CR> 4g

FILENAME Conventions:
The filename given to the table file should be the name of the corresponding chapter file at Office-1, followed by a "T" and a number. For example, the first table you input for AFM 85-652CH3 would carry the filename "AFM85-652CH3T1. 4g1

Wait for "<ROETTER, AFM,..,>" to appear to indicate that the file has been successfully created and loaded before going on: 4g2

BASE C: Insert Text (to follow) A: 0 +e <CR>
T:
This table is to follow statement # XXX. <CR> 4h

[where XXX = the SID or statement # of the statement in the Office-1 file which this table is to follow.] 4h1

BASE C: Insert Statement ... 4i

Start typing the table. Use <CTRL-I> to tab to the next column position. Use <CTRL-V><CR> to put in second lines of table entries. Use <CTRL-E> to end statement and input the next one. Using <CTRL-A> and <CTRL-W> to backspace your input will move

your carriage back and overprint and leave you in the right column position, <CTRL-V><CR> will mess up your spacing by printing <EOL> at the beginning of the next line, but your next TAB will put you in the right position; the <EOL> will not go into your file, 4i1

BASE C: Update File <CR> 4j

--when you have finished typing this table, Only one table is to go into each file, If you have been using <CTRL-E> to insert statements, remember to type <CTRL-X> before you Update, 4j1

INPUTTING ANOTHER TABLE-NEW FILE-SAME TABSTOPS: 5

BASE C: <SP>CReate File T: AFM85-652CH3T2 <CR> 5a

BASE C: Insert Text (to follow) A: 0 +e <CR>
T:
This table is to follow statement # XXX, <CR> 5b

[where XXX = the SID or statement # of the statement in the Office-1 file which this table is to follow,] 5b1

BASE C: Insert Statement ... 5c

Start typing the table, Use <CTRL-I> to tab to the next column position, Use <CTRL-V><CR> to put in second lines of table entries, Use <CTRL-E> to end statement and input the next one, Using <CTRL-A> and <CTRL-W> to backspace your input will move your carriage back and overprint and leave you in the right column position, <CTRL-V><CR> will mess up your spacing by printing <EOL> at the beginning of the next line, but your next TAB will put you in the right position; the <EOL> will not go into your file, 5c1

BASE C: Update File <CR> 5d

--when you have finished typing this table, Only one table is to go into each file, If you have been using <CTRL-E> to insert statements, remember to type <CTRL-X> before you Update, 5d1

INPUTTING ANOTHER TABLE/FILE WITH DIFFERENT TABSTOPS: 6

BASE C: Goto Useroptions <CR> 6a

USER C: <SP>Printoptions C: Tab (stop settings) T:
<SP> <SP> <SP> <SP> <SP> X <SP> <SP> <SP> X <SP> <SP> <SP> X <CR>
[or]

Procedure for PR at Gunter to input columnar material into NLS

25,35,42,65 <CR> [if you have already worked out the tabstops for this type of table and noted the column numbers] 6b

USER C: Quit Nls <CR> 6c

@ NLS <CR> 6d

BASE C: <SP>CReate File T: AFM85-652CH3T3 <CR> 6e

BASE C: Insert Text (to follow) A: 0 +e <CR>
T:
This table is to follow statement # XXX, <CR> 6f

[where XXX = the SID or statement # of the statement in the Office-1 file which this table is to follow.] 6f1

BASE C: Insert Statement ... 6g

Start typing the table. Use <CTRL-I> to tab to the next column position. Use <CTRL-V><CR> to put in second lines of table entries. Use <CTRL-E> to end statement and input the next one. Using <CTRL-A> and <CTRL-W> to backspace your input will move your carriage back and overprint and leave you in the right column position. <CTRL-V><CR> will mess up your spacing by printing <EOL> at the beginning of the next line, but your next TAB will put you in the right position; the <EOL> will not go into your file. 6g1

BASE C: Update File <CR> 6h

--when you have finished typing this table. Only one table is to go into each file. If you have been using <CTRL-E> to insert statements, remember to type <CTRL-X> before you Update. 6h1

WHEN SEVERAL TABLE/FILES HAVE BEEN CREATED & UPDATED: 7

BASE C: Goto Tenex <CR> 7a

@ SNDMSG <CR> 7b

To: weinberg@usc-isic, beck@bbnb,
roetter@bbnb,wold@bbnb,weinberg@bbnb 7b1

cc: dsdc-pr@office-1 7b2

Subject: Please transfer files to DSDC-PR at Office-1 7b3

Message: Please FTP the following files:
<AFMXXXT1,>

Procedure for PR at Gunter to input columnar material into NLS

<AFMXXXT2,>
to the appropriate chapter of AF manual at Office-1, RSVP to
DSDC-PR@Office-1, ...signed
<CTRL-Z> <CR> 7b4

@ LOGO <CR> 7c

TRANSFERRING THE FILES TO O-1 & MERGING THE TABLES INTO THE DOCUMENT: 8

These steps in the procedure will be carried out by SRI-ARC personnel until the procedures are more detailed and a Gunter person has been trained to take this over. 8a

When the message from a PR person is received, the specified files in Roetter's directory at USC-ISIC should be FTP'd to directory DSDC-PR at Office-1, to files of the same name. 8b

At Office-1, each transferred file should be loaded in turn and the origin statement printed to find the statement address for the location of the table in the document-proper. 8c

Load the chapter file and look at the statement whose SID is specified in the tables file to check that a table belongs there. 8d

Copy the Plex: <AFMXXXT1, 1> to follow <AFMXXX, SID> 8e

Recheck the chapter file to verify that the tables are included correctly (It will later be formatted so that tables appear at the left margin, so some lines may overlap now tho they will later accomodate 72 characters to the line). Update the chapter file. 8f

Delete the files AFMXXXT1, etc. from the directory DSDC-PR. 8g

Log in to USC-ISIC, and delete the files of the same name from Roetter's directory. 8h

Send a SNDMSG, from your own directory, to DSDC-PR, including in the Subject line the name of the PR person who did the tables. List the files that have been successfully transferred. cc: copy to all those in the original distribution. 8i

JMB 27-AUG-75 11:58 26332

Procedure for PR at Gunter to input columnar material into NLS

(J26332) 27-AUG-75 11:58;;;; Title: Author(s): Jeanne M. Beck/JMB; Distribution: /CFP([ACTION]) JVW([ACTION]) US([INFO-ONLY]) POOH([INFO-ONLY]) RWW([INFO-ONLY]) KIRK([INFO-ONLY]) LAC([INFO-ONLY]) MAS2([INFO-ONLY]) DVN([INFO-ONLY] Ann said she thought you might possibly be interested) JHB([INFO-ONLY]) ; Sub-Collections: SRI-ARC US; Clerk: JMB; Origin: < BECK, TABLES.NLS;6, >, 27-AUG-75 11:54 JMB ;;;;####;

26332 Distribution

Cynthia F. Pattillo, Josephine V. Wagner, Susan Gail Roetter, Priscilla A. Wold, Jeanne M. Beck, Pamela K. Allen, Rita Hysmith, Sandy L. Johnson, Ann Weinberg, Richard W. Watson, Kirk E. Kelley, Lawrence A. Crain, Marilynne A. Sims, Dirk H. Van Nouhuys, James H. Bair,

HGL 27-AUG-75 17:43 26333

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

The syntax of the Output command MUST be changed immediately before we tech Gunter about the COM stuff. Dean's design of long ago was good. We need applications approval before we can act, however. We also MUST document all the new directives mentioned in earlier messages. Hope this holds together in my absence!

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

Introduction 1

I have made the following changes to the output processor which permit switching between character tables and spacing algorithms for the Singer 6000 and the Comp80 on the basis of a parameter passed to the OP through a cell in the initialization table. Further work must be done to interface the syntax of the user command to this code: we currently must set a global in DDT for the switch to work; the user should be able to say something like Output COM Singer (or something similarly appropriate,) Code algorithms and character spacing tables for the Singer were taken from the <SRINLS> directory. Additionally, I fixed a bug in these algorithms which put out the wrong font size number for dot split directives for the Singer. The SRINLS version should no longer be necessary; the PORGEN version should be brought up as the running OP as soon as a new GNLS is brought up. Note that these changes were made both to PORGERN and XPORGEN (which has the diagram directives). Also, changes in the NLS code were made in both NLS and NIC-NLS, but they should be compatible with the old OP. 1a

Modifications to the OP: 2

LSGCOL: 2a

CHARSP is REFd at the beginning of the file 2a1

The former procedure charsp has been replaced by two procedures (porgen, lsgcol, c80charsp) and (porgen, lsgcol, singcharsp). Charsp is now a cell set up at initialization time which contains the address of one of these procedures depending on the variable COMDEV passed in the initialization table. CHARSP must be REFd at the beginning of each file in which it is called for the correct dispatch to occur. 2a2

Two tables contain the character sizes for Comp80 and Singer 6000: they begin at c80start and singstart repectively; they end at c80end annd c80start. The former tables (which contain overlapping subtables for various fonts) are set up with the Comp80 default. At initialization, OPEXEC BLTs the appropriate table into this space. (See below under OPEXEC.) 2a3

OPEXEC: 2b

Code added to initialize charsp with the address of the appropriate spacing procedure (or the default) and to BLT the appropriate character size tables on the basis of the value of COMDEV which is passed as the sixteenth cell of the initialization table. COMDEV is zero for Comp80, 1 for Singer. 2b1

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

```
(opexec) PROC,...                                                    2b1a
    .                                                                2b1a1
    .                                                                2b1a2
    .                                                                2b1a3
    LOCAL ...,  sptabstrt % char space table starting
    address%, spsize, lw, ...                                        2b1a4
    .                                                                2b1a5
    .                                                                2b1a6
    .                                                                2b1a7
    comdev _ [table + 15];                                           2b1a8
    .                                                                2b1a9
    .                                                                2b1a10
    .                                                                2b1a11
    % set default charsp routine; may be changed depending on
    comdev value, %                                                  2b1a12
       charsp _ sc80charsp;                                          2b1a12a
    .                                                                2b1a13
    .                                                                2b1a14
    .                                                                2b1a15
    CASE dev OF                                                      2b1a16
       .                                                             2b1a16a
       .                                                             2b1a16b
       .                                                             2b1a16c
       = comnm,...                                                   2b1a16d
          .                                                          2b1a16d1
```

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

```
                                                                    2b1a16d2
                .
                % set character spacing tables, charsp procedure,
                depending on value of comdev:  singer =0, comp80 =
                1; otherwise default to comp80 values, %            2b1a16d3
                   CASE comdev OF                                   2b1a16d3a
                      = singer:                                     2b1a16d3a1
                         BEGIN                                      2b1a16d3a1a
                         charsp _ $singcharsp;                      2b1a16d3a1b
                         sptabstrt _ $singstart;                    2b1a16d3a1c
                         spsize _ $singend - $singstart;            2b1a16d3a1d
                         END;                                       2b1a16d3a1e
                      ENDCASE % comp80%                             2b1a16d3a2
                         BEGIN                                      2b1a16d3a2a
                         charsp _ $c80charsp;                       2b1a16d3a2b
                         sptabstrt _ $c80start;                     2b1a16d3a2c
                         spsize _ $c80end - $c80start;              2b1a16d3a2d
                         END;                                       2b1a16d3a2e
                   lw _ spsize + $coursp - 1; %address of last word
                   to be block transferred to %                     2b1a16d3b
                   r1,LH _ sptabstrt;                               2b1a16d3c
                   r1,RH _ $coursp;                                 2b1a16d3d
                   !BLT r1, @lw;                                    2b1a16d3e
                .                                                   2b1a16d4
                .                                                   2b1a16d5
                .                                                   2b1a16d6
OPDATA:                                                             2c
```

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

New declarations: 2c1

comdev-- set from 16th element of initialization table passed from NLS, 2c1a

singer = 1, comp80 = 0-- legal values of comdev, 2c1b

charsp-- contains the address of appropriate character spacing procedure: set up in OPEXEC, 2c1c

PSTPRC: 2d

CHARSP is REFd at the beginning of the file 2d1

(porgen, pstprc, 0943)-- for dotsplit code; (porgen, pstprc, 0931) for other line segments. Both of these are in the procedure postcom, 2d2

```
CASE comdev OF                                          2d2a
  = singer:                                             2d2a1
    BEGIN                                               2d2a1a
    DIV (72*lsdfont,fsize)/1000,numpts,rem;             2d2a1b
    out1b (0);                                          2d2a1c
    out1b (numpts);                                     2d2a1d
    END;                                                2d2a1e
  ENDCASE %comp80%                                      2d2a2
    BEGIN                                               2d2a2a
    out2b (lsdfont,fsize);                              2d2a2b
    END;                                                2d2a2c
```

DLIBE, STFMT, NUMBER, DOCFMT: 2e

CHARSP is REFd at the beginning of the file 2e1

Modifications to NLS: (made in <NLS> and <NIC-NLS>) 3

(nls, seqfil, opinit) 3a

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

takes and additional parameter comdev which is placed into oprwrk[15]. 1 if singer com or 0 if comp80 or default. 3a1

(nls, cedit1, coutproc) 3b

call on opinit changed: 3b1

```
opinit (&da, jfn, device, opflags, gproc, IF comexflag THEN
1 %singer% ELSE 0 %comp80%); % Fix this when the command
syntax is fixed!!! Get rid of comexflag-- use for now as
flag to be set in DDT. %
```
3b1a

(nls, bdata,) 3c

COMEXFLAG declared and initialized to 0. To get singer tables, it must be set to be 1 in ddt. It is a kludge and should be deleted when we can get the information from the user through the OUTPUT command. 3c1

OPRWRK increased to 16 cells. 3c2

Outline of Changes in Output Processor to handle both Singer and Comp80 COM from the same system

(J26333) 27-AUG-75 17:43;;;; Title: Author(s): Harvey G. Lehtman/HGL; Distribution: /EKM([ACTION]) RLB2([ACTION]) KIRK([INFO-ONLY]) POOH([INFO-ONLY]) NDM([INFO-ONLY]) JCN([INFO-ONLY]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: HGL; Origin: < LEHTMAN, OPDOC.NLS;2, >, 27-AUG-75 17:38 HGL
;;;;####;

26333 Distribution
Elizabeth K. Michael, Robert Louis Belleville, Kirk E. Kelley, Ann Weinberg, N. Dean Meyer, James C. Norton, Richard W. Watson,

DVN 28-AUG-75 11:30 26334

Distribution and Cataloging of Journal Items Related to Documentation Production

Jeanne,
yes I am interested in the contents of (ijournal,26332,). So probably are a number of other people. For a long time we have kept a distribution group of people interested in document production, it is called DPCS. It is also a catalog subcollection that has been extracted a couple of times from the overall journal catalogs when information on that area was needed e.g. for the final report. For the last year or so some one (Currently Dee) has collected all items sent to DPCS into a notebook in her office for easy reference. For example Jake is very interested in manipulation of columns and she is on that distribution. This note is to encourage you and anyone having journal itmes related to publishing through NLS to add DPCS to the distribution for information.

1

DVN 28-AUG-75 11:30 26334

Distribution and Cataloging of Journal Items Related to Documentation Production

(J26334) 28-AUG-75 11:30;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /US([ACTION]) DMB([ACTION] dpcs notebook please) SRI-ARC([INFO-ONLY]) DPCS([INFO-ONLY]) ; Sub-Collections: SRI-ARC US DPCS; Clerk: DVN;

26334 Distribution

Douglas C. Engelbart, James C. Norton, Richard W. Watson, Charles H. Irby,
Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White, Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, Marilynne A. Sims, Delorse M. Brooks, Elizabeth F. Finney, Beverly Boli, Joseph L. Ehardt, James H. Bair, Robert N. Lieberman, Pat Whiting O'Keefe, James H. Bair, Robert Louis Belleville, Ann Weinberg, Thomas L. Humphrey, Jeanne M. Leavitt, Kirk E. Kelley, Duane L. Stone, Elizabeth J. Feinler, N. Dean Meyer, Dirk H. Van Nouhuys
Susan Gail Roetter, Priscilla A. Wold, Jeanne M. Beck, Pamela K. Allen, Rita Hysmith, Sandy L. Johnson, Delorse M. Brooks, Mary Ann Kellan, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Joseph L. Ehardt, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Rene C. Ochoa, Ann Weinberg, Joan Hamilton, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney

DVN 28-AUG-75 11:38 26335

DDSI Asks for Each File Only Once on a Tape

It has been our custom to put each file intended for processing into COM at DDSI twice on the tape sent from ISI. Although this duplication was intended for secruity in case there was a problem on the tape, that has never happened and the two copies have confused production people at DDSI a couple of times. They have asked us to begin putting each file on only once. We will agree to do that unless one of you raises objections. 1

DVN 28-AUG-75 11:38 26335

DDSI Asks for Each File Only Once on a Tape

(J26335) 28-AUG-75 11:38;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /DMB([ACTION] dpcs notebook please) DPCS([ACTION]) MEH([ACTION] what do you think of this?) IMM([ACTION]) SLJ([INFO-ONLY]) ; Sub-Collections: SRI-ARC DPCS; Clerk: DVN;

26335 Distribution

Delorse M. Brooks, Marilynne A. Sims, Delorse M. Brooks, Elizabeth F. Finney, Beverly Boli, Joseph L. Ehardt, James H. Bair, Robert N. Lieberman, Pat Whiting O'Keefe, James H. Bair, Robert Louis Belleville, Ann Weinberg, Thomas L. Humphrey, Jeanne M. Leavitt, Kirk E. Kelley, Duane L. Stone, Elizabeth J. Feinler, N. Dean Meyer, Dirk H. Van Nouhuys, Douglas C. Engelbart, James C. Norton, Richard W. Watson, Charles H. Irby, Martin E. Hardy, Inez M. Mattiuz, Sandy L. Johnson,

Past and Future AMC Training

Sounds like last week was a very busy week. I'd be interested in your impressions of how the training went and any suggestions you might have on the follow-up for all these new users. It would be good for us to coordinate other future plans - the following is a list of what's happening that I'm aware of: 1

We're planning a trip to Ft. Monmouth, New Jersey, Sept. 10-11, to give Grobstein and Dames and possibly some of their secretaries some Basic NLS training. Priscilla Wold is planning to do that training. 1a

Rita Hysmith is prepared to work with the people in the Washington area (Murdock and Edgewood Arsenal) on MSG when you arrange the schedule. I'd like to know what thoughts you have on dates for these two additional places so we can coordinate with Rita's other training commitments. 1b

One problem that I understand was significant at the various Mid-West locations was getting access to a TIP, ANTS etc. You might be able to obtain better service for the users of the AMC slot by making arrangements for them to dial a TIP that is closer geographically. I've talked with Jake Feinler and she suggested that the most likely place to gain access would be the WPAFB-TIP in Dayton, Ohio. In order to find out the Autovon numbers, arrangements will have to be made between yourself (as AMC's architect) and a person at WPAFB. The ARPANET liaison there is Leonard Fall and his phone number is (513) 255-6247. He would probably be the first person to check with. 2

Past and Future AMC Training

(J26336) 28-AUG-75 12:24;;;; Title: Author(s): Susan Gail Roetter/SGR; Distribution: /ESV([ACTION]) US([INFO-ONLY]) JCN([INFO-ONLY]) BJP([INFO-ONLY]) ; Sub-Collections: SRI-ARC US; Clerk: SGR;

26336 Distribution
E. S. VonGehren, Susan Gail Roetter, Priscilla A. Wold, Jeanne M. Beck, Pamela K. Allen, Rita Hysmith, Sandy L. Johnson, James C. Norton, Buddie J. Pine,

DAP 28-AUG-75 13:58 26337

Fall KWAC Meeting

I assume from your message of 19-AUG-75 that October 13-17 are definitely the dates for the next KWAC meeting. Accordingly, I am reserving those dates on my calendar; I definitely plan to attend. 1

DAP 28-AUG-75 13:58 26337

Fall KWAC Meeting

(J26337) 28-AUG-75 13:58;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /RMS2([ACTION]) ; Sub-Collections: NIC; Clerk: DAP;

26337 Distribution
Robert M. Sheppard,

DAP 28-AUG-75 14:07 26338

Automatic Letter Typing

Does anybody out there have a clean, efficient way to use NLS as an automatic typewriter -- i.e., for producing form letters? What we'd like to be able to do is to (for example) have a full mailing list in one file, complete both with the full mailing address and with the appropriate salutation kfor each intended recipient; put a letter in another file; then have the system generate said letter for each recipient, putting in the inside address and salutation for each. It ought to be fairly simple, but it apparently isn't, or at least not for me...But other systems, like IBM's SCRIPT package, provide for this sort of thiong, and I keep telling everyone how much better NLS is than those greasy kid stuff imitations, so...Can anyone help?

Regards,

David 1

Automatic Letter Typing

(J26338) 28-AUG-75 14:07;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /AID([ACTION]) JHB([INFO-ONLY]) FEEDBACK([
INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: NIC AID FEEDBACK;
Clerk: DAP;

26338 Distribution

Frank G. Brignoli, Inez M. Mattiuz, Connie K. McLindon, Michael A. Placko, David A. Potter, Terry H. Proch, Rudy L. Ruggles, Robert M. Sheppard, Duane L. Stone, Stanley M. (Stan) Taylor, Ronald P. Uhlig, James H. Bair, Special Jhb Feedback, James C. Norton,

DAP 28-AUG-75 16:10 26339

KWAC Fall Meeting Agenda Suggestions

Here are five suggestions for items to be included on the agenda for our fall meeting. I don't promise not to send more; these are some of my current and enduring concerns.

KWAC Fall Meeting Agenda Suggestions

DEX 1

After many months of no experience with DEX, I remain convinced that it's the way to go if one is to make NLS a maximally cost-effective tool. But I still don't know much about how it works out for users who are going in over the network. I have the impression from a paper Connie sent out a few months ago (I can't find it, but the title was "Flaky DEX" or something like that) that there are still many problems to be worked out. I'd like very much to find out more, as I suspect that DEX is something that would make NLS much more attractive to the non-users around here who foot the bill. 1a

Pricing Changes 2

What does the future hold? Will NLS cost more? Less? Will it shift from a flat rate for a slot to something hopefully more responsive to shifting usage patterns? What would be the impact of other potential developments (see 3 below)? 2a

PDP-11 as Interface 3

Jim Norton has mentioned the psopossibility/probability of (sometime around January) being able to use a PDP-11 of some sort to run the frontend of (I think) NLS-9. As I understand it this would hold the possibility of greatly increasing the number of users who could peacefully coexist on one slot (I may be mistaken on this), which for an organization with its own 11 would significantly inccrease cost-effectiveness. I'd like to learn a great deal more about this if it's not science fiction. 3a

Stupid Calculator 4

This has been a pet peeve of mine for many months. I get the feeling that other user groups aren't very quantitative, or maybe just can do all these petty numeric operations without the aid of artificial inteligence. So if nobody else is interested, maybe this doesn't belong on the agenda...If you ARE interested, please send Frank a note seconding my suggestion, which is basically that CALCULATOR ought to be able to outperform my little Bowmar pocket model. 4a

It can't. For example, most statistical techniques, even a simple descriptive one like the standard deviation, require that you perform different operations on the same set of numbers. To continue with the standard deviation example, one must square each number in the set, find the total of the numbers and find the total of the squares of the numbers. Now my little Bowmar (and 'most every other halfway decent little brain on the market) can

find the sum of the numbers and the sum of the squares simultaneously, but CALCULATOR can't, or at least I haven't found a decent way to do it. 4b

That's a simple example. Of course, some pocket calculators can do, much more -- e.g., a Texas Instruments machine selling for somewhere aroune $100 finds the standard deviation for you at the push of a button (of course, you still do have to enter the numbers). Let's not even think about what the $750 or so H-P calculators can do.. 4c

CALCULATOR is by contrast a garden-variety mental defective. It's a four-function, 10-memory calculator and nothing more; it's handy for doing my expense account, and maybe for balancing the checkbook, and not much else. It seems to me that with the resources of a PDP-10 at its disposal, CALCULATOR ought to be user-programmable to some extent at least. It seems, though, to be a pretty low-priority item at SRI; if other user groups need something more, however, perhaps SRI's priorities might be changed. 4d

Graphics 5

Last February I couldn't even SPELL graffics. I'm still not sure what it is. But there are many occasions when it sure would be nice to be able to get some graphs or tables or figures in the middle of a document without having to stand on my research assistant's head. So I thought it would be nice to find out what's happening in this area... 5a

KWAC Fall Meeting Agenda Suggestions

(J26339) 28-AUG-75 16:10;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /FGB([ACTION]) KWAC([INFO-ONLY]) ;
Sub-Collections: NIC KWAC; Clerk: DAP; Origin: < POTTER,
AGENDA.NLS;1, >, 28-AUG-75 16:07 DAP ;;;;####;

26339 Distribution

Frank G. Brignoli, Marilynne A. Sims, Elizabeth F. Finney, Lawrence A. Crain, E. S. VonGehren, Glenn A. Sherwood, Kathey L. Mabrey, Jeanne M. Beck, David A. Potter, Robert N. Lieberman, Terry H. Proch, Ronald P. Uhlig, Susan Gail Roetter, Michael A. Placko, Stanley M. (Stan) Taylor, Elizabeth J. Feinler, Rudy L. Ruggles, Frank G. Brignoli, Robert M. Sheppard, Richard W. Watson, Douglas C. Engelbart, James C. Norton, James H. Bair, Duane L. Stone, Inez M. Mattiuz, Connie K. McLindon,

JAKE 28-AUG-75 18:37 26340

Altmode in TNLS

It would really be useful if ALTMODE were also available for the completion of directory and filenames in TNLS as well as DNLS. Due to the number of files I maintain (and I would believe other users doing documentation might have a similar amount of files) many of them have long or similar names. It is time consuming checking the names in order to load them when in most cases ALTMODE would do the trick or indicate that the choice was not the right one. Is this a particular problem in TNLS, or just one of those things we haven't had a chance to implement as yet? 1

Altmode in TNLS

(J26340) 28-AUG-75 18:37;;;; Title: Author(s): Elizabeth J. Feinler/JAKE; Distribution: /FEEDBACK([ACTION]) SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: JAKE;

26340 Distribution

James E. (Jim) White, Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews,

Special Jhb Feedback, Mary Ann Kellan, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Joseph L. Ehardt, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Rene C. Ochoa, Ann Weinberg, Joan Hamilton, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer

KEV 28-AUG-75 17:29 26341

1st rough draft of the Programmers' Guide to Writing Language Modules for the Debugger

i would appreciate any comments (and especially comments relavent to design flaws or problems).

1st rough draft of the Programmers' Guide to Writing Language Modules for the Debugger

INTRODUCTION 1

A language module (LM) is that module in the debugger that is responsible for any langugae specific function, e.g. interpretting symbolic input according to the semantical and syntactical rules of the current high level language, or displaying a cell in the current high level language. Each language module in the debugger is designed to: 1a

be run under a specific operating system, and to 1a1

provide support for one and only one language, running in a specific environment, e.g. there is a separate BCPL language module for support of BCPL on a TENEX and for support of BCPL on an ELF. 1a2

A LM is loaded dynamically by the debugger dispatcher (DD) in response to certain commands by a user. Each LM is responsible for providing a number of routines (with well defined interfaces) and has available to it a number of routines and data structures in the DD and operating system (OS) modules (also with well defined interfaces). 1b

This document is a programmers guide to writing and implementing language modules (with specific detail to writing LMs to run under a TENEX environment). 1c

GROSS STRUCTURE OF A LANGUAGE MODULE 2

A language module consists basically of a dispatch table, routines and data structures that will be called and referenced by other modules of the debugger (hereafter refered to as external routines and data structures), and any routines and data structures (hereafter refered to as support routines and data structures) needed for the support of the external routines and data structures. 2a

At the heart of any LM is its dispatch table. The dispatch table contains: 2b

addresses of external routines, and 2b1

addresses of external data structures, and 2b2

in some instances, a dispatch table entry is itself an external data structure. (A dispatch table entry that is itself a data structure will be called a simple data structure.) 2b3

When the debugger dispatcher receives a language specific request from the debugger frontend (in response to some user action), the DD looks in the LM dispatch table for the address of the LM routine that supports the requested function. The DD will then call the LM routine and expect the LM routine to perform its function and optionally to return some results. 2c

(If a function is not supported by a language module, then the appropriate entry in the dispatch table shall be 0.) 2c1

To perform its function, a language module routine may find it necessary to call routines provided by the DD and/or the OS modules, or to refernce data structures in these other modules. To do so, the LM routine will use the dispatch table for these other modules and can thus call or reference routines and/or data structures that it does not provide itself. 2d

GENERATING A LANGUAGE MODULE 3

The following discussion is specific for generating a language module designed to run under TENEX. However, the principles involved are the same regardless of what operating system the language module will be run under. 3a

Language modules designed to run under TENEX live in the address space of the debugger in pages 240(octal) - 377(octal). The language module dispatch table MUST be the first thing in each language module. 3b

A language module is a TENEX SSAVE file that will be "GET"ted at the appropriate time. 3c

To generate a language module, the debugger loader must be used. The debugger loader contains: 3d

debugger-wide definitions, 3d1

the L10 runtime environment (for the debugger dispatcher and any other modules written in L10), and 3d2

the debugger frontend to backend communication package. 3d3

The following are the current TENEX and debugger loader commands to generate a language module (comments are bracketed by percent signs; atsign is the TENEX prompt character indicating willingness to accept a TENEX command; asterick is the debugger loader prompt character indicating willingness to accept a debugger loader

command; upper case refers to primitives that are discussed below): 3e

```
@get <nsw-debugger>beldr.sav        % get the debugger loader %        3e1
@reenter                            % start it properly %              3e2
*/240000c          % start loading the language module at page 240 %   3e3
*FILE1                                                                 3e4
...                                                                    3e5
*FILEn                                                                 3e6
*<ALTMODE>          % done loading command to the loader %             3e7
@OPTIONAL                                                              3e8
@SSAVE (pages from) 240 (to) 377 (on)
<nsw-debugger>FILENAME.sav                                             3e9
```

FILE1 ... FILEn are the rel files that comprise the language module, NOTE THAT THE DISPATCH TABLE MUST BE THE FIRST CELLS LOADED. 3f

OPTIONAL is an oppurtunity for the language module to perform some pre-initialization (such as saving the symbol table pointer in the 2nd entry in the dispatch table). 3g

FILENAME consists of the language name followed by a slash (/) followed the environment that this module is designed to run under, followed by a dash (-), followed by the name of the target process' environment. Thus a BCPL module designed to run under TENEX to debug code running under ELF would have the FILENAME: BCPL/TENEX-ELF 3h

THE DISPATCH TABLE 4

The symbolic offset names for the entries in the language module dispatch table are contained in the file <nsw-debugger>landsp.nls. Also, the debugger loader contains these definitions. (Note that an offset of 0 refers to the first entry in the dispatch table, i.e, the word loaded at 240000(octal) on a TENEX.) 4a

decimal	symbolic offset

offset	name	meaning	
			4b
0	lnini	address of initialization procedure	4b1
1	lnsymp	symbol table pointer for this module	4b2
2	lnbpte	address of procedure to call when a breakpoint is hit	4b3
3	lnbptl	address of procedure to call prior to resuming from a breakpoint	4b4
4	----	RESERVED FOR FUTURE USE	4b5
5	lnsadr	address of coroutine for displaying and assigning to address ranges	4b6
6	lnfmem	address of coroutine for displaying and assigning to content searches	4b7
7	----	RESERVED FOR FUTURE USE	4b8
8	lnmass	address of procedure for setting search mask	4b9
9	----	RESERVED FOR FUTURE USE	4b10
10	lnmems	address of procedure to execute "memory set" command	4b11
11-50	----	RESERVED FOR FUTURE USE	4b12

GENERAL DISCUSSION OF THE FUNCTIONS OF EXTERNAL ROUTINES 5

CALLING SEQUENCES & DATA STRUCTURES 5a

The language of the debugger is L10. This means that all inter-module communication must conform to L10 standards. This applies: 5a1

to all procedure calls, returns, argument passing, and result returning; 5a1a

to all coroutine OPENPORTs, PCALLs, argument passing, and result returning; 5a1b

and to all external data structures (those data structures maintained by one module and accessible to other modules through the appropriate dispatch table). 5a1c

This does not mean, however, that a LM cannot be written in a language other than L10. A LM could be written in any language as long as all inter-module communication and all external data structures and referencs conform to L10 standards. 5a2

COROUTINES 5b

Most external coroutines conform to the following standards: 5b1

When they are OPENPORTed they are passed some (or no) arguments that remain valid for this instance of the coroutine; 5b1a

The PORT ENTRY code for a coroutine does some intialization code (e.g. opening of other ports) that is valid for this instance of the coroutine and then does its EXIT PCALL; 5b1b

No arguments are returned to the owning routine in the EXIT PCALL; 5b1c

The results specified in the EXIT PCALL (or a terminating PCALL) phrase become the arguments for the first (or nTH) cycle of the coroutine; 5b1d

(A cycle is considered to start after the EXIT PCALL or after the PCALL that terminates a cycle; a cycle is considered to terminate when the coroutine (or some routine on the coroutine's behalf) does a PCALL to the coroutine's owner with the first argument returned being 0.) 5b1d1

Usually, one of the arguments for a coroutine is the address of an output string to be filled in with one line of information for the user; 5b1e

The coroutine writes this output string with one line of information and then PCALLs its owner. 5b1f

The value of the first argument in this returning PCALL is interpretted in the following manner: 5b1f1

If the value is greater than 0, then the output string should have a carriage-return linefeed sequence appended to it and then it should be presented to the

user; the coroutine has not completed a cycle yet and expects to be PCALLed again (with no new arguments) to continue its operation. This type of return will be called a positive return. 5b1f1a

If the value is less than 0, then the output string should have a space, followed by the assignment operator character, followed by several spaces, appended to it and then it should be presented to the user; the coroutine has not completed a cycle yet and expects to be PCALLed again to continue its operation; however, in this case, the coroutine usually expects to get 2 new arguments returned as results of this PCALL. This type of return will be called a negative return. 5b1f1b

These 2 new arguments usually consist of the address of a (potentially NULL) new value string (or FALSE if the user did not specify a new value), and the address of a current input mode record to be used to interpret the new value string. 5b1f1b1

If the value is equal to 0, then the coroutine has completed a cycle; if the output string has a non-zero length, then it (the output string) is considered to contain an error message to be presented to the user. At this time, the coroutine is ready to accept new arguments to start a new cycle. This type of return will be called a terminating return or a 0 return. 5b1f1c

DEBUGGER WIDE DATA STRUCTURES 5c

Many external routines in the language module must maintain certain data structures in the DD module. They do this either by calling routines in the dispatcher (through the DD's dispatch table) or by manipulating the data structures directly (once again, however, the location of the data structure is obtained through the DD's dispatch table). It is the responsibility of language module external routines to see that the following data structures are kept current: 5c1

LSTVDIS - this is a simple data structure which consists of one cell in the DD dispatch table which contains the most recently displayed value 5c1a

(the user represents this value by entering ESCAPECHAR-Q) 5c1a1

LSTEADR - this is a simple data structure which consists of

one cell in the DD dispatch table which contains the value of the most recently evaluated address range element 5c1b

(the user represents this value by entering ESCAPECHAR-L) 5c1b1

LSTADIS - this is a data structure containing the addresses of the last n displayed cells 5c1c

(n is currently set to 4) 5c1c1

(language module routines maintain this data structure by using the DD external routine whose address is at offset ddarng in the DD's dispatch table) 5c1c2

(the user represents this value by entering ESCAPECHAR-A) 5c1c3

(For a more detailed discussion of these data structures and routines see the appendix and the Programmers' Guide to the Debugger Dispatcher.) 5c2

INPUT / OUTPUT MODE RECORDS 5d

Many LM external routines take as arguments the address of the current input or output mode records. These records lie at the heart of the debugger and are used to govern the way input from a user is interpretted and the way output is formatted. Both of these are L10 records, and what follows is the L10 declarations for these records and an explanation of the possible values and meaning of the individual fields. 5d1

(All the following symbolic definitions are a part of the debugger loader and are thus available to all LMs.) 5d1a

THE INPUT MODE RECORD 5d2

(inmode) RECORD 5d2a

ihlang[5], 5d2a1

iclang[2], 5d2a2

iradix[5], 5d2a3

itmode[5], 5d2a4

ibytesize[6], 5d2a5

irname[ADDRESS]; 5d2a6

field	possible values	meaning	
ihlang	this field specifies what the current high level language for input is; i.e. if the current level of language in use (as specified by iclang) is highlanguage, then user input should be considered to conform to the semantical and syntacical rules of this high level language		5d2b
ihlang	l10	the current high level language in use is L10	5d2c
	cobol	the current high level language in use is COBOL	5d2d
	fortran	the current high level language in use is FORTRAN	5d2e
	bcpl	the current high level language in use is BCPL	5d2f
	pl1	the current high level language in use is PL1	5d2g
iclang	this field specifies what level of language should be used for the interpretation of user input		5d2h
iclang	machine	the current level of language in use is machine language	5d2i
	assembly	the current level of language in use is assembly language	5d2j

	highlanguage	the current level of language in use is the current high level language	5d21
iradix	a number	all numeric input should be interpretted as being numbers in the base specified by this field	5d2m
itmode	this field specifies what the current input mode is		5d2n
itmode interpretted	tmcurlang	user input should be according to the current language specifications of fields iclang and ihlang	5d2o
interpretted	tmascii	user input should be as ascii values	5d2p
interpretted	tmsixbit	user input should be as sixbit values	5d2q
interpretted	tmrad50	user input should be as radix 50 values	5d2r
interpretted	tmfloat	user input should be as floating point numbers	5d2s
interpretted each byte specified by	tmbyte	user input should be as successive bytes, with having a bytesize as ibytesize	5d2t
ibytesize current input	a number	bytesize to use if the mode is tmbyte	5d2u

```
   irname          ---          currently unused                    5d2v

THE OUTPUT MODE RECORD                                              5d3

   (outmode) RECORD                                                 5d3a

      ohlang[5],                                                    5d3a1

      oclang[2],                                                    5d3a2

      oradix[5],                                                    5d3a3

      otmode[5],                                                    5d3a4

      obytesize[6],                                                 5d3a5

      osymadr[1],                                                   5d3a6

      orname[ADDRESS];                                              5d3a7
```

field	possible values	meaning	
ohlang	this field specifies what the current high level language for output is; i,e, if the current level of language in use (as specified by oclang) is highlanguage, then output should be formatted to conform to the syntacical and semantical rules of this high level language		5d3b, 5d3c
ohlang	110	the current high level language in use is L10	5d3d
	cobol	the current high level language in use is COBOL	5d3e
	fortran	the current high level language in use is FORTRAN	5d3f

language in	bcpl	the current high level use is BCPL	5d3g
language in	pl1	the current high level use is PL1	5d3h
oclang should	this field specifies what level of language be used for the formatting of output		5d3i
oclang in	machine	the current level of language use is machine language	5d3j
in	assembly	the current level of language use is assembly language	5d3k
in language	highlanguage	the current level of language use is the current high level	5d3l
oradix base	a number	all numeric output should be formatted as numbers in the specified by this field	5d3m
otmode	this field specifies what the current output mode is		5d3n
otmode formatted language oclang	tmcurlang	user output should be according to the current specifications of fields and ohlang	5d3o
formatted	tmascii	user output should be as ascii values	5d3p
formatted	tmsixbit	user output should be as sixbit values	5d3q

	tmrad50	user output should be formatted as radix 50 values	5d3r
	tmfloat	user output should be formatted as floating point numbers	5d3s
	tmbyte	user output should be formatted as successive bytes, with each byte having a bytesize as specified by obytesize	5d3t
	tmnumeric	user output should be formatted numerically as numbers in the base specified by oradix	5d3u
	tmstring	user output should be formatted strings conforming to string data types of the current high level language	5d3v
	tmrecord	user output should be formatted as instances of the record, named by the string pointed to by orname, conforming to record data types of the current high level language	5d3w
	tmlist	user output should be formatted as lists conforming to list data types of the current high level language	5d3x

	tmarray	user output should be formatted as arrays conforming to array data types of the current high level language	5d3y
	tmequal	this output mode means to tell the user the numeric value of input address lists	5d3z
	tmquestion	this output mode means to tell the user in which block an input symbol (as part of an address list) is defined	5d3a@
obytesize	a number	bytesize to use if the current output mode is tmbyte	5d3aa
osymadr	BOOLEAN	if this field is TRUE, then display addresses as a symbol plus an offset (as discussed elsewhere); if this field is FALSE, then display addresses numerically	5d3ab
orname	address	the L10 string pointed to by this field is the name of the record descriptor to be used if the current output mode is tmrecord	5d3ac

CHARACTER SET 5e

Since the debugger is designed to support a number of different languages, and since most languages do not use the same character sets as valid characters in identifiers, etc,, the

interpretation of strings input by a user cannot be handled by the DD but must be handled by individual LMs. In order to maintain some consistancy for users while debugging many languages, the following approach has been adopted: 5e1

The DD contains an external data structure (Generic Function String - GFS) that is a 128 character L10 string whose address is in the DD dispatch table. The iTH character of this string contains a value that represents the generic function for ascii character code i. 5e1a

When a LM wishes to interpret user input strings, the LM must look up each character in the user input string in the DD GFS to determine the function of the user character and then act accordingly. 5e1b

If a LM wishes to modify the GFS it MUST use the DD external routine whose address is at offset ddchrs in the DD's dispatch table. 5e1c

For documentation and communication purposes, it is convienient to have a generic name to refer to the character that is currently serving a specific generic function. Thus, while the specific charcter may change, it can still be refered to by its generic name. The generic name for a character is the uppercase word of the generic function symbolic name, e.g. the generic name for the character that is currently serving the generic function of an address list delimetr (semicolonchar) is SEMICOLONCHAR. 5e2

The symbolic names for the generic function values are maintained in the file <nsw-debugger>ddtdef.nls and are defined in the debugger loader. Thus, these symbolic values are avaiable for all LM modules. The symbolic names and the meaning of these generic functions are as follows (the debugger default character, in the absence of user or LM modification, for a generic function will appear under the meaning column delimited by a left angle bracket (<) and a right angle bracket followed by a semicolon (>;): 5e3

generic function smbolic name	meaning of character i
	5e3a
normchar	this character is a normal character that can be interpretted in any manner the LM chooses 5e3b

pluschar	<+>; the user is using this character as the arithmetic addition operator	5e3c
minuschar	<->; the user is using this character as the arithmetic subtraction operator	5e3d
timeschar	<*>; the user is using this character as the arithmetic multiplication operator	5e3e
dividechar	</>; the user is using this character as the arithmetic division operator	5e3f
lparenchar	<(>; the user is using this character as the arithmetic left grouping character	5e3g
rparenchar	<)>; the user is using this character as the arithmetic right grouping character	5e3h
blockchar	<&>; the user is using this character as a block delimeter; e.g. the string: string1&string2 should be interpretted as symbol string2 in block string1 if & is the current BLOCKCHAR	5e3i
fieldchar	<.>; the user is using this character to delimit the fields of a record; if the current language does not support records, then this character may be interpretted as a normchar	5e3j
escapechar	<altmode or escape>; the user is using this character to mean interpret the next character as a debugger builtin variable; e.g., ESCAPECHAR followed by a 'Q (or 'q) refers to the builtin debugger variable which has the value of the last displayed cell	5e3k
spacechar	<space>; the user is using this character as a space character; the SPACECHAR should normally be interpretted as an entity delimeter, however, in the evaluating of address range elements, two strings separated only by SPACECHARs should be interpretted as having a PLUSCHAR between the 2 strings	5e3l
commachar	<,>; the user is using this character as an	

	address range delimeter to separate the two elements of an address range; under normal circumstances, a LM will never see the COMMACHAR in user input strings	5e3m
semicolonchar	<;>; the user is using this character to separate address ranges within address lists; under normal circumstances, a LM will never see the SEMICOLONCHAR in user input strings	5e3n
larrowchar	<_>; the user is using this character as the debugger assignment character; under normal circumstances, a LM will never see the LARROWCHAR in user input strings	5e3o
tabchar	<tab>; the user is using this character to mean display the cell addressed by the most recently displayed cell; under normal circumstances, a LM will never see the TABCHAR in user input strings	5e3p
poundchar	<#>; the user is using this character to mean back up to the previous displayed cell; under normal circumstances, a LM will never see the POUNDCHAR in user input strings	5e3q
lfchar	<linefeed>; the user is using this character to mean display the next sequential cell; under normal circumstances, a LM will never see the LFCHAR in user input strings	5e3r
uparrowchar	<^>; the user is using this character to mean display the previous sequential cell; under normal circumstances, a LM will never see the UPARROWCHAR in user input strings	5e3s
bslashchar	<\>; the user is using this character to mean display an address list in string mode; under normal circumstances, a LM will never see the BSLASHCHAR in user input strings	5e3t
equalchar	<=>; the user is using this character to mean display the value of the input address list; under normal circumstances, a LM will never see the EQUALCHAR in user input strings	5e3u
excmarkchar	<!>; the user is using this character to mean	

display cells as ascii values; under normal circumstances, a LM will never see the EXCMARKCHAR in user input strings 5e3v

lsquarechar <[>; the user is using this character to mean display an address list numerically; under normal circumstances, a LM will never see the LSQUARECHAR in user input strings 5e3w

qmarkchar <?>; the user is using this character to mean tell where symbols in an address list are defined; under normal circumstances, a LM will never see the QMARKCHAR in user input strings 5e3x

rsquarechar <]>; the user is using this character to mean display an address list as records; under normal circumstances, a LM will never see the RSQUARECHAR in user input strings 5e3y

slashchar </>; the user is using this character to mean display an address list symbollically; under normal circumstances, a LM will never see the SLASHCHAR in user input strings 5e3z

ADDRESS RANGES 5f

Due to the wide variety of syntacical and semantical rules of different languages for expression evaluation, it is not possible for the DD to evaluate address ranges. Thus many LM external routines take as input the addresses of the 2 ARE strings that compose an address range and it is the responsibility of the LM to evaluate the AREs. 5f1

However, each LM is expected to obey certain debugger standards in the evaluation of the AREs: 5f2

all characters must be checked to determine what generic function they are currently serving (see the above discussion on character sets), 5f2a

before evaluating an individual ARE, the LM must call the DD external routine whose address is at offset dddca in the DD's dispatch table to determine the gross type of the address range it has received, 5f2b

The following are the symbolic names (available to LMs

since the definitions are a part of the debugger loader)
and the meanings of the gross address range types: 5f2b1

dill - 5f2b1a

this is an illegal address range 5f2b1a1

dmem - 5f2b1b

this address range refers to cells in the address space of the target process 5f2b1b1

dfra - 5f2b1c

this address range refers to stack frames that reflect the target process' current language state for stack oriented languages 5f2b1c1

dfor - 5f2b1d

this address range refers to the formal parameters of a procedure in a procedural oriented language 5f2b1d1

dcat - 5f2b1e

this address range refers to the catchphrases for a procedure in a procedural oriented language that supports exceptional clauses (CATCHPHRASEs in L10; ON CONDITIONs in PL1) 5f2b1e1

dsig - 5f2b1f

this address range refers to the signal status of the target process 5f2b1f1

dadr - 5f2b1g

this address range refers to the memory utilization of the address space of the target process (e.g. a TENEX EXEC MEMSTAT command) 5f2b1g1

if a specific gross type is not supported by a specific LM, it must generate the appropriate error return rather than interpretting the ARE in some other manner. 5f2c

DETAILED DISCUSSION OF EACH ENTRY IN THE LANGUAGE MODULE DISPATCH TABLE 6

This section will discuss in detail each entry in a language module's dispatch table. Each entry will be discussed under its symbolic offset name. 6a

lnini 6b

entry type - procedure address 6b1

procedure function (brief) - 6b2

perform language and/or module initialization 6b2a

when called - 6b3

This procedure is called (once and once only) after the language module has been loaded by the debugger. (This is guaranteed to occur after the OS module has been loaded and initialized.) 6b3a

arguments - 6b4

1st argument: 6b4a

The address of the debugger dispatch table. 6b4a1

2nd argument: 6b4b

The address of the permanent output mode record. 6b4b1

3rd argument: 6b4c

The address of the permanent input mode record. 6b4c1

4th argument: 6b4d

The address of an output string to be used for potential error conditions or for presenting an initialization message to the user. 6b4d1

results - 6b5

1st result: 6b5a

If initialization is successful then this procedure should return TRUE as its first result; if initialization is not successful, then this procedure should return FALSE as its first result. In either case, this procedure may write the output string (whose address is

passed as the 4th argument) with a message to be presented to the user. 6b5a1

error conditions - 6b6

Any error conditions detected by this procedure should either be dealt with by this procedure or translated into a FALSE return with an appropriate error message written in the output string. 6b6a

external data structures maintained - 6b7

may wish to modify the DD GFS 6b7a

discussion - 6b8

The function of this procedure is to perform any language and/or module initialization required by the language module. 6b8a

This procedure should set up the permanent input and output mode records with the defaults for this language. Specifically, the fields OHLANG and OCLANG should be set up in the output mode record; and the fields IHLANG and ICLANG should be set up in the input mode record. In addition any other fields (such as the default radix to be used) may be setup. 6b8b

An example of language specific initialization might be: 6b8c

The default debugger character for the address range delimeter character (COMMACHAR) is a comma (',); however, a comma has syntactical and semantical meaning for L10 (and indeed for many languages), therefore a language module initialization procedure might wish to redefine the character that will be used as an address range delimeter at this time (in fact, the L10 module changes a comma to a colon (':) at this time and also changes some other character definitions). 6b8c1

(For a detailed discussion of how to change generic characters see the Programmers' Guide to the Debugger Dispatcher.) 6b8c1a

This procedure may also wish to copy entries from the DD's dispatch table into local variables to speed up future references. This is not necessary, but merely an efficiency consideration, as the address of the DD dispatch table, and

its entries (with the exception of those entries that are simple data data structures), is guaranteed not to change for the lifetime of an instance of an LM. 6b8d

(Note: the addresses of the permanent input and output mode records passed as arguments should not be "remembered" as the passed addresses are merely instances used for LM initialization purposes. Any external routines that reference these records (or the current (as opposed to permanent) input/output mode records) will be passed addresses for the pertinent records (which will probably be a separate instance.) 6b8d1

lnsymp 6c

entry type - symbol table pointer 6c1

discussion - 6c2

This entry is a symbol table pointer for the symbol table for the LM. (For most languages running on a TENEX this consists of the lefthalf of the word being a negative count of the number of words in the symbol table and the righthalf of the word being the address of the first word of the symbol table.) This entry is not used by the debugger, but is merely a convience to aid in the debugging of the LM itself. 6c2a

(Note that the LM symbol table must reside in the same part of the debugger address space allocated to the LM.) 6c2a1

lnbpte 6d

entry type - procedure address 6d1

procedure function (brief) - 6d2

Perform any language and/or module specific action required at breakpoint hit (and other, see below) time(s). 6d2a

when called - 6d3

This procedure will be called: 6d3a

when a target process is specified, and 6d3a1

when a breakpoint is encountered in the target process, and 6d3a2

when a tracepoint is encountered in the target process. 6d3a3

(In all instances, this procedure will not be called until after the OS module enter breakpoint procedure has been called.) 6d3b

arguments - 6d4

1st argument: 6d4a

a value (n) that indicates why the procedure is being called this time as follows: 6d4a1

n = 0: user has just specified a target process 6d4a1a

n > 0: n is the number of the breakpoint just encountered 6d4a1b

n < 0: -n is the number of the tracepoint just encountered 6d4a1c

results - 6d5

NONE 6d5a

error conditions - 6d6

NONE 6d6a

external data structures maintained - 6d7

NONE 6d7a

discussion - 6d8

The function of this procedure is to obtain the language state of the target process and to build any support data structures that may be required to reflect this state. 6d8a

For example, when this procedure is called after a target process has been specified, it will probably want to obtain the symbol table for the target process. 6d8b

Also for example, in the LM for a stack oriented procedural language such as L10 or ALGOL, this procedure might wish to build support data structures to represent the current state of the stack and the current frame (perhaps including the name of the routine that was interrupted). 6d8c

This procedure will be called after a breakpoint or tracepoint is encountered in the target process or when a new (or the first) target process is specified by the user. It will be called after the OS module enter breakpoint routine has been called and it therefore will have available to it the OS state of the target process (e.g. the registers of the target process). 6d8d

lnbpt1 6e

entry type - procedure address 6e1

procedure function (brief) - 6e2

address of to call prior to resuming from a breakpoint 6e2a

when called - 6e3

to be specified later 6e3a

arguments - 6e4

to be specified later 6e4a

results - 6e5

to be specified later 6e5a

error conditions - 6e6

to be specified later 6e6a

external data structures maintained - 6e7

NONE 6e7a

discussion - 6e8

to be specified later 6e8a

lnsadr 6f

entry type - coroutine address 6f1

coroutine function (brief) - 6f2

The function of this coroutine is to build strings (using the current output mode) for the display of, and optionally

to assign to (using the current input mode), the address ranges specified by the user. 6f2a

when called - 6f3

This coroutine will be called in response to user requests to display and/or assign to address lists. 6f3a

arguments - 6f4

at openport time - 6f4a

1st argument: 6f4a1

a boolean that is TRUE if this instance of this coroutine should perform assignments to, as well as displaying, address ranges; the boolean will be FALSE if this instance is to be used only for the display of address ranges. 6f4a1a

at cycle start time - 6f4b

1st argument: 6f4b1

the address of a starting ARE string 6f4b1a

2nd argument: 6f4b2

the address of the corresponding ending ARE string 6f4b2a

3rd argument: 6f4b3

the address of the output string that should be written by this coroutine 6f4b3a

4th argument: 6f4b4

the address of a 2 word block that contains in the first word the address of the current output mode record, and in the second word the address of the current input mode record 6f4b4a

at pulse after a positive return - 6f4c

NONE 6f4c1

at pulse after a negative return - 6f4d

1st argument: 6f4d1

FALSE, meaning no new value specified by the user; or the address of a string which must be evaluated according the current input mode record passed as the second argument (note that this may be a NULL string which for some cases is different that not specifying a new value string). The value of this string should then be assigned to the previously displayed entity. 6f4d1a

2nd argument: 6f4d2

The address of the current input mode record to be used to evaluate the 1st argument. 6f4d2a

results - 6f5

at openport time - 6f5a

NONE 6f5a1

all other times - 6f5b

returns > 0 means it has written the output string and that string should be presented to the user, it is not finished and expects to be called again with no arguments. 6f5b1

returns = 0 means it is done, i.e. it has completed the current cycle; the output string may be NULL or if non null, then it has detected an error and the output string is the error condition to be presented to the user. in either case it may be called again, but it must be presented fresh arguments. 6f5b2

(Note that if this error return occurs on a "first" pulse, it more than likely means that this coroutine got invalid or unsupported address range elements.) 6f5b2a

returns < 0 means it has written the output string and that string should be presented to the user and that the user should be asked for a new value to replace the old value just presented to him/her. In this case it expects to be called again with the new value string and new input mode record. 6f5b3

error conditions - 6f6

any error conditions detected by this coroutine should be translated into a 0 return with an appropriate error message written in the passed output string. The coroutine should then be ready to accept new arguments to start a new cycle. 6f6a

external data structures maintained - 6f7

may wish to update LSTADIS, LSTVDIS, and LSTEADR depending on the specific operation of this instance (discussed below) 6f7a

discussion - 6f8

This is the main routine for displaying, and modifying, the address space and state information of a target process. This routine is invoked by the DD in response to a number of commands by the user to display and/or assign to address lists. 6f8a

When this coroutine is OPENPORTed, it will be passed a boolean to indicate if this instance is to be used for displaying and assigning to address ranges, or only for the display of address ranges. 6f8b

A (complete) cycle consists of the display of, and optionally the assignment to, all the referenced cells in the passed address range. After each cycle, this coroutine must be ready to accept new arguments and to start a new cycle. This coroutine indicates cycle completion by giving a 0 return, with an optional error message written in the output string. 6f8c

This routine is designed to present a "line" of information at a time to the user. Thus this routine should format the passed output string with a line of information and then PCALL appropriately its caller. In the case of merely displaying address ranges, the formatted output string will have a carriage-return, linefeed sequence appened to it before it is displayed to the user. In the case of an assignment, the formatted output string will have a space followed by the LARROWCHAR (by default a left arrow ('_)) followed by several spaces appended to it before being presented to the user. 6f8d

The output line should be formatted according to the current output mode record. 6f8e

The following guidelines should be adhered to in normal cases: 6f8e1

The DD will display a header line that indicates the address range currently being displayed (and optionally being assigned to). 6f8e1a

For the display of data structures, the first line ouptut should be a header that indicates the type of the data structure and the address (discussed below) of the instance of this data structure (and optionally other data structure dependent information, e.g. the length of a record). 6f8e1b

Successive lines of output should contain either the entire data structure (e.g. an entire string) or successive elements of the data structure (e.g. successive fields of a record). 6f8e1b1

(It is recommended that these succesive lines have 3 spaces at their front.) 6f8e1b1a

For the display of certain gross address range types (e.g. stack frames), the first ouptut line should be a header indicating the type of gross address range and any other pertinent header type information. 6f8e1c

Successive lines of output should contain further information relavent to this address range. 6f8e1c1

(It is recommended that these succesive lines have 3 spaces at their front.) 6f8e1c1a

For the normal display of cells in the address space of the target process the output line should be formatted as follows: 6f8e1d

the address of the current cell being displayed, 6f8e1d1

addresses should be displayed either numerically in the current output mode radix, or as a symbol optionally followed by a plus sign and a numeric offset (in the current output mode radix), depending on the current output mode record field OSYMADR. Even if OSYMADR indicates that addresses should be displayed as a symbol plus an offset, if the offset is greater than MAXOFFSET (in the DD dispatch table), then the address should be displayed numerically. 6f8e1d1a

the address should be followed by a slash character ('/), followed by 3 spaces, 6f8e1d2

and finally the value of the cell(s) according to the current output mode record, 6f8e1d3

The following guidelines should be adhered to for exceptional cases: 6f8e2

If the current output mode specifies a data structure not supported for the current language, e.g. output as lists for COBOL, then this coroutine aught to give an 0 return with an appropriate message, 6f8e2a

If the current output mode specifies a data mode not yet implemented for the current language, e.g. output floating point numbers for L10, then this coroutine aught to give an 0 return with an appropriate message, 6f8e2b

If the current output mode specifies a data mode not yet implemented for the current language, e.g. output in source language for L10, then this coroutine may substitute a subset output mode for this case, e.g. output in assembly language, 6f8e2c

If the combination of address range gross type and current output mode conflict, e.g. an output mode of tmquestion (tell where symbols are defined) and a gross type of dfra (stack frame), then the gross type of address range should take precedence and this coroutine should format the output string in the manner appropriate to the gross type, 6f8e2d

If this instance of the coroutine is an assignment instance, and the address range specifies a type for which assignment is not meaningfull (e.g. signal status), then this instance should be treated as only a display instance for the current cycle, 6f8e2e

Note that for some address ranges it may be meaningful to assign to part of the address range and only to display other parts of the address range. Remember that this coroutine can control what will and will not be assigned to by the type of PCALL back to its owner that it performs (greater than 0 for display only; or less than 0 for display and assign). 6f8e2e1

This coroutine is responsible for maintaining the following DD data structures according to the following guidelines: 6f8f

LSTEADR - after this coroutine evaluates an ARE it should write the resulting value in lsteadr. 6f8f1

(Note: it is the discretion of the LM to decide whteher or not to do this for certain gross address ranges, e.g. the LM may wish to update LSTEADR for a normal memory range address range but not for a signal status address range.) 6f8f1a

LSTADIS - just prior to displaying a line of information, this coroutine should update the data structure LSTADIS (using the appropriate DD external routine) with the value of the address about to be displayed. 6f8f2

(Once again it is the discretion of the LM to decide whteher or not to do this for certain gross address ranges, e.g. the LM may wish to update LSTADIS for a normal memory range address range but not for a signal status address range.) 6f8f2a

LSTVDIS - just prior to displaying a line of information, this coroutine should update the data structure LSTVDIS with the value of the cell about to be displayed. 6f8f3

(Once again it is the discretion of the LM to decide whether or not to do this for certain gross address ranges or ceratin data structure types, e.g. the LM may not wish to update LSTVDIS for displaying of memory as lists.) 6f8f3a

lnfmem 6g

entry type - coroutine address 6g1

coroutine function (brief) - 6g2

The function of this coroutine is to build strings (using the current output mode) for the display of, and optionally to assign to (using the current input mode), cells within the specified address range that meet certain specified content requirements. 6g2a

when called - 6g3

This coroutine will be called in response to the user issuing the FIND command, 6g3a

arguments - 6g4

at openport time - 6g4a

1st argument: 6g4a1

a boolean that is TRUE if this instance of this coroutine should perform assignments to, as well as displaying, filtered address ranges; the boolean will be FALSE if this instance is to be used only for the display of address ranges, 6g4a1a

at cycle start time - 6g4b

1st argument: 6g4b1

the address of a 3 word parameter block that contains the following: 6g4b1a

1st word: a value indicating what kind of search to perform as follows: 6g4b1a1

cmlreferences: 6g4b1a1a

search for cells whose address portion is equal to the passed searchee value in the passed address range (when both have been apporpriately masked), 6g4b1a1a1

cmlcontent: 6g4b1a1b

search for cells in the passed address range whose content is equal to the passed searchee value (when both have been apporpriately masked), 6g4b1a1b1

cmlnot: 6g4b1a1c

search for cells in the passed address range whose content is not equal to the passed searchee value (when both have been apporpriately masked), 6g4b1a1c1

2nd word: FALSE indicating that no mask should be used in performing the search; or the address of a

string that is to be evaluated to become the mask to use. If this string is a NULL string, then use the default debugger mask (DEFMASK). 6g4b1a2

Even if this word is FALSE implying no mask, if this is a reference search an implicit mask may be used for some environments (e.g. under TENEX a reference search actually uses a mask that only causes the right half of words to be examined). 6g4b1a2a

3rd word: the address of the searchee string that must be evaulated to the searchee value according to the current input mode record. 6g4b1a3

2nd argument: 6g4b2

the address of a 2 word data block containing the following: 6g4b2a

1st word: the address of the first ARE string of an address range 6g4b2a1

2nd word: the address of the corresponding second ARE string of the above address range 6g4b2a2

3rd argument: 6g4b3

the address of a 2 word block that contains in the first word the address of the current output mode record, and in the second word the address of the current input mode record 6g4b3a

4th argument: 6g4b4

the address of the output string that should be written by this coroutine 6g4b4a

at pulse after a positive return - 6g4c

NONE 6g4c1

at pulse after a negative return - 6g4d

1st argument: 6g4d1

FALSE, meaning no new value specified by the user; or the address of a string which must be evaluated

according the current input mode record passed as the second argument (note that this may be a NULL string which for some cases is different from not specifying a new value string). The value of this string should then be assigned to the previously displayed entity. 6g4d1a

2nd argument: 6g4d2

The address of the current input mode record to be used to evaluate the 1st argument. 6g4d2a

results - 6g5

at openport time - 6g5a

NONE 6g5a1

all other times - 6g5b

returns > 0 means it has written the output string and that string should be presented to the user, it is not finished and expects to be called again with no arguments. 6g5b1

returns = 0 means it is done, i.e. it has completed the current cycle; the output string may be NULL or if non null, then it has detected an error and the output string is the error condition to be presented to the user, in either case it may be called again, but it must be presented fresh arguments. 6g5b2

returns < 0 means it has written the output string and that string should be presented to the user and that the user should be asked for a new value to replace the old value just presented to him/her. In this case it expects to be called again with the new value string and new input mode record. 6g5b3

error conditions - 6g6

any error conditions detected by this coroutine should be translated into a 0 return with an appropriate error message written in the passed output string. The coroutine should then be ready to accept new arguments to start a new cycle. 6g6a

external data structures maintained - 6g7

LSTADIS, LSTVDIS, and LSTEADR 6g7a

6g8

discussion -

This routine is used for displaying, and optionally assigning to, cells within an address range that meet certain content requirements. This routine is called by the DD in response to the FIND command issued by the user. 6g8a

When this coroutine is OPENPORTed, it will be passed a boolean to indicate if this instance is to be used for displaying and assigning to the cells in the passed address range that meet the specified content requirements, or only for the display of those cells. 6g8b

The operation of this routine is identical to the operation of lnsadr (see above) and it should obey all the same conventions (including the maintenance of the debugger wide data structures). The only difference is that before a cell is considered for display (and optional modification), it must pass the specified content requirements as follows: 6g8c

Both the potential cell to be displayed and the passed searchee value are masked (logically anded) with the appropriate mask. 6g8c1

If the user does not specify to use a mask (indicated by word 2 of the 1st argument being FALSE), then the mask to be used is one that will select an entire word for cmlcontent or cmlnot searches; or one that selects the address portion of a cell for cmlreferences searches. 6g8c1a

If the user does specify a mask that is a non null string, then the mask to be used is the evaluation of this string according to the current input mode. This mask is used regardless of the type of search. 6g8c1b

If the user specifies a null string as a mask, then the mask to be used is the external data structure DEFMASK in the DD, once again regardless of the type of search. 6g8c1c

The resulting 2 masked values are then compared and if equal and if this is a cmlcontent search or a cmlreferences search, then this cell is considered to pass the filter and should be displayed to the user (and optionally modified). 6g8c2

If the resulting 2 values are unequal, then this cell is considered to pass only if this is a cmlnot search. 6g8c3

lnmass 6h

entry type - procedure address 6h1

procedure function (brief) - 6h2

to set the DD external data structure DEFMASK, the debugger default mask for searches and memory setting 6h2a

when called - 6h3

This procedure is called by the DD in response to the user command to set the default mask. 6h3a

arguments - 6h4

1st argument: 6h4a

FALSE or the address of a (possibly NULL) string to be evaluated to become the debugger default mask 6h4a1

2nd argument: 6h4b

the address of the current input mode record to use in the evaluation of the 1st argument 6h4b1

3rd argument: 6h4c

the address of an output string for possible (error) messages 6h4c1

results - 6h5

1st result: TRUE indicating success; FALSE indicating error detected. 6h5a

in either case, if the output string is non NULL (which it should be for a FALSE return), it will be presented to the user. 6h5a1

error conditions - 6h6

any error conditions detected by this procedure should be translated into a 0 return with an appropriate error message written in the output string. 6h6a

external data structures maintained - 6h7

DEFMASK assuming all goes well (discussed below) 6h7a

discussion - 6h8

The function of this procedure is to set the debugger default mask (a DD external simple data structure whose offset in the DD's dispatch table is dddfmk) which is used by the FIND and MEMORY SET commands. 6h8a

The passed mask string should be evaluated according to the passed current input mode record and the resulting value written in DEFMASK. 6h8b

Any error conditions, such as no mask string, a null mask string, or an invalid mask string, should generate a 0 return with an appropriate error message written in the output string. 6h8c

This procedure should not modify any other external data structures. 6h8d

lnmems 6i

entry type - coroutine address 6i1

coroutine function (brief) - 6i2

to set all cells (masked appropriately) in the specified address range to the specified new value (masked appropriately). 6i2a

when called - 6i3

This coroutine is invoked by the DD in response to a "MEMORY SET" command issued by the user. 6i3a

arguments - 6i4

at OPENPORT time - 6i4a

1st argument: 6i4a1

FALSE or the address of a (possibly null) new value string. 6i4a1a

2nd argument: 6i4a2

the address of the current input mode record to use for the evaluation of the new value string and for the evaluation of any mask specified. 6i4a2a

3rd argument: 6i4a3

the address of the current output mode record to use for any output strings written 6i4a3a

4th argument: 6i4a4

TRUE to indicate the use of the mask as specified by the next argument; FALSE means don't use a mask. 6i4a4a

5th argument: 6i4a5

FALSE to mean use the debugger default mask as the mask to use; or the address of a (possibly null) string to be evaluated, according to the current input mode record, as the mask to use. 6i4a5a

6th argument: 6i4a6

the address of an output string for possible (error) messages. 6i4a6a

at cycle start time - 6i4b

1st argument: 6i4b1

the address of an ARE string 6i4b1a

2nd argument: 6i4b2

the address of the corresponding second ARE string 6i4b2a

3rd argument: 6i4b3

the address of an output string 6i4b3a

all other times - 6i4c

NONE 6i4c1

results - 6i5

at OPENPORT time - 6i5a

this coroutine should return a boolean in its EXIT PCALL phrase that is TRUE indicating that the passed arguments have been evaluated successfully, or FALSE if the passed arguments could not be evaluated successfully. In the case of a FALSE return the output string should contain an error message. 615a1

all other times - 615b

returns > 0 means it has written the output string and that string should be presented to the user. It is not finished and expects to be called again with no arguments. 615b1

returns = 0 means it is done, i.e. it has completed the current cycle; the output string may be NULL or if non null, then it has detected an error and the output string is the error condition to be presented to the user. In either case it may be called again, but it must be presented fresh arguments. 615b2

error conditions - 616

any error conditions detected by this routine should be translated into the appropriate 0 (or FALSE) return (at OPENPORT or other times) with an appropriate error message in the output string 616a

external data structures maintained - 617

LSTEADR 617a

discussion - 618

This routine is used to set all cells in a specified address range to a specific value. An optional mask may be employed to select only certain fields of the cells. 618a

If this routine completes a cycle successfully, it should generate a positive return indicating the number of words set (by writing the output string), and then give a cycle terminating PCALL back to its owner. 618b

If this routine encounters a problem during a cycle, it should generate a cycle terminating PCALL with an appropriate error message in the output string. The error message should include some indication of how many, and the addresses, of the cells that were set, as well as some

indication of why it could not complete its cycle coompletely, 6i8c

APPENDIX - SUMMARY OF THE DEBUGGER DISPATCHER'S DISPATCH TABLE 7

Each entry in the DD's dispatch table will be discussed under the name its symbolic offset. The symbolic offsets are available to LMs since they are a part of the debugger loader. 7a

ddiosi 7b

entry type - procedure address 7b1

procedure function (brief) - 7b2

This is the OS module's first time initialization routine. 7b2a

use by a LM - 7b3

This procedure should not be used by a LM. However, before the LM first time initialization routine is called, this routine will have been called. 7b3a

ddbpte 7c

entry type - procedure address 7c1

procedure function (brief) - 7c2

This is the OS module's breakpoint entry routine. 7c2a

use by a LM - 7c3

This procedure should not be used by a LM. However, before the LM breakpoint entry routine is called, this routine will have been called. 7c3a

ddbptl 7d

entry type - procedure address 7d1

procedure function (brief) - 7d2

This is the OS module's breakpoint leave routine. 7d2a

use by a LM - 7d3

This procedure should not be used by a LM. However, before

actually resuming from a breakpoint, or tracepoint, this procedure will be called after the LM's breakpoint leave routine is called. 7d3a

ddmsta 7e

entry type - coroutine address 7e1

coroutine function (brief) - 7e2

This is the OS module's coroutine that is used to show the utilization of the address space of the target process 7e2a

use by a LM - 7e3

This courtine is used by a if the address range passed to lnsadr is of gross type dadr (represented by the user's typing ESCAPECHAR-A). 7e3a

arguments - 7e4

at OPENPORT time - 7e4a

NONE 7e4a1

at cycle start time - 7e4b

1st argument: the address of an output string 7e4b1

(The LM lnsadr routine should pass the address of the output string it was passed to this routine.) 7e4b1a

2nd argument: the address of the current output mode record 7e4b2

(The LM lnsadr routine should pass the address of the current output mode record it was passed to this routine.) 7e4b2a

at all other times - 7e4c

NONE 7e4c1

results - 7e5

at OPENPORT time - 7e5a

NONE 7e5a1

at all other times - 7e5b

This coroutine will generate either: 7e5b1

a positive return indicating that it has written the output string and that string should be presented to the user. The LM upon receiving a positive return from this routine should generate a positive return to its owner, passing along the same output string, and then the LM should PCALL this routine again so that it may continue its cycle, 7e5b1a

a cycle terminating 0 return indicating that it is done. The output string may or may not have something written in it. The LM upon receiving this return from this routine should generate the equivalent return to its owner, passing along the output string. If it wishes, the LM may again PCALL this routine, however, it must pass it fresh arguments so that this routine may start a new cycle, 7e5b1b

ddrd1w 7f

entry type - procedure address 7f1

procedure function (brief) - 7f2

This procedure is used to read one word in the address space of the target process, 7f2a

use by a LM - 7f3

Whenever a LM routine wishes to examine (for whatever reason) a word in the address space of the target process, it must use this procedure, 7f3a

arguments - 7f4

1st argument: the address of the word the LM wishes to read 7f4a

results - 7f5

1st result: 7f5a

a value (n) indicating the success or failure of this routine as follows: 7f5a1

n = 0: word read successfully 7f5a1a

n < 0: invalid address passed to this routine 7f5a1b

n > 0: address passed to this routine represents a non-existant page in the target process 7f5a1c

2nd result: 7f5b

the contents of the addressed word, or 0 on error conditions 7f5b1

ddrdnw 7g

entry type - procedure address 7g1

procedure function (brief) - 7g2

This procedure is used to read one or more words in the address space of the target process. 7g2a

use by a LM - 7g3

Whenever a LM routine wishes to examine (for whatever reason) words in the address space of the target process, it must use this procedure. 7g3a

arguments - 7g4

1st argument: the address of the first word the LM wishes to read 7g4a

2nd argument: the number of words the LM wishes to read 7g4b

3rd argument: an address at which to store the read words 7g4c

results - 7g5

1st result: 7g5a

the number of words read and returned to the LM 7g5a1

2nd result: 7g5b

a value (n) indicating the success or failure of this routine as follows: 7g5b1

n = 0: words read successfully 7g5b1a

n < 0: invalid address passed to this routine 7g5b1b

n > 0: address passed to this routine represents a non-existant page in the target process 7g5b1c

ddwr1w 7h

entry type - procedure address 7h1

procedure function (brief) - 7h2

This procedure is used to write one word in the address space of the target process. 7h2a

use by a LM - 7h3

Whenever a LM routine wishes to write (for whatever reason) a word in the address space of the target process, it must use this procedure. 7h3a

arguments - 7h4

1st argument: the address of the word the LM wishes to write 7h4a

2nd argument: the value to be written in the addressed word 7h4b

results - 7h5

1st result: 7h5a

TRUE if the word was written successfully; FALSE otherwise. 7h5a1

ddwrnw 7i

entry type - procedure address 7i1

procedure function (brief) - 7i2

This procedure is used to write one or more words in the address space of the target process. 7i2a

use by a LM - 7i3

Whenever a LM routine wishes to write (for whatever reason) one or more words in the address space of the target process, it must use this procedure. 7i3a

arguments - 7i4

1st argument: the address of the word the LM wishes to write 7i4a

2nd argument: the number of words to write 7i4b

3rd argument: an address from which to get successive words to write in the target process' address space 7i4c

results - 7i5

1st result: 7i5a

TRUE if the words were written successfully; FALSE otherwise. 7i5a1

2nd result: 7i5b

the number of words actually written if the first result is FALSE 7i5b1

ddsrcm 7j

entry type - procedure address 7j1

procedure function (brief) - 7j2

This procedure will search the target process' address space between 2 passed addresses (inclusively) for cells that contain the passed value (after both have been masked appropriately). 7j2a

use by a LM - 7j3

this procedure is used by the LM lnfmem coroutine to perform content searches in the address space of the target process. 7j3a

arguments - 7j4

1st argument: a starting address 7j4a

2nd argument: an ending address 7j4b

3rd argument: value to search for 7j4c

4th argument: mask to apply to search value and words in target process 7j4d

5th argument: TRUE for a content search; FALSE for a not content search 7j4e

results - 7j5

1st result: TRUE if this procedure found a word that met the passed requirements; FALSE if not. 7j5a

2nd result: the address of the found word on success; FALSE otherwise. 7j5b

3rd result: the contents of the found word on success; indeterminate otherwise 7j5c

ddgpfs 7k

entry type - procedure address 7k1

procedure function (brief) - 7k2

This is the OS module's procedure for obtaining the OS state of a target process. 7k2a

use by a LM - 7k3

This procedure will not normally be used by a LM; rather the LM will use the external data structure that this routine sets up to reflect the OS state of the target process. 7k3a

ddspfs 71

entry type - procedure address 711

procedure function (brief) - 712

This is the OS module's procedure for modifying the OS state of a target process. 712a

use by a LM - 713

This procedure is used by the LM to modify the OS state of the target process. 713a

arguments - 714

To be specified later. 714a

results - 715

To be specified later. 715a

ddalos 7m

entry type - procedure address 7m1

procedure function (brief) - 7m2

To be specified later, % adr get storage proc % 7m2a

use by a LM - 7m3

To be specified later, 7m3a

arguments - 7m4

To be specified later, 7m4a

results - 7m5

To be specified later, 7m5a

ddrels 7n

entry type - procedure address 7n1

procedure function (brief) - 7n2

To be specified later, % adr free storage procedure % 7n2a

use by a LM - 7n3

To be specified later, 7n3a

arguments - 7n4

To be specified later, 7n4a

results - 7n5

To be specified later, 7n5a

ddcfrk 7o

entry type - address of a data structure 7o1

data structure name - CURFORK 7o2

data structure meaning - 7o3

this is a debugger internal name for the target process that is currently being debugged. 7o3a

data structure type - 7o4

this data structure is a single word 7o4a

use by a LM - 7o5

A LM probably wishes to maintain some internal data structures that correspond to which process is currently being debugged, e.g. the internal handle that corresponds to the process whose symbol table the LM is currently aware of. This data structure will always contain the internal debugger name of the current target process. 7o5a

ddsvec 7p

entry type - address of a data structure 7p1

data structure name - PFSTATE 7p2

data structure meaning - 7p3

this data structure contains the OS state of the current target process. 7p3a

data structure type - 7p4

this data structure is composed of 50 words (under TENEX, the first 16 of these words represent the registers of the target process; the meaning of the rest of the words will be specified later.) 7p4a

use by a LM - 7p5

Whenever a LM wishes to read the OS state of the current target process, it should reference this data structure 7p5a

ddland 7q

entry type - address of a data structure 7q1

data structure name - LANDSP 7q2

data structure meaning - 7q3

this is the language module's dispatch table 7q3a

data structure type - 7q4

this data structure is composed of 50 words (see above for description of the entries) 7q4a

use by a LM - 7q5

The LM must provide this data structure, but probably has no use for it directly. 7q5a

ddadro 7r

entry type - simple data structure 7r1

data structure name - MAXOFFSET 7r2

data structure meaning - 7r3

if addresses are being displayed as a symbol plus an offset, then if the offset is greater than the value of this cell, the address should be displayed numerically. 7r3a

data structure type - 7r4

this data structure consists of the single word in the DD dispatch table 7r4a

use by a LM - 7r5

The LM must use this cell whenever it is displaying addresses symbolically. 7r5a

ddlste 7s

entry type - simple data structure 7s1

data structure name - LSTEADR 7s2

data structure meaning - 7s3

the value of this data structure is the value of the last completely evaluated address range element. 7s3a

data structure type - 7s4

this data structure consists of the single word in the DD dispatch table 7s4a

use by a LM - 7s5

The LM should update this cell every time it evaluates an address range element for which it is meaningfull to update this cell (e.g. it is not meaningful to update this cell after the evaluation of an ARE that corresponds to the target process' signal status). 7s5a

dddcda 7t

entry type - procedure address 7t1

procedure function (brief) - 7t2

This procedure is used to determine the gross type of AREs.. 7t2a

use by a LM - 7t3

Before evaluating any ARE, the LM must call this procedure to determine the gross type of an address range. 7t3a

arguments - 7t4

1st argument: the address of the first ARE string 7t4a

2nd argument: the address of the corresponding second ARE string 7t4b

results - 7t5

1st result: 7t5a

a value indicating the gross type of the address range 7t5a1

ddchrt 7u

entry type - address of a data structure 7u1

data structure name - GFS 7u2

data structure meaning - 7u3

this is the DD Generic Function String 7u3a

data structure type - 7u4

this is a 128 character L10 string 7u4a

use by a LM - 7u5

Whenever the LM is reading user input strings, it must first look up in this data structure the generic function being served by the user characters before it can use the characters and it must use the characters according the function specified in this data structure. 7u5a

ddchrs 7v

entry type - procedure address 7v1

procedure function (brief) - 7v2

This procedure is used to modify the GFS. 7v2a

use by a LM - 7v3

If a LM wishes to change which character will be used for which generic function (e.g. at initialization time), the LM must use this procedure and NOT modify the GFS directly. 7v3a

arguments - 7v4

1st argument: the address of a string containing as its first character the ascii character that is to perform a generic function 7v4a

2nd argument: the generic function the character is to serve 7v4b

3rd argument: must be zero for LM use 7v4c

results - 7v5

this procedure can generate L10 HELP and ABORT signals if it receives bad input. (details to be specified later.) 7v5a

ddarng 7w

entry type - procedure address 7w1

procedure function (brief) - 7w2

This procedure is used for (reading or writing) the DD data structure LSTADIS (which contains the address of the last n displayed cells). 7w2a

use by a LM - 7w3

The LM is responsible for updating the data structure (by using this procedure) whenever it displays a cell in the target process' address space, 7w3a

arguments - 7w4

1st argument: 7w4a

FALSE to indicate read an entry from the LSTADIS data structure; TRUE to make a new entry in LSTADIS, 7w4a1

2nd argument: 7w4b

if this is a read operation, then this argument is the index of the last displayed address desired, e.g. the most recently displayed address has an index of 0, the address displayed before that has an index of 1, etc.; if this is a write operation, then this is the new address to add to LSTADIS 7w4b1

results - 7w5

for write operations - 7w5a

NONE 7w5a1

for read operations - 7w5b

the index-th (mod n, where n is the number of entries maintained, and is currently set to 4) last displayed address 7w5b1

ddldval 7x

entry type - simple data structure 7x1

data structure name - LSTVDIS 7x2

data structure meaning - 7x3

this is the value of the last displayed cell 7x3a

data structure type - 7x4

this data structure consists of the single word in the DD dispatch table 7x4a

use by a LM - 7x5

The LM is responsible for maintaining this cell 7x5a

dddfmk 7y

entry type - simple data structure 7y1

data structure name - DEFMASK 7y2

data structure meaning - 7y3

this is the this is the debugger default mask for content searches and memory setting commands of the last displayed cell 7y3a

data structure type - 7y4

this data structure consists of the single word in the DD dispatch table 7y4a

use by a LM - 7y5

The LM lnmass routine is responsible for setting this cell; and the LM routines lnfmem and lnmems may use this cell is responsible for maintaining this cell 7y5a

APPENDIX - SYMBOLS DEFINED IN THE DEBUGGER LOADER 8

The following are the symbols (and their values where appropriate) of the variables defined in the debugger loader and hence available to all LMs. All values are decimal unless a number is followed by an upper case B. Values are included here for the sake of completeness, but a LM should refer to these values symbolically. 8a

symbol	value	use	
------	-----	---	8a1

the following group of symbols are the symbolic offsets into the debugger dispatcher's dispatch table 8b

ddiosi	0	address of OS module 1st time init procedure	8b1
ddbpte	2	address of OS module breakpoint enter procedure	8b2
ddbptl	3	address of OS module breakpoint resume procedure	8b3

ddmsta	4	address of OS module memstat coroutine	8b4
ddrd1w	5	address of OS module read 1 word procedure	8b5
ddrdnw	7	address of OS module read n words procedure	8b6
ddwr1w	9	address of OS module write 1 word procedure	8b7
ddwrnw	11	address of OS module write n words procedure	8b8
ddsrcm	13	address of OS module search memory procedure	8b9
ddgpfs	15	address of OS module get state vector procedure	8b10
ddspfs	16	address of OS module set state vector procedure	8b11
ddalos	18	address of OS module get storage procedure	8b12
ddrels	19	address of OS module free storage procedure	8b13
ddcfrk	27	address of target process internal handle	8b14
ddsvec	28	address of target process state vector	8b15
ddland	29	address of language dispatch table	8b16
ddadro	30	maximum offset for symbolic address display	8b17
ddlste	31	LSTEADR - last evaluated ARE value	8b18
dddcda	32	address of decode address range procedure	8b19
ddchrt	33	address of character translation string	8b20
ddchrs	34	address of procedure to define character set	8b21
ddarng	35	address of procedure to manage LSTADIS	8b22
ddldval	36	LSTVDIS - last displayed value	8b23
dddfmk	37	DEFMASK - default search mask	8b24

the following group of symbols are the symbolic offsets into the language module's dispatch table 8c

lnini procedure	0	address of LM 1st time initialization	8c1
lnsymp	1	symbol table pointer for the LM	8c2
lnbpte	2	address of LM enter breakpoint procedure	8c3
lnbptl	3	address of LM breakpoint resume procedure	8c4
lnsadr	5	address of LM display address range coroutine	8c5
lnfmem	6	address of LM find content coroutine	8c6
lnmass	8	address of LM set mask procedure	8c7
lnmems	10	address of LM memory set coroutine	8c8

the following group of symbols are the symbolic offsets into the operating system module's dispatch table 8d

osini procedure	0	address of OS module 1st time initialization	8d1
ossymp	1	symbol table pointer this module	8d2
osbpte procedure	2	address of OS module breakpoint enter	8d3
osbptl procedure	3	address of OS module breakpoint resume	8d4
osmsta	4	address of OS module memstat routine	8d5
osrd1w	5	address of OS module read 1 word procedure	8d6
osrdnw	7	address of OS module read n words procedure	8d7
oswr1w	9	address of OS module write 1 word procedure	8d8
oswrnw	11	address of OS module write n words procedure	8d9
ossrcm	13	address of OS module search memory procedure	8d10
osgpfs procedure	15	address of OS module get state vector	8d11
osspfs procedure	16	address of OS module set state vector	8d12

osalos	18	address of OS module get storage procedure	8d13
osrels	19	address of OS module free storage procedure	8d14
oscfrk	27	CURFORK - curretn target process internal handle	8d15
ossvec	28	target process state vector	8d16
osdspl	78	length of OS module dispatch table	8d17
symbol table types			8e
l10symtab	0	L10-10	8e1
l11symtab	0	L10-11	8e2
m10symtab	1	macro-10	8e3
gross types of address ranges			8f
dill	0	illegal address range	8f1
dmem	1	normal memory display	8f2
dfra	2	stack frame	8f3
dfor	3	formal parameters	8f4
dcat	4	catchphrases	8f5
dsig	5	signal status	8f6
dadr	6	memstat	8f7
subtypes of normal memory adr ranges			8g
dmnor	1	normal cell display	8g1
dmrec	2	record field	8g2
dmfor	3	procedure formal	8g3
dmloc	4	procedure local	8g4
generic funtion values (see discussion else where for meaning)			8h
normchar	0		8h1

agravechar	25	8h2
atsignchar	26	8h3
blockchar	21	8h4
bslashchar	17	8h5
colonchar	27	8h6
commachar	10	8h7
dividechar	5	8h8
dollarchar	28	8h9
dquotechar	4	8h10
equalchar	15	8h11
excmarkchar	29	8h12
fieldchar	23	8h13
langlechar	14	8h14
larrowchar	20	8h15
lcurlychar	30	8h16
lparenchar	6	8h17
lsquarechar	16	8h18
minuschar	11	8h19
percentchar	31	8h20
pluschar	9	8h21
poundchar	32	8h22
qmarkchar	24	8h23
ranglechar	33	8h24
rcurlychar	34	8h25

```
        rparenchar      7                                       8h26
        rsquarechar     18                                      8h27
        semicolonchar   13                                      8h28
        slashchar       12                                      8h29
        spacechar       22                                      8h30
        tildechar       35                                      8h31
        timeschar       8                                       8h32
        uparrowchar     19                                      8h33
        vbarchar        36                                      8h34
        lfchar          2                                       8h35
        escapechar      3                                       8h36
        tabchar 1                                               8h37
    input output control  (see discussion elsewhere for meanings)  8i
      (outmode) RECORD                                          8i1
        ohlang[5],                                              8i1a
        oclang[2],                                              8i1b
        oradix[5],                                              8i1c
        otmode[5],                                              8i1d
        obytesize[6],                                           8i1e
        osymadr[1],                                             8i1f
        orname[ADDRESS];                                        8i1g
      (inmode) RECORD                                           8i2
        ihlang[5],                                              8i2a
        iclang[2],                                              8i2b
```

```
        iradix[5],                                                  8i2c
        itmode[5],                                                  8i2d
        ibytesize[6],                                               8i2e
        irname[ADDRESS];                                            8i2f
    values for fields ihlang and ohlang in input / output mode records  8j
        l10     3                                                   8j1
        cobol   4                                                   8j2
        fortran 5                                                   8j3
        pl1     6                                                   8j4
        bcpl    7                                                   8j5
    values for fields iclang and oclang in input / output mode records  8k
        machine         0                                           8k1
        assembly        1                                           8k2
        highlanguage    2                                           8k3
    values for fields itmode and otmode in input / output mode records  8l
        tmascii         0                                           8l1
        tmsixbit        1                                           8l2
        tmrad50         2                                           8l3
        tmbyte          3                                           8l4
        tmcurlang       4                                           8l5
        tmnumeric       5                                           8l6
        tmstring        6                                           8l7
        tmrecord        7                                           8l8
        tmlist          8                                           8l9
```

tmarray	9	8l10
tmfloat	10	8l11
tmequal	11	8l12
tmquestion	12	8l13

signal names 8m

niy	2000	signal value for not yet implemented functions 8m1

cml punctuation character command word values 8n

cmlagrave	99	8n1
cmlampersand	76	8n2
cmlasteric	80	8n3
cmlatsign	93	8n4
cmlbslash	95	8n5
cmlcolon	87	8n6
cmlcomma	83	8n7
cmldollar	74	8n8
cmldquote	72	8n9
cmlequal	90	8n10
cmlexcmark	71	8n11
cmllfangle	89	8n12
cmllarrow	98	8n13
cmllcurly	100	8n14
cmllparen	78	8n15
cmllsquare	94	8n16
cmlminus	84	8n17

cmladdresses	50	8p1
cmlarray	52	8p2
cmlascii	53	8p3
cmlbyte	54	8p4
cmlfloating	55	8p5
cmllanguage	56	8p6
cmllist	57	8p7
cmlnumeric	58	8p8
cmlr50	59	8p9
cmlradix	60	8p10
cmlrecord	61	8p11
cmlsixbit	62	8p12
cmlstring	63	8p13
cmlsymbolic	51	8p14
cmlabsolute	52	8p15
cmlmachine	50	8p16
cmlassembly	51	8p17
cmlhigh	52	8p18
cmlcobol	53	8p19
cmlfortran	54	8p20
cmll10	55	8p21
cmll11	56	8p22
cmlpl1	57	8p23

cml selection type command word values 8q

cmloctal	8	8v3
cmlhex	16	8v4

KEV 28-AUG-75 17:29 26341

1st rough draft of the Programmers' Guide to Writing Language Modules for the Debugger

(J26341) 28-AUG-75 17:29;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /JBP([ACTION]) RWW([ACTION]) CHI([ACTION]) DIA([ACTION]) DSM([ACTION]) EKM([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: KEV; Origin: < NSW-DEBUGGER, LANGUAGE-MODULES.NLS;2, >, 28-AUG-75 17:05 KEV ;;;;####;

26341 Distribution
Jonathan B. Postel, Richard W. Watson, Charles H. Irby, Don I. Andrews, David S. Maynard, Elizabeth K. Michael,

DVN 28-AUG-75 23:53 26342

Scrambled Markers Ride Again

I just updated my initial file and scrambled the markers. 1

DVN 28-AUG-75 23:53 26342

Scrambled Markers Ride Again

(J26342) 28-AUG-75 23:53;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /FEEDBACK([ACTION]) JDH([INFO-ONLY]) KJM([INFO-ONLY]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: DVN;

26342 Distribution
Special Jhb Feedback, J. D. Hopper, Karolyn J. Martin,

DVN 29-AUG-75 00:05 26343

DDSI and COM TAPES

follows 33362

DVN 29-AUG-75 00:05 26343

DDSI and COM TAPES

Yes, I knew about your problem, but there was another occasion when by chance we did not put a file on twice and hence they never processed the suceeding file. They are something of screwups. 1

DDSI and COM TAPES

(J26343) 29-AUG-75 00:05;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /DMB([ACTION] dpcs notebook please) IMM([INFO-ONLY]) &DPCS([INFO-ONLY]) ; Sub-Collections: SRI-ARC DPCS; Clerk: DVN;

26343 Distribution
Delorse M. Brooks, Inez M. Mattiuz, Documentation Production and Control System Interest Group ,

Automatic Letter Writing

follows 26338

DVN 29-AUG-75 00:10 26344

Automatic Letter Writing

Dave,Kirk is working on a automatic letter writting program for NSW, I don't know just where it stands. Jake has some programs that creat mailing lables from the ident list.

1

Automatic Letter Writing

(J26344) 29-AUG-75 00:10;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /&DPCS([INFO-ONLY]) DAP([INFO-ONLY]) KIRK([INFO-ONLY]) FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC DPCS FEEDBACK; Clerk: DVN;

26344 Distribution
Documentation Production and Control System Interest Group , David A. Potter, Kirk E. Kelley, Special Jhb Feedback,

JBP 29-AUG-75 01:04 26345

Altmode in TNLS

Altmode (known as Escape since 1968) works for me, check your user options and terminal type. --jon. 1

JBP 29-AUG-75 01:04 26345

Altmode in TNLS

(J26345) 29-AUG-75 01:04;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JAKE([ACTION]) FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: JBP;

26345 Distribution
Elizabeth J. Feinler, Special Jhb Feedback,

JBP 29-AUG-75 01:16 26346

high level language note

if you havent seen it before you might be interested in (26083,) on the proposed high order language, --jon, 1

JBP 29-AUG-75 01:16 26346

high level language note

(J26346) 29-AUG-75 01:16;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JGN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

26346 Distribution
J. Gregory Noel,

SGR 29-AUG-75 11:44 26347

Ident RAR2 - Response to (33360,)

Looks like rar2 should be able to use dsdc-sc - is this what you wanted? Also, when you have questions or problems such as this it would be best not to send it to sri-arc. That distribution has about 40 people in it and many of them wonder why they are being asked questions such as the above. A good distribution is myself and feedback. I'll see that rar2 can use dsdc-sc - let me know if that's not right. 1

Ident RAR2 - Response to (33360,)

(J26347) 29-AUG-75 11:44;;;; Title: Author(s): Susan Gail Roetter/SGR; Distribution: /JLC([ACTION]) US([INFO-ONLY]) FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC US FEEDBACK; Clerk: SGR;

26347 Distribution
Johnny L. Crabtree, Susan Gail Roetter, Priscilla A. Wold, Jeanne M. Beck, Pamela K. Allen, Rita Hysmith, Sandy L. Johnson, Special Jhb Feedback,