Review of Energy Information System Work
21-DEC-73 0705-PST

21-DEC-73 0705-PST RUSSELL: REVIEW OF ENERGY INFORMATION SYSTEM WORK
cc: WILLIS, ROMNEY, BLACK, ENERGY at SRI-ARC, CERL, RUSSELL,
Received 21-DEC-73 07:05:22 1

PART I,
FOR BOB RODDEN, OTHERS FOR INFORMATION, 1a

PLEASE ARRANGE A REVIEW OF THE STATUS OF THE SRI ENERGY INFORMATION SYSTEM WORK FOR WEDNESDAY, 30 JAN 74 AT SRI, MENLO PARK, CALIF BEGINNING AT 0900. ATTENDING FROM ARPA WILL BE DR. WILLIS, MR. BLACK AND COLONEL RUSSELL. PLEASE PROVIDE A RECOMMENDED AGENDA BY 18 JAN 74. BOTH THE MORNING AND AFTERNOON OF THE 30TH ARE AVAILABLE, IF REQUIRED. PLEASE CONFIRM THAT THIS DATE IS ACCEPTABLE. 1a1

PART II FOR CERL, 1b

IF SCHEDULES WILL PERMIT, A CERL REPRESENTATIVE IS INVITED TO ATTEND THE 30 JAN REVIEW. PLEASE ADVISE. 1b1

PART III FOR JIM NORTON, 1c

PLEASE ARRANGE TO CONDUCT AN IN-DEPTH TRAINING SESSION ON NLS FOR DR. WILLIS ON TUESDAY 29 JAN 73 AT THE SRI-ARC NLS FACILITY. 1c1

DCR2 29-MAR-74 09:16 30328

Review of Energy Information System Work
21-DEC-73 0705-PST

Distribution: ehw cfr rab4 deis cerl

DCR2 29-MAR-74 09:16 30328

Review of Energy Information System Work
21-DEC-73 0705-PST

(J30328) 29-MAR-74 09:16; Title: Author(s): David C. Russell/DCR2;
Sub-Collections: DEIS; Clerk: NDM;

NLS Support of the DEIS Program
25-DEC-73 1051-PST

25-DEC-73 1051-PST RUSSELL: NLS SUPPORT OF THE DEIS PROGRAM
cc: WILLIS, ROMNEY, BLACK, RUSSELL
Received 25-DEC-73 10:51:03 1

FOR BOB RODDEN
PLEASE CALL ME WHEN YOU RETURN TO WORK AFTER CHRISTMAS TO DISCUSS THE ARC PROPOSAL TO APPLY A MAJOR ELEMENT OF NLS TO THE DEIS PLANNING PROBLEM. DOUG ENGELBART'S SUGGESTIONS ARE INTERESTING AND I WOULD LIKE TO PURSUE THEM WITH YOU IN THE CONTEXT OF THE ENERGY INFORMATION SYSTEM PROGRAM. 1a

DCR2 29-MAR-74 09:24 30329

NLS Support of the DEIS Program
25-DEC-73 1051-PST

Distribution: ehw cfr rab4 deis

RMR2 29-MAR-74 09:33 30330

Reference 26Decc73 telecom between Russell and Rodden
26-DEC-73 1353-PST

26-DEC-73 1353-PST ENERGY at SRI-ARC: Reference 26Dec73 telecom between Russell and Rodden
cc: russell at ISI, cerl at ISI, engelbart
Received 26-DEC-73 13:53:44 1

Your suggested meeting at ARPA to discuss the Energy Information System Project and possible NLS support on 3Jan74 is acceptable to SRI. SRI will be represented by Rodden, Engelbart, Capps, Brown and possible others. 1a

Project review meeting for Energy Information Project has been scheduled for 30Jan74 at SRI-Menlo beginning at 0900. Recommended agenda for this meeting will be submitted for your approval prior to 18Jan74. 1b

RMR2 29-MAR-74 09:33 30330

Reference 26Decc73 telecom between Russell and Rodden
26-DEC-73 1353-PST

Distribution: dcr2 cerl dce deis

RAS 29-MAR-74 09:41 30331

Draft Report, etc.
2-JAN-74 1224-PST

Distribution: deis

Draft Report, etc.
2-JAN-74 1224-PST

2-JAN-74 1224-PST ENERGY: DRAFT REPORT,ETC
cc: ENERGY at SRI-ARC
Received 2-JAN-74 12:24:33 1

FIVE COPIES OF THE DRAFT REPORT BEING CARRIED BACK TO YOU BY
RODDEN AND BROWN. FIVE COPIES ALSO BEING SENT TO CERL THIS DATE.
SECOND SUBJECT., HAVE YOU RETREIVED THE ARC SUPPORT ESTIMATE
DUCUMENT NUMBER 21228 ? ROODDEN BRINGING EXTRA COPIES WITH HIM.
PLEASE KEEP IN TOUCH. 1a

RAS 29-MAR-74 09:41 30331

Draft Report, etc.
2-JAN-74 1224-PST

(J30331) 29-MAR-74 09:41; Title: Author(s): Richard A. Schmidt/RAS;
Sub-Collections: DEIS; Clerk: NDM;

DRB OWW 29-MAR-74 11:11 30332

Information for Col. D. Russell on Computer Support for Mr. Christensen
10-JAN-74 1757-PDT

Distribution: agc ras dce rww jcn deis dcr2

DRB OWW 29-MAR-74 11:11 30332

Information for Col. D. Russell on Computer Support for Mr. Christensen
10-JAN-74 1757-PDT

10-JAN-74 1757-PDT MEYER at SRI-ARC: Sally Miller: Please give hard copy to Arlie Capps immediately. We'll take care of other copies.
cc: norton, energy at NIC
Received 10-JAN-74 17:59:49 1

SRI can assist Mr. Christensen in his use of computer resources, particularly in use of information services and resources available through the ARPA Network. 1a

A description of possible terminal facilities to be located in or nearby Mr. Christensen's office is being prepared by Doug Engelbart. This memo attempts to set the terminal hardware in a larger context. 1b

The ultimate development of computer support for the DEIS cannot be described at this time. Our preliminary thoughts are presented here, together with some of the immediate and short-range plans that now seem to be fairly obvious. 1c

The ARPA Network terminal facility to be established in or nearby Mr. Christensen's office will be used for handling unclassified information. This will provide access to the ARPA energy community, its correspondence, memoranda, reports, models, problem solvers, data, etc. The usefulness of this connection should be considerably enhanced by the establishment of the two demonstration EPACs that SRI has recently proposed to ARPA. The EPACs can be used in connection with Mr. Christensen's terminal facility for the development of the design of the DEIS and at the same time for the experimental development of some of the analysis and problem-solving techniques that will be incorportated into the design. Thus, we see the possibility of experimental technique development in the EPACs being transferred to operational use in Mr. Christensen's terminal facility on an evolving basis. Also, Mr. Christensen may be able to use the computer resources available to him via this terminal to do some special things of his own, for example, in response to special requests for information. 1d

We expect that several different ARPA host computers will be supporting the DEIS development and operation. These possibilities have not been fully explored, but at this time we forsee the possibility of support from (1) SRI's ARC in the form of NLS, (2) the Data Computer for data-base development, (3) MIT MULTICS with JANUS. 1e

The terminal facility described above will not be adequate for classified information. For classified information the MULTICS

system now operating in the Pentagon near Mr. Christensen's office is suitable. This system includes JANUS, a data management system, that is currently operating with data-bases somewhat similar to the initial DEIS data-base. Other subsystems under MULTICS may also prove to be useful. The terminal facility for use with classified information would have to be separate from the terminal facility described previously, probably located in the same secure area as the present MULTICS system. 1f

In our development efforts for the DEIS, and in our support of Mr. Christensen, our objective would be to make use of whatever resources were available that would be both effective and economical. 1g

DRB CWW 29-MAR-74 11:11 30332

Information for Col. D. Russell on Computer Support for Mr. Christensen
10-JAN-74 1757-PDT

(J30332) 29-MAR-74 11:11; Title: Author(s): David R. Brown, Oliver W. Whitby/DRB CWW; Sub-Collections: DEIS; Clerk: NDM;

RMR2 29-MAR-74 11:19 30333

Contract Modification
11-JAN-74 1504-PDT

Distribution: dcr2 cerl deis

Contract Modification
11-JAN-74 1504-PDT

11-JAN-74 1504-PDT ENERGY at SRI-ARC: contract modification
cc: russell at ISI, energy, cerl at ISI
Received 11-JAN-74 15:05:43 1

Contract modification paperwork for energy Information Project received at Menlo on January 10. It has been reviewed, approved and is being forwarded to the District office in Chicago on 11Jan74. 1a

RMR2 29-MAR-74 11:19 30333

Contract Modification
11-JAN-74 1504-PDT

(J30333) 29-MAR-74 11:19; Title: Author(s): Robert M. Rodden/RMR2;
Sub-Collections: DEIS; Clerk: NDM;

AGC 29-MAR-74 11:26 30334
Tentative Agenda: Defense Energy Information System Project Review
15-JAN-74 1346-PDT

15-JAN-74 1346-PDT MEYER at SRI-ARC: To B. Rodden from A. Capps:
Tentative Agenda - 1/30/74
cc: energy, ndm/ndm at NIC
Received 10-OCT-10 10:15:10 1

TENTATIVE AGENDA 1a

DEFENSE ENERGY INFORMATION SYSTEM
PROJECT REVIEW
STANFORD RESEARCH INSTITUTE
MENLO PARK, CALIFORNIA 94025
30 JANUARY 1974 1b

0900	Welcome	D.R. Scheuch, VP, Research Operations	1c
	- National importance		1d
	- Multidivisional		1e
	- Experience in energy, information systems		1f
0905	Introduction	R.M. Rodden, Project Supervisor	1g
	- Project team		1h
	- Objectives of meeeting		1i
0910	The DoD Energy Structure	R.A. Black, ARPA Management	1j
0915	ARPA's Role in the DEIS	D.C. Russell, ARPA	1k
0920	Review of DEIS Work to Date	R.A. Schmidt, SRI	1l
	- Volume I		1m
	- Volume II		1n
0950	Project Research Plan	A.G. Capps, SRI	1o
	- Elements		1p
	- Network charts		1q
1020	Coffee Break		1r
1030	Energy Management Problems	A.G. Capps	1s

Tentative Agenda: Defense Energy Information System Project Review
15-JAN-74 1346-PDT

	- Types of Problems		1t
	- Questions		1u
	- etc.		1v
1100	Energy Hardship Management	R.A. Schmidt	1w
	- Security		1x
	- Demonstration of EPAC Links		1y
1200	Lunch		1z
1315	DEIS Design Options	D.R. Brown, SRI	1a@
	(to be detailed by Dave Brown)		1aa
1345	Discussion of Candidate Data System Technologies	D.C. Russell, ARPA (Chairman)	1ab
	ARPANET	Schelonka, USAF	1ac
	NLS	Engelbart/Norton/Watson/SRI	1ad
	Data Computer	Tom Marill, CCA	1ae
	LTLTHBR	Markowitz, MIT	1af
	BBN	Representative to be determined	1ag
1530	Coffee Break		1ah
1545	Continued Discussion		1ai
1645	Summary	D.C. Russell, ARPA	1aj
1700	Adjourn	R.M. Rodden, SRI	1ak

AGC 29-MAR-74 11:26 30334

Tentative Agenda: Defense Energy Information System Project Review
15-JAN-74 1346-PDT

(J30334) 29-MAR-74 11:26; Title: Author(s): Arlie G. Capps/AGC;
Sub-Collections: DEIS; Clerk: NDM;

AGC 29-MAR-74 11:26 30334

Tentative Agenda: Defense Energy Information System Project Review
15-JAN-74 1346-PDT

Distribution: rmr2 deis

Proposed Agenda
15-JAN-74 1604-PDT

Distribution: agc deis

Proposed Agenda
15-JAN-74 1604-PDT

15-JAN-74 1604-PDT ENERGY at SRI-ARC: Message to CAPPS from RODDEN
re proposed agenda
cc: energy
Received 15-JAN-74 16:04:54 1

Discussions regarding proposed agenda continuing. Questions and comments include: 1a

1. Schedule seems a little busy. May not be enough time for discussion. 1a1

2. The time allotted to ARC is very short. 1a2

3. More information on types of problems to be covered by Capps at 1030 would be appreciated. 1a3

4. No mention of AUTODIN or WWMCCS . Perhaps representatives can attend. Also Defense Communications Agency. Rubenson may have a suggestion regarding DCA representative. This representative should have knowledge of packet switching development plans. 1a4

5. Might want to reverse 0950 and 1030 sessions. 1a5

RMR2 29-MAR-74 11:30 30335

Proposed Agenda
15-JAN-74 1604-PDT

(J30335) 29-MAR-74 11:30; Title: Author(s): Robert M. Rodden/RMR2;
Sub-Collections: DEIS; Clerk: NDM;

DCE 29-MAR-74 11:34 30336

30 Jan Agenda
16-JAN-74 1229-PDT

Distribution: rmr2 jcn rww deis

30 Jan Agenda
16-JAN-74 1229-PDT

16-JAN-74 1229-PDT ENGELBART at SRI-ARC: To Rodden from Engelbart, re 30 Jan Agenda
cc: energy, norton, watson, engelbart
Received 16-JAN-74 13:31:35 1

From ARC's viewpoint, proposed agenda seems generally good; Specific assumptions about how details get arranged are as follows: 1a

Morning demonstration: Project tasks were set up so that Capps handles our NLS application to DEIS Project work, and that Jim Norton is managing ARC's support; assume then that Arlie will negotiate with Norton what he wants demonstrated, and how. 1b

Afternoon technology presentations and discussions: Assume that Dave Brown will structure the afternoon, responding to Capps' and Rodden's direction. Dick Watson will support him for ARC in planning our role which is consistent with the Info-sys analysis and design activity going on between Brown and Watson. Watson will be attending another ARPA meeting and can't be present Jan 30; Engelbart will therefore carry ARC's part of afternoon activity. 1c

DCE 29-MAR-74 11:34 30336

30 Jan Agenda
16-JAN-74 1229-PDT

(J30336) 29-MAR-74 11:34; Title: Author(s): Douglas C. Engelbart/DCE; Sub-Collections: DEIS; Clerk: NDM;

Pollock's (of CERL) Mailbox
16-JAN-74 1747-PDT

Distribution: deis

Pollock's (of CERL) Mailbox
16-JAN-74 1747-PDT

16-JAN-74 1747-PDT ENERGY: TO PEGGY:POLLACK'S "MAILBOX"
cc: ENERGY at SRI-ARC
Received 16-JAN-74 17:47:42 1

HI PEGGY, 1a

POLLACK HAS THE USER NAME CERL 1b

SO, JUST SEND HIS MAIL TO : CERL (IF YOU ARE ON ISI)
CERL<AT-SIGN> (IF YOU ARE ON SRI) 1c

ALSO, DIRK SHOULD BE ABLE TO SHOW YOU HOW TO FIND HIM (SPELLED
POLLOCK) 1d

VIA THE NET, IF DIRK IS NOT AROUN SEE MY EXAMPLE RUN FROM SRI
NUMBER VI. 1e

GOOD LUCK AND LET ME KNOW HOW IT TURNS OUT. 1f

PAMELA 1g

PS WHERE IS OUR COPY OF THE AGENDA SENT TO USSELL? I AM ON IS NOW
BUT WILL GO SRI AND LOOK FOR IT THERE. IF BOB DID NOT SEND IT,
PLEASE DO. THANKS. 1h

PGK 29-MAR-74 11:37 30337

Pollock's (of CERL) Mailbox
16-JAN-74 1747-PDT

(J30337) 29-MAR-74 11:37; Title: Author(s): Pamela G. Kruzic/PGK;
Sub-Collections: DEIS; Clerk: NDM;

RMR2 29-MAR-74 11:38 30338

Received Comments on Agenda
17-JAN-74 0822-PDT

Distribution: agc deis

Received Comments on Agenda
17-JAN-74 0822-PDT

17-JAN-74 0822-PDT ENERGY at SRI-ARC: for capps from rodden
Received 17-JAN-74 08:22:08 1

your message regarding our comments on the agenda received, I will attempt to review with Brown and Norton immediately. 1a

RMR2 29-MAR-74 11:38 30338

Received Comments on Agenda
17-JAN-74 0822-PDT

(J30338) 29-MAR-74 11:38; Title: Author(s): Robert M. Rodden/RMR2;
Sub-Collections: DEIS; Clerk: NDM;

NDM 29-MAR-74 11:41 30339

MESSAGE SYSTEM: Reading Schedule
17-JAN-74 1200-PDT

Distribution: deis jcn dvn

MESSAGE SYSTEM: Reading Schedule
17-JAN-74 1200-PDT

17-JAN-74 1200-PDT MEYER at SRI-ARC: MESSAGE SYSTEM
cc: energy, meyer, norton, vannouhuys
Received 17-JAN-74 12:00:46 1

One and all, be it known that: Menlo Park will read its messages at 800, 1100, 1315, 1500, and 1645 PDT, Washington D.C. will read its messages at 800, 1200, 1400, 1600, and 1730 EDT. This will be done every working day until further notice. Modifications will go through Dirk vanNouhuys and Dean Meyer. 1a

NDM 29-MAR-74 11:41 30339

MESSAGE SYSTEM: Reading Schedule
17-JAN-74 1200-PDT

(J30339) 29-MAR-74 11:41; Title: Author(s): N. Dean Meyer/NDM;
Sub-Collections: DEIS; Clerk: NDM;

EPAC Tape Recorder
17-JAN-74 1702-PDT

Distribution: agc ndm deis

EPAC Tape Recorder
17-JAN-74 1702-PDT

17-JAN-74 1702-PDT ENERGY: MSG FOR CAPPS FROM R, RODDEN
cc: MEYER
Received 17-JAN-74 17:02:46 1

PATTY CONE IS ORDERING A SONY TAPE RECORDER AND TRANSCRIBER FOR YOUR USE IN THE ENERGY PROJECT, WE WILL MAKE ARRANGEMENTS TO GET IT TO YOU WHEN IT COMES IN. 1a

RMR2 29-MAR-74 11:43 30340

EPAC Tape Recorder
17-JAN-74 1702-PDT

(J30340) 29-MAR-74 11:43; Title: Author(s): Robert M. Rodden/RMR2;
Sub-Collections: DEIS; Clerk: NDM;

DVN 29-MAR-74 11:47 30341

Agenda File, Message Cleanup
21-JAN-74 1233-PDT

Distribution: ecw deis

Agenda File, Message Cleanup
21-JAN-74 1233-PDT

21-JAN-74 1233-PDT VANNOUHUYS at SRI-ARC: Agenda File, Message cleanup
cc: energy
Received 21-JAN-74 12:33:45 1

I have extracted the agenda and some comments on it into <energy>agenda.nls;44. I have deleted the old message file and it will start fresh over from now. 1a

DVN 29-MAR-74 11:47 30341

Agenda File, Message Cleanup
21-JAN-74 1233-PDT

(J30341) 29-MAR-74 11:47; Title: Author(s): Dirk H. Van Nouhuys/DVN;
Sub-Collections: DEIS; Clerk: NDM;

AGC 29-MAR-74 11:50 30342

Justification for Renovations to SRI-Wash
25-JAN-74 0942-PDT

Distribution: deis (charlotte sakellaris of the SRI-W office)

Justification for Renovations to SRI-Wash
25-JAN-74 0942-PDT

25-JAN-74 0942-PDT ENERGY at SRI-ARC: FR CAPPS FOR SAKELLARIS RE JUSTIFICATION FOR RENOVATION
cc: ENERGY
Received 25-JAN-74 09:42:59 1

FOR CHARLOTTE SAKELLARIS FROM CAPPS - 1a

JUSTIFICATION FOR RENOVATION OF SPACES IN SIR WDC FOR ENERGY PROJECT FOLLOWS: TWO ENERGY ANALYSIS CENTERS ARE PLANNED - - ONE IN MENLO, ONE IN WDC - - AS INTERACTING DESIGN TEST/DEMONSTRATION VEHICLES IN SUPPORT OF SRI'S DESIGN WORK. RENOVATION REQUESTED IS NECESSARY IN ORDER TO PROVIDE ADEQUATE ROOM SIZE AND ALLOW PROPER SPACE/EQUIPMENT RELATIONSHIP. SIMILAR RENOVATION IS BEING ACCOMPLISHED IN MENLO (ROOM K 3008). 1b

AGC 29-MAR-74 11:50 30342

Justification for Renovations to SRI-Wash
25-JAN-74 0942-PDT

(J30342) 29-MAR-74 11:50; Title: Author(s): Arlie G. Capps/AGC;
Sub-Collections: DEIS; Clerk: NDM;

NDM 29-MAR-74 11:55 30343

All Messages to Date in NLS
30-JAN-74 1505-PDT

Distribution: jcn dvn deis

All Messages to Date in NLS
30-JAN-74 1505-PDT

30-JAN-74 1505-PDT MEYER at SRI-ARC: All Messages to Data in NLS
cc: energy, norton, vannouhuys
Received 30-JAN-74 15:05:53 1

I have just finished copying all messages sent to the directory <ENERGY> at both SRI-ARC and ISI into an NLS file at SRI-ARC called <ENERGY>MAIL.NLS. I have also renamed the source files to be: at ISI: message.10-jan-74
.25-jan-74
.29-jan-74
at SRI: oldmess.18-jan-74
.30-jan-74
Protection on all these files is 666600. 1a

All Messages to Date in NLS
30-JAN-74 1505-PDT

(J30343) 29-MAR-74 11:55; Title: Author(s): N. Dean Meyer/NDM;
Sub-Collections: DEIS; Clerk: NDM;

MAP 29-MAR-74 12:35 30344

gasp (last?)

in <mit-multics>usage;5 at office-1 is what I fervently
hope is the last (=final) version of the neted
usage thing, please look at it and tell me if you find any
substantive screwups which should prevent me from laying in
on USING as our "final report" as soon as I can, please
also tell me how yr implementations are doing (and please
finish doing them as soon as possilbe), for that matter,
please reassure me you're all alive (except wayne, who was
alive as recently as yesterday),
exhausted cheers, map

MAP 29-MAR-74 12:35 30344

gasp (last?)

(J30344) 29-MAR-74 12:35; Title: Author(s): Michael A. Padlipsky/MAP; Distribution: /NETED; Sub-Collections: NIC NETED; Clerk: MAP;

JCN 29-MAR-74 12:43 30345

DEIS Proposal to D. Russell
2-FEB-74 1810-PST

Distribution: deis

DEIS Proposal to D. Russell
2-FEB-74 1810-PST

2-FEB-74 1810-PST NORTON at OFFICE-1: Deis Proposal to D Russell
cc: energy at SRI-ARC, norton at SRI-ARC, norton
Received 2-FEB-74 18:13:19 1

I had a link with Dave Russell and he does have the DEIS Proposal we sent Friday night and is studying it now. Jim Norton 1a

DEIS Proposal to D. Russell
2-FEB-74 1810-PST

(J30345) 29-MAR-74 12:43; Title: Author(s): James C. Norton/JCN;
Sub-Collections: DEIS; Clerk: NDM;

clarity

it's clear don, very clear. 1

clarity

(J30346) 29-MAR-74 12:44; Title: Author(s): Barbara R. Sternick/BRS2; Distribution: /DON; Sub-Collections: NIC; Clerk: BRS2;

DON 29-MAR-74 12:54 30347

on-line mail ddelivery

Everything seems ok now, jiimm, thanks a lot. don 1

DON 29-MAR-74 12:54 30347

on-line mail ddelivery

(J30347) 29-MAR-74 12:54; Title: Author(s): Don Cantor/DON;
Distribution: /JEW; Sub-Collections: NIC; Clerk: DON;

DCR2 29-MAR-74 13:02 30348

Draft Proposal for a Preliminary Analysis for DEIS-1
5-FEB-74 0705-PDT

Distribution: ehw rab4 jcn eps deis

Draft Proposal for a Preliminary Analysis for DEIS-1
5-FEB-74 0705-PDT

5-FEB-74 0705-PDT RUSSELL: DRAFT PROPOSALFOR A PRELIMINARY ANALYSIS FOR DEIS-1
cc: ENERGY at SRI-ARC, WILLIS, BLACK, RUSSELL, NORTON at OFFICE-1, SCHELONKA
Received 5-FEB-74 07:05:13 1

FOR A.G. CAPPS, OTHERS FOR INFORMATION. REFERENCE YOUR MEMORANDUM OF 1 FEB 74 ON A DRAFT PROPOSAL FOR A PRELIMINARY ANALYSIS FACILITY FOR DEIS-1. DR. WILLIS, RUDY BLACK AND I HAVE REVIEWED YOUR DRAFT PROPOSAL, THE FOLLOWING CHANGES ARE SUGGESTED TO BRING IT MORE IN LINE WITH THE SITUATION IN THE DEIS AS SEEN FOR THE BALANCE OF FY74. 1a

1. PROVIDE THE DOD ENERGY OFFICE AND DSA ONLY TYPEWRITER TERMINALS, 1 TERMINET, 1 ACOUSTIC COUPLER AND 1 TI FOR THE DEFENSE ENERGY OFFICE AND 1 TERMINET FOR DSA/DFSC. I HAVE SPOKEN TO LT COL SCHELONKA OF RML CONCERNING THIS RQUIREMENT. HE SUGGESTS THAT HE CAN OBTAIN THIS EQUIPMENT THRU DECCO ALMOST IMMEDIATELY USING AN ARPA PRIORITY. JIM NORTON, PLEASE CALL HIM DIRECTLY IF YOU SEE ANY PROBLEM IN GOING THIS WAY. 1a1

2. DEFER AN NLS DISPLAY CAPABILITY IN THE DEFENSE ENERGY OFFICE UNTIL FY75. 1a2

3. PROVIDE THE DIRECTOR OF ENERGY AND DSA SUFFICIENT HELP TO USE AND UNDERSTAND TENEX SNDMSG AND TECO. 1a3

4. WORK WITH THE MIT MULTICS GROUP IN DEVELOPING THE INITIAL DEIS-1 ANALYTICAL CAPABILITIES DESCRIBED IN SECTION IIIA OF YOUR PROPOSAL. 1a4

5. USE THE WASHINGTON EPAC PROPOSED IN SRI PROPOSAL ISU 74-25 TO DEVELOP THE ANALYTICAL CAPABILITIES DESCRIBED IN 4 ABOVE AND AS A PLACE TO DEMONSTRATE THESE CAPABILITIES TO THE STAFF OF THE DOD DIRECTOR OF ENERGY. 1a5

PLEASE ADVISE IF THESE MODIFICATIONS ARE ACCEPTABLE. I AM PROCEEDING WITH A REQUEST TO THE DIRECTOR TO FUND YOUR PROPOSAL ISU 74-25. 1b

DCR2 29-MAR-74 13:02 30348

Draft Proposal for a Preliminary Analysis for DEIS-1
5-FEB-74 0705-PDT

(J30348) 29-MAR-74 13:02; Title: Author(s): David C. Russell/DCR2;
Sub-Collections: DEIS; Clerk: NDM;

RAS 29-MAR-74 13:09 30349

Include Black in DEIS Distributions
4-FEB-74 1052-PDT

Distribution: agc deis

RAS 29-MAR-74 13:09 30349

Include Black in DEIS Distributions
4-FEB-74 1052-PDT

4-FEB-74 1052-PDT ENERGY at SRI-ARC: MESSAGE FOR CAPPS RE CC TO BLACK
cc: ENERGY
Received 5-FEB-74 08:09:00 1

BLACK CALLED THIS MORNING RE HIS COPY OF THE PROPOSAL THAT WAS SENT TO RUSSELL. EVIDENTLY HE DID NOT RECEIVE A COPY OF IT BUT THOUGHT HE SHOULD HAVE. WE WILL SEND HIM A COPY OF THE PROPOSAL THIS, BUT IN THE FUTURE, WOULD YOU INCLUDE HIM ON THE DISTRIBUTION LIST OF ANYTHING GOING TO RUSSELL. THANKS. - - ENERGY@SIR-ARC 1a

RAS 29-MAR-74 13:09 30349

Include Black in DEIS Distributions
4-FEB-74 1052-PDT

(J30349) 29-MAR-74 13:09; Title: Author(s): Richard A. Schmidt/RAS;
Sub-Collections: DEIS; Clerk: NDM;

AGC 29-MAR-74 13:16 30350

Draft Proposal Dated February 1, 1974
5-FEB-74 0835-PST

Distribution: drb dce ras deis

Draft Proposal Dated February 1, 1974
5-FEB-74 0835-PST

5-FEB-74 0835-PST ENERGY at OFFICE-1: To Rodden From Capps
cc: Energy at SRI-ARC
Received 5-FEB-74 08:33:47 1

To Rodden From Capps 1a

Subject: Draft Proposal Dated February 1, 1974 1b

Ed and I met with Col, Russell 0900 this date and got the following message concerning subject proposal. 1c

1. ARPA does not want at the outset to provide the full capability we describe. 1c1

2. Instead, they will propose to provide Adm. Sonenshein's office one Terminet plus coupler and one TI 700 and one Terminet plus coupler at DSA. 1c2

3. SRI EPAC's will provide limited analytical services. Above position will be discussed between ARPA and Director of Energy (Adm. Sonenshein) within next few days. We will be instructed as to how to rework our proposal subsequently. 1c3

New but related subject, Col. Russell is preparing ARPA order on Norton proposal to provide ARC support to SRI EPAC's. We should hear soon that this is signed by Lukasik. 1d

Another related topic Col. Russell stated that the pending review with R&D representatives on the initial module for R&D (DEIS 1R) may be a major decision point in our program. I interpret this to mean that we should put serious effort into developing in detail what the DEIS 1R would do for the R&D community and how it would work. I urge we put priority on this and have something ready to discuss with Black in approximately one week. Let us discuss this item as soon as possible. 1e

AGC 29-MAR-74 13:16 30350

Draft Proposal Dated February 1, 1974
5-FEB-74 0835-PST

(J30350) 29-MAR-74 13:16; Title: Author(s): Arlie G. Capps/AGC;
Sub-Collections: DEIS; Clerk: NDM;

CGK 29-MAR-74 13:19 30351

360 JCL Manual
5-FEB-74 0841-PDT

Distribution: drb edr2 deis

360 JCL Manual
5-FEB-74 0841-PDT

5-FEB-74 0841-PDT ENERGY at SRI-ARC: TO ED RODRIGUES FROM C. KERNS, REGARDING 360 MANUALS
cc: ENERGY
Received 5-FEB-74 08:41:50 1

ED RODRIGUES: 360 JCL MANUAL IS BEING SHIPPED BY IDA RICHARDS, OTHER MANUAL IS BEING ORDERED AND WILL BE SENT ON ARRIVAL HERE. ADDITIONAL MANUALS MAYBE ORDERED ACCORDING TO PROCEDURES BEING MAILED TO YOU BY IDA. CARROL 1a

CC: DAVE BROWN, CARROL KERNS 1b

CGK 29-MAR-74 13:19 30351

360 JCL Manual
5-FEB-74 0841-PDT

(J30351) 29-MAR-74 13:19; Title: Author(s): Carrol G. Kerns/CGK;
Sub-Collections: DEIS; Clerk: NDM;

ARPANET Supporting Documentation
5-FEB-74 1418-PDT

Distribution: cgk deis

ARPANET Supporting Documentation
5-FEB-74 1418-PDT

5-FEB-74 1418-PDT ENERGY: TO: CAROL KERNS
cc: ENERGY at SRI-ARC
Received 5-FEB-74 14:18:26 1

FROM: ED RODRIGUES RE: ARPANET SUPPORTING DOCUMENTATION 1a

TO FACILITATE OPERATIONS AT DSA/DFSC, ADDITIONAL COPIES OF THE FOLLOWING DOCUMENTS ARE REQUIRED: 1b

1. "TNLS BEGINNERS GUIDE", PUBLISHED BY ARC 1c

2. "SRI-ARC TENEX USER'S GUIDE", PUBLISHED BY ARC 1d

3. "TENEX EXECUTIVE LANGUAGE," BOLT, BERANEK AND NEWMAN INC.,
CAMBRIDGE, MASS., JANUART, 1971. 1e

4. "TENEX USER'S GUIDE," BOLT, BERANEK AND NEWMAN INC. 1f

5. "TERMINAL INTERFACE MESSAGE PROCESSOR USER'S GUIDE",
BOLT, BERANEK AND NEWMAN INC. END OF MESSAGE. 1g

ARPANET Supporting Documentation
5-FEB-74 1418-PDT

(J30352) 29-MAR-74 13:24; Title: Author(s): Edward D, Rodrigues/EDR2; Sub-Collections: DEIS; Clerk: NDM;

AGC 29-MAR-74 13:26 30353

Draft Proposal for a Preliminary Analysis for DEIS-1
5-FEB-74 1514-PDT

Distribution: ehw rab4 rmr2 jcn deis

Draft Proposal for a Preliminary Analysis for DEIS-1
5-FEB-74 1514-PDT

5-FEB-74 1514-PDT ENERGY: DRAFT PROPOSAL FOR A PRELIMINARY ANALYSIS FOR DEIS I.
cc: ENERGY at SRI-ARC
Received 5-FEB-74 15:14:57 1

TO: COL. DAVE RUSSELL, OTHERS FOR INFO. FROM: A. G. CAPPS CC: WILLIS, BLACK, RODDEN, NORTON 1a

REFERENCE YOUR SUGGESTED MODIFICATIONS TO MY DRAFT PROPOSAL TO YOU DATED 1 FEB. 74. SRI HAS REVIEWED AND ACCEPTS THE INDICATED MODIFICATIONS. WE ARE RE-EXAMINING THE COST PORTION OF OUR DRAFT PROPOSAL IN LIGHT OF THE MODIFICATIONS AND POSSIBLE RELATED COST IMPLICATIONS AND WILL BE PREPARED TO DISCUSS WITH YOU AT YOUR CONVENIENCE. 1b

AGC 29-MAR-74 13:26 30353

Draft Proposal for a Preliminary Analysis for DEIS-1
5-FEB-74 1514-PDT

(J30353) 29-MAR-74 13:26; Title: Author(s): Arlie G. Capps/AGC;
Sub-Collections: DEIS; Clerk: NDM;

BJN 29-MAR-74 13:28 30354

Security Office Records on Contract No, DACA23-73-C-0014, Project 2513
7-FEB-74 1308-PST

Distribution: rmr2 ras deis

BJN 29-MAR-74 13:28 30354

Security Office Records on Contract No, DACA23-73-C-0014, Project 2513
7-FEB-74 1308-PST

7-FEB-74 1308-PST JORDAN at OFFICE-1: To Rodden/Schmidt from B, Neitzel
cc: energy
Received 8-FEB-74 06:21:49 1

The SRI-WASH Security Office has no paper work to show that Contract No, DACA23-73-C-oo14, Project 2513, has been extended. As far as this office is concerned the contract ended December 31, 1973 and my visit requests were cancelled. I managed to get them re-instated, but our Security Office said that Contracts-Menlo should send a copy of the Project Authorization to SRI-WASH Security Office ASAP. 1a

BJN 29-MAR-74 13:28 30354

Security Office Records on Contract No. DACA23-73-C-0014, Project 2513
7-FEB-74 1308-PST

(J30354) 29-MAR-74 13:28; Title: Author(s): Betty J. Neitzel/BJN; Sub-Collections: DEIS; Clerk: NDM;

more

note, by the by, that I've bought the "soandso written," 1
formulation for the file written out message (per Chris 2
and Wayne). cheers, map 3

more

(J30355) 29-MAR-74 13:29; Title: Author(s): Michael A. Padlipsky/MAP; Distribution: /NETED; Sub-Collections: NIC NETED; Clerk: MAP;

VGC 29-MAR-74 13:33 30356

Changeing Phone Number for Vint Cerf

To all and sundry: Stanford is going CENTREX on April 22, 1974 and my new telephone number will be 497-1365, Good Luck! Vint Cerf 1

VGC 29-MAR-74 13:33 30356

Changeing Phone Number for Vint Cerf

(J30356) 29-MAR-74 13:33; Title: Author(s): Vinton G. Cerf/VGC; Distribution: /NAG INWG NSAG; Sub-Collections: NIC NAG INWG NSAG; Clerk: VGC;

RAS 29-MAR-74 13:34 30357

Message from Russell of TI Terminal
6-FEB-74 1420-PDT

Distribution: agc deis

Message from Russell of TI Terminal
6-FEB-74 1420-PDT

6-FEB-74 1420-PDT ENERGY at SRI-ARC: MESSAGE FOR CAPPS
cc: ENERGY
Received 8-FEB-74 15:48:33 1

ARLIE, PLEASE DONT FORGET TO RETRIEVE THE MESSAGE FROM DAVE RUSSELL REGARDING THE TI TERMINAL AS PART OF OUR LATEST PROPOSAL, SCHMIDT/EPAC-WEST 1a

RAS 29-MAR-74 13:34 30357

Message from Russell of TI Terminal
6-FEB-74 1420-PDT

(J30357) 29-MAR-74 13:34; Title: Author(s): Richard A. Schmidt/RAS;
Sub-Collections: DEIS; Clerk: NDM;

RAS 29-MAR-74 13:36 30358

State of EPAC=west
7-FEB-74 0802-PDT

Distribution: jcn deis

State of EPAC=west
7-FEB-74 0802-PDT

7-FEB-74 0802-PDT ENERGY at SRI-ARC: epac-west
cc: norton, energy, schmidt at OFFICE-1
Received 8-FEB-74 15:48:53 1

jim, the carpet has been installed in epac-west, and we have reserved a place of honor for the delta data display and the other equipment that arc will be providing to the epac. we would like to discuss with you the schedule for installing this gear. it would be ideal from our standpoint and from the overall project view if the installation could be begun this week. also, there will be an energy committee meeting next week, and it wpuld be a good thing if we could have it in epac and show s e of the capabilities that are to be part of the deis, can we talk about this later today? dick schmidt 1a

RAS 29-MAR-74 13:36 30358

State of EPAC=West
7-FEB-74 0802-PDT

(J30358) 29-MAR-74 13:36; Title: Author(s): Richard A. Schmidt/RAS;
Sub-Collections: DEIS; Clerk: NDM;

something happened there,,,,,I don't know where I am, If this looks okay, I'll have about 50 copies made up; should I send them out as the requests come in, or should we each keep a stack of them (more personal, but less efficient.)

MIKE 29-MAR-74 13:40 30359

Proposed Form Letter for ARPAnet News-based Requests

ARPAnet News readers - A Form Reply 1

On behalf of the other members of Bell Canada's Business Planning Group, I would like to thank you for you interest in the work of the department (as overviewed in the ARPAnet News for the month of March). 1a

The papers abstracted in the newsletter represent a few of the volumes in the aperiodic series, Business Planning Papers. This series was initiated primarily for the presentation of working papers to our senior management on an informal basis. As such, they are not normally available to the general public. It is not the confidentiality of the information that imposes theses restrictions, but rather our limited budget for printing expenses. (The references in the ARPAnet News would have pointed this out if we had known about them prior to their appearance.) 1b

We recognize that the information contained in the Business Planning Papers may be of value to other individuals or groups or individuals doing similar research in these areas. In fact we have have adopted the policy of trading them freely with other groups who have similar trading data available. 1c

If you are involved in research similar to our interests in the "wired city", in travel/communications substitutability, or in technology assessment and corporate social responsibility, we would be most interested in hearing about your work. 1d

I regret that we are unable to supply the information you requested, but I look forward to our continuing communications in these subjects. 1e

(Id)
Yours sincerely,

Michael T. Bedford
Supervisor - Business Planning 1f

MIKE 29-MAR-74 13:40 30359

Proposed Form Letter for ARPAnet News-based Requests

(J30359) 29-MAR-74 13:40; Title: Author(s): Michael T. Bedford/MIKE;
Distribution: /DAY MIKE; Sub-Collections: NIC; Clerk: MIKE;

Army Corps of Engineers
14-FEB-74 2022-PDT

14-FEB-74 2022-PDT SCHELONKA: ARMY CORPS OF ENGINEERS
cc: SCHELONKA
Received 14-FEB-74 20:22:23 1

TO PAM KRUZIC Hello there, My original schedule for the briefing was delayed until the 15th and then a very severe time crunch occurred so that I could not go up but sent up the ARPANET movie and a copy of my briefing slides, If the Army group is getting interested, have them come down here for a visit where I could spend some time with them showing them actual equipment and methods of using the net in addition to all the background information, Best Regards.....Ed Schelonka 1a

EPS 29-MAR-74 13:42 30360

Army Corps of Engineers
14-FEB-74 2022-PDT

Distribution: pgk deis

EPS 29-MAR-74 13:42 30360

Army Corps of Engineers
14-FEB-74 2022-PDT

(J30360) 29-MAR-74 13:42; Title: Author(s): Edward P. Schelonka/EPS;
Sub-Collections: DEIS; Clerk: NDM;

this is just a test message. 1

(J30361) 29-MAR-74 14:22; Title: Author(s): Virginia L. Kallal/VLK;
Distribution: /VLK; Sub-Collections: NIC; Clerk: VLK;

IWC 29-MAR-74 19:30 30363

0TH BIRTHDAY GREETINGS

DEAR TOM: THE ARPANET AND ALL US MIDNIGHT SKULKERS WOULD LIKE TO WISH YOU WELL ON THE ADDITION OF A NEW TURNPYKE (SIMILAR TO A TOMPYKE ONLY SMALLER). THIS FEEBLE MESSAGE COMPOSED ON HOME TELEVISION USING DIGILOG. PS: MARCY WANTS TO WISH CAROL WELL FOR HER SMALL ROLE. 1

IWC 29-MAR-74 19:30 30363

OTH BIRTHDAY GREETINGS

(J30363) 29-MAR-74 19:30; Title: Author(s): Ira W. Cotton/IWC;
Distribution: /TNP; Sub-Collections: NIC; Clerk: IWC;

JOVIAL Manual--Chapter 7

(J30364) 30-MAR-74 11:08; Title: Author(s): Duane L. Stone/DLS;
Distribution: /RJC; Sub-Collections: RADC; Clerk: DLS;
Origin: <CARRIER>C7,NLS;2, 28-MAR-74 20:56 RJC ;

Contains structure and _-" font markers, (newword plex so long that program aborted on the sort...should find out upper limit)

Chapter 7 1

"DECLARATIONS 1a

7.1 Introduction 1a1

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

7.2 Undefined and Predefined "Names 1a2

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

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

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

of the program. Definitions to be associated with "names as if through the use of _DEFINE may occur in either the compool, the library, or both. 1a2a2

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

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

7.3 Scope of Definition of "Names 1a3

Before considering the various "declarations, it is important to understand the concept of scope because it is this concept that determines the portion of the "program:declaration or system of "program:declarations in

which the declared "name is active. The scope of a "name then is defined as that segment of code over which a "name has meaning. 1a3a

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

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

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

.4 Within a "program:declaration, scopes are defined by the "program:declaration itself and also by all "procedure:declarations within the "program:declaration. Data declared in the "program:declarations within the "program:declaration but not within any "procedure:declaration is of main scope. Data declared within a "procedure:declaration is of procedure scope. 1a3a4

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

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

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

.8 While uniqueness of "names is required within a scope, "names can be repeated in different scopes. To the compiler, a "name comprises both a spelling and a scope; "names with the same spelling but different scope are easily distinguishable and not the same "name at all. Thhr h'r bdrt'hn 'dv'nt'fdr fnr JNVH@L programmers. It allows a freer choice of "names within a "program:declaration. The programmer is able to create "names without regard for total uniqueness; his worries are confined to the local scope. Also, an independently created "procedure:declaration can be incorporated into a "program:declaration without fear that its local "names will be in conflict with any identically spelled existing "names. 1a3a8

.9 The scope of a "name local to a "procedure:dclaration

is the "procedure:declaration in which it is defined and all contained "procedures:declarations that do not have their own definitions of the "name. A main scope "name is defined wherever in the "program:declaration inner "procedure:declarations do not have local definitions of the same "name. 1a3a9

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

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

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

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

7.4 "Declarations 1a4

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

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

7.5 "Null:Declaration 1a5

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

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

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

7.6 "Data:Declarations 1a6

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

.1 Much of the remainder of this chapter is concerned with the specification of the various kinds of "data:declarations. First, some concepts common to all data structures are developed. Then the individual "declarations are considered. 1a6a1

7.7 Fixed and Controlled Allocation 1a7

The data structures about to be described in the various

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

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

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

.3 Controlled allocation of data space means that space assignment for a particular data structure is established by the progr'm dynamicallx 't run time as opposed to being fixed during compilation or during linking and loading of the program, or by the system during activation of a scope. Data structures are identified as controlled allocation structures by the inclusion of an "allocation:specifier within the "data:declaration. Pointers are thereby established which locate the data structure. 1a7a3

.4 Controlling the location of data structures within dynamic space is left to the JOVIAL programmer who, by assigning values to pointers, establishes at least temporary allocation of a pointed-to structure. Each reference to a variable declared to have controlled

allocation must employ, either explicitly or implicitly, an associated pointer. The value of the pointer will be used in the calculation of the effective address of the structure. 1a7a4

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

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

```
_GETSPACE (10 : POINTER1) ;                    1a7a6a
```

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

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

.8 Data space which is compiled into the program or which is reserved at load time can have controlled allocation structures dynamically associated with it just as can space which is reserved at execution time. If a program were required to operate on several different

data structures but could arrange its processing sequence to finish processing some of the structures before starting to process others, data space in the program could be kept to a minimum if the program were to use controlled allocation to overlay different structures on the same space. Such space can be reserved at compile time by declaring a table or data block large enough for the requirement. The structures to be processed would be declared for controlled allocation and at appropriate points in the execution of the program the values of the pointers to the controlled allocation "variables would be set to equal addresses within the large table. 1a7a9

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

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

7.8 Pointers and Their Association with Structures 1a8

Controlled allocation structures are also called pointed-to rtrtbttrdr. Thhr n'ld dxprdrrdr thd requirement that there be a pointer that locates these structures as space is dynamically allocated them. The pointer is expressed as a "pointer:formula. 1a8a

.1 "Pointer:formulas describe the address of pointed-to structures. The "pointer:formula may contain "pointer:variables as well as other "numeric:variables and "constants. The result of the evaluation of the "formula (truncated to an integer) is the pointer value used in referencing the structure. ("Pointer:formula is

simply a special case of the "numeric:formula--one which evaluates to a location.) 1a8a1

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

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

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

```
_iTEM  "name              "pointer:formula
_U  ;
```
1a8a4a

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

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

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

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

"variable and the compiler will supply the associated pointer. 1a8a8

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

```
"name      _[     "index      _]  _@
"pointer:formula                          1a8a9a
```

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

7.9 "Allocation:Specifier 1a9

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

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

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

.3 Absence of the "pointer:formula indicates that while the data structure is indeed pointed to, there is to be no pointer associated with the structure at the point of "declaration. The required association must be made explicitly at some subsequent point during reference. 1a9a3

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

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

7.10 Data Permanence 1a10

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

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

.2 Data that is environmental to the main scope is said to be "reserved" or "in reserve." All data declared in a "program:declaration, whether it be of main scope or procedure scope, automatically becomes reserved unless its permanence is restricted by being made private to some inner scope by the occurrence of an "environmental:specifier. Data that is to be initialized or preset must be in reserve. Reserved data are allocated when a program is loaded and remain until the program in unloaded even though the program is not active (executing). A program can be entered and exited many times and at each entry it can rely on the validity of reserved data as long as the program has not been unloaded. The loading and unloading is a function of a

program's use as part of link-edited segments and of the loader employed by the operating system; in short, loading and unloading is system-dependent and outside the scope language. 1a10a2

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

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

.5 Data generated by the compiler incidental to the processing of forms other than "data:declarations become part of the unnamed data space of the procedure. This may include things such as temporary procedure space, register save areas, return address, perhaps some parameter space, and other linkage convention space. As a part of the procedure's data space they will be in reserve unless the entire data space is restricted by being made private to some lesser scope (see Section 8.6). 1a10d

7.11 "Environmental:Specifier 1a11

The "environmental:specifier is an optional part of "data:declarations. It is normally used to restrict the permanence of data by making it private to some scope. Without the restrictions imposed hereby the data being

declared would become environmental to the program and in reserve. 1a11a

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

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

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

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

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

.6 A peculiar situation arises if an inner scope "procedure:name, "alternate:entrance:name or "statement:name has been made external and then invoked externally. The reserve data associated with this called program comes into existence when the called program is

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

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

.8 The "environmental:specifier and the "allocation:specifier must not appear in the same "declaraton. Controlled data are neither private nor reserved, although every pointed-to datum must be ultimately tied to a private or reserved pointer, perhaps through a long chain of pointed-to pointers. The permanence of controlled data depends on both the permanence of their pointers and the execution of relevant "statements. Certainly if a program or procedure obtains space from the system and allocates controlled data to that space, the controlled data exist so long as the program or procedure is active, the space is not relinquished, and all elements of the pointer exist. It is entirely the programmer's responsibility if he allocates other data to that same space. If the pointer ceases to exist, then so does what it points to (unless another pointer has taken up the burden). It is dependent on the system whether space is automatically relinquished when the program or procedure that has obtained it goes inactive. 1a11a8

7.12 "Packing:Specifier 1a12

The "packing:specifier directs the compiler in the

positioning of items or entries in a word. The information it provides is also used in the generation of code to access the items or entries in the most efficient manner. 1a12a

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

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

.3 _M specifies medium packing. Although this degree of packing is machine dependent, the intention is to provide an effective compromise between item accessibility and space utilization. An additional degree of specification is provided for medium packing by following the _M with a "number. This "number represents a machine-dependent field size or pattern to be used in arranging space for the item. For instance, _M _2 might indicate on a particular compiler that the item is to be allocated to a half word. 1a12a3

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

.5 For a specified table in which the programmer arranges the items within an entry or for a simple item which the programmer positions in a word, the "packing:specifier does not direct the compiler in the packing of the items. It does, however, inform the compiler of any convenience provided in the specified packing. This information is used by the compiler as it generates code to reference the items. Normally, the "packing:specifier should be at least as dense as the actual packing of the item. If it is not, for example, if an item actually medium packed is designated as being

unpacked, code may result that can erroneously disturb surrounding items. 1a12a5

7.13 "Constant:List 1a13

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

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

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

```
_3 ( + 5,1 )  = + 5,1 , + 5,1 , + 5,1                    1a13a1a1

_2 ( -4,05 , ) = -4,05 , , -4,05 ,                       1a13a1a2

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

__2 ( 'NO' , 3 ( 'YES' )) = 'NO' , 'YES' , 'YES' ,
'YES' ,                                                  1a13a1a4

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

For every _constant:list written in terms of this repetition notation, there exists, therefore, an equivalent, "parenthesis free, fully expanded "constant:list. The repetition notation is available as a shorthand for the user. The compiler algorithms which associate the "constant:list with the table to be initialized are described in terms of the equivalent fully expanded "constant:list. 1a13a1b

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

a. A pseudo-entry counter is set to the first

entry, of the first row, of the first plane, etc. 1a13a1b1a

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

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

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

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

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

g. The fully expanded "constant:list may be thought of as consisting of "constants and nulls separated by "commas and "indices. Between any two "commas there is either a "constant or a null. Between a "comma and a following "index there is either a "constant or a null. After each "index there is either a "constant or a null. Before a "comma beginning the list or after a "comma ending the list there is a null. The pointer is now moved from the null, "constant, or "index it points to so that it then points to the next null, "constant, or "index, unless there are no more nulls, "constants, or "indices in the list, in which

case this presetting process is completed. Otherwise, return to step c. 1a13a1b1g

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

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

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

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

. 1a13a1b3c

. 1a13a1b3d

. 1a13a1b3e

. 1a13a1b3f

The plane within that volume. 1a13a1b3g

The row within that plane. 1a13a1b3h

The particular entry within that row. 1a13a1b3i

.6 The ability to include "indices within the "constant:list allows the user to specify that different parts of the "constant:list are to be applied to different parts of the table. In terms of the processing algorithms, the "indices supply new values for the pseudo-entry counter. The

counter can be increased or decreased. There is a risk, however, that the "constant:list may be reapplied to entries already initialized. Whenever multiple initialization of the same bits is attempted the result is undefined. 1a13a1b4

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

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

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

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

7.14 Data Structures 1a14

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

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

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

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

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

7.15 "Item:Declarations 1a15

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

.1 In the paragraphs to follow, first the "item:description is discussed as an element common to all "item:declarations. Then comes the "status:list:declaration which is referenced within

^item:descriptions. Next, the ^simple:item:declaration is specified. ^Ordinary:table:item:declarations and ^specified:table:item:declarations are then discussed together. Each will also be individually discussed in brief with their corresponding ^table:declarations. 1a15a1

7.16 ^Item:Description 1a16

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

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

_C	character	1a16a1a
_F	floating	1a16a1b
_S	signed (integer or fixed)	1a16a1c
_U	unsigned (integer or fixed)	1a16a1d

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

.3 For floating items an _R says to round upon setting the value of the item if the ^formula providing the value has extra bits in the significand--absence of the _R says do not round. The ^significand:specifier, if present, gives a minimum size of the significand in bits--excluding the sign. The size of the exrad is based on the assumption that the radix is 2 regardless of the actual representation of floating numbers. If both of these ^numbers are omitted, the default is the system-dependent single precision. If the system can provide alternative forms for floating values, it chooses one to accommodate the stated sizes of the significand and exrad. If it cannot do this, the ^declaration is in error. 1a16a3

.4 Signed and unsigned items can be integer or fixed. The _R, if present, says to round upon setting the value. If the _R is absent, do not round. The ^size:specifier, if present, gives the size of the item in bits--excluding the sign for signed items. If this ^number is omitted, the size is system-dependent--the size normally used by

the compiler for addresses (at least for unsigned items). If the "precision:specifier is present (even if its value is zero), the time is fixed and the "number (together with its sign, if present) tells how many fraction bits there are. This "number may be of any value. It may be positive and larger than the "number (or default) giving the size--in which case there are assumed or understood leading zeros between the binary point and the significant fraction bits. It may be negative--in which case there are trailing bits of unknown value between the significant integer bits and the binary point. 1a16a4

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

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

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

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

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

.3 The following is an example of a

^status:list:declaration incorporating a ^status:list with a gap in the sequence of values. 1a17a3

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

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

```
_V(Q) V(R) V(S) V(T) V(U) V(V)
_-2   -1   0   1   4   5
```
1a17a4a 1a17a4b

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

```
_STATUS LIST V(A), V(B), V(C), V(D), V(E);
_STATUS BOOL V(ON), V(OFF);
STATUS RANGE [-6] V(POOR), V(FAIR);
             [37] V(GOOD) [91] V(EXCELLENT);
_STATUS SIZE V(SMALL), V(MED), V(LARGE);
_STATUS TEMP [1] V(HI), V(MED), V(LO);
```
1a17a4c1 1a17a4c2 1a17a4c3 1a17a4c4 1a17a4c5 1a17a4c6

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

7.18 ^Simple:Item:Declarations 1a18

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

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

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

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

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

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

_N No packing 1a18a5a

_D Dense packing. 1a18a5b

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

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

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

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

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

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

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

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

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

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

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

.5 The "specified:table:item:declaration must include information to position the item within an entry. If the table is serial or parallel structured, the "bit:number gives the position of the first bit of the item in a word of an entry. Numbering of bits starts with zero on the ldft 'nd bntntr tn thd rhfht. Thd "vnrd:ntlbdr tells the word of the entry (starting with zero) in which the item resides or begins. For tight structured tables only the "bit:number, the position of an item within an entry, is specified. The position information is enclosed in "brackets. 1a19a5

.6 Initial values may be provided for items by means of the "constant:list. There must be no attempt to preset (i.e., no "constant:list) items that are not in reserve. The "constant:list provides initial values for the item in certain entries depending on the number of "constants and on the "indices in the "constant:list. If the size of the item is defaulted in the "item:description and any

"constant in the "constant:list requires a size greater than the default size, the largest size implied by the "constant:list becomes the size of the item. If the size of any "constant in the "constant:list exceeds the size stated in the "item:description, the "constant is truncated or rounded as if for assignment to the item. 1a19a6

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

7.20 "Table:Declarations 1a20

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

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

7.21 "Allocation:Increment 1a21

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

.1 For controlled allocation tables the "allocation:increment indicates the units in which space is to be allocated to the table. The units can be either an entry, an entire table, or some submanifold. (A manifold is: a row, a plane, a volume, and so on. A submanifold that is something less than the entire table.) 1a21a1

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

Value	Unit of Increment	1a21a2a
_0	_1 entry	1a21a2b
_1	_1 row	1a21a2c

```
        _2          _1 plane                                    1a21a2d
        _3          _1 volume                                   1a21a2e
        _4          _1 hyper volume                             1a21a2f
        _.                                                      1a21a2g
        _.                                                      1a21a2h
        _.                                                      1a21a2i
```

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

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

7.22 "Dimension:List 1a22

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

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

.2 "Lower (explicit or implicit) and "upper:bound are given for each dimension of the table. The number of "upper:bounds in the "declaration is the number of dimensions (_t) in the table. The "lower:bound (or the implied zero) and the "upper:bound in each position give the range of values for an "index:component in the corresponding position. In subsequent references to

^variables of this table, the corresponding component of the ^index must be within this range; an out-of-range ^index:component is undefined. 1a22a2

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

.4 The extent of the table in a particular dimension is ^upper:bound _+ 1 - ^lower:bound. The extent of the entire table is the product of the extent of each dimension of the table. A _3 x 3 x 3 volume has a size of _27 entries. 1a22a4

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

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

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

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

.2 The dimensionality of the ^pointer:formula within the ^allocation:specifier must be complementary with those of the ^allocation:increment and the table itself. If

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

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

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

7.24 "Structure:Specifier 1a24

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

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

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

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

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

.5 The basic reason for having both parallel and serial structured tables is efficiency. Serial tables lend themselves efficient indexed operations. It may be necessary, in some implementations, to establish prohibitions against situations which would run counter to the goals of efficiency. For example, implementation-specific requirements may force restrictions on the use of multiple-word numeric items in parallel tables. If the system includes machine instructions which depend upon sequential storage locations for multiple-word operands then, in the interest of efficient object code, it may be necessary to require these items be located in serially structured tables. 1a24a5

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

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

7.25 "Ordinary:Table:Declaration 1a25

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

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

7.26 "Ordinary:Table:Heading 1a26

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

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

.2 If the table is to be independently, dynamically allocated, the "allocation:specifier must be present. If the permanence of the table is to be restricted to something less than in reserve, the "environmental:specifier must be present. If neither an "allocation:specifier nor an "environmental:specifier is present, the table has fixed allocation and is in reserve. If the table is to be collected within a data block, there must be no "allocation:specifier or "environmental:specifier in the "ordinary:table:declaration. The desired effect must be achieved at the data block level via the "data:block:declaration. 1a26a2

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

regard to the programmer's intended usage and it affects the meaning of parallel storage. 1a26a3

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

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

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

.7 An ^item:description can be given to apply to the ^table:name. Its effect upon the size of the entry and its position within the entry are the same as if it were the description of a single item contained in the table. If the ^item:description occurs in the ^ordinary:table:heading, the ^ordinary:table:heading is then the entire ^ordinary:table:declaration and there is nn ^nrdhn'rx:t'ble:body. If this ^item:description is present, the ^table:name may be used as a ^table:variable. If this ^item:description is missing, such a reference means a reference to an ^entry:variable of type "bit". 1a26a7

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

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

7.27 "Ordinary:Table:Body 1a27

The "ordinary:table:body follows the "ordinary:table:heading if that heading did not include an "item:description. The "nrdhn'rx:t'bld:bndx lhrtr thnrd "htdl:ddclarations and "subordinate:overlay:declarations which make up an entry of the table. 1a27a

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

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

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

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

7.28 "Subordinate:Overlay:Declaration 1a28

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

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

in that same "ordinary:table:body may appear in the "subordinate:overlay:declaration and the "ordinary:table:item:declarations must precede the "subordinate:overlay:declaration. 1a28a1

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

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

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

Items _BY and _BZ are assigned nonoverlapping space yet the space they occupy is shared with item _BX; the size of _BX is larger than the combined sizes of _BY and _BZ so it, therefore, determines the amount of space controlled by the embedded "subordinate:overlay:expression. Items _BC and _BA are each assigned space that does not overlap either that of the other or the common space assigned _BX, _BY and _BZ. Items _AA, _AB, and _AC are, in the same way, assigned nonoverlapping space. Finally, the space assigned the _A-items does overlap the space assigned the _B-items. The size of the space controlled by the "subordinate:overlay:string referencing the _B-items is the larger of the two so it determines the space controlled by the "subordinate:overlay:declaration. 1a28a5

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

7.29 "Specified:Table:Declaration 1a29

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

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

7.30 "Specified:Table:Heading 1a30

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

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

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

.3 The size of an entry of a specified table and perhaps its disposition within a computer word is described by a "number or series of "numbers. If the table is of parallel or serial structure, the "words:per:entry must be scripted as a "number to tell how many computer words are occupied by each entry. For a tightly structured table the entries are located within a computer word by one, two, or three "numbers. "Bits:per:entry gives the size of an entry in bits. The size must be no greater than the size of a computer word. "Bit:number gives the position within the word of the first bit of the first entry (in each word). It defaults to zero.

"Entries:per:word tells how many entries go in each word. This must be compatible with the packing:specifier (which defaults to "dense"). If the third "number is absent, it defaults to the maximum possible in light of the other two "numbers and the "packing:specifier. 1a30a3

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

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

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

7.31 "Specified:Table:Body 1a31

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

.1 The "specified:table:body resembles the "ordinary:table:body. It can consist of a single "null:declaration. Also, the "specified:table:body may

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

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

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

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

7.32 "Data:Block:Declaration 1a32

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

.1 The "environmental:specifier and "allocation:specifier function the same as in other "data:declarations. If neither is present, the data block is environmental to the program and in reserve. The "environmental:specifier can make the data block less permanent by declaring it to be private to some encompassing procedure or program. The "allocation:specifier can provide controlled allocation for the data block. In this case, space is only

allocated to the data block dynamically in response to instructions within the object program and every reference to the data block or to the data declared within it requires resolution in terms of either an implicit or explicit pointer. 1a32a1

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

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

.4 "Independent:overlay:declarations can be used within the "data:block:declaration to arrange the named data structures of the data block. An "overlay:declaration within a "data:block:declaration is restricted to arranging elements declared within the data block; it cannot attempt to relate the data block to external members of the environment and is, therefore, prohibited from mentioning any named data structures that are not declared within the "data:block:declaration. 1a32a4

7.33 "Independent:Overlay:Declaration 1a33

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

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

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

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

.4 An "independent:overlay:expression usually expresses

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

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

_, T5), T15 1a33a5a

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

.6 The absolute location of data referenced in ^independent:overlay:declarations can be determined in several ways. Compool data occupy locations assigned outside the writing and compilation of the program. If a compool datum is mentioned in an ^independent:overlay:declaration, it thus fixes the locations of all data mentioned in the same ^independent:overlay:declaration. If the optional ^number or ^pattern:constant in ^brackets is included in an ^independent:overlay:declaration, it thus fixes (by whatever meaning the system ascribes to such a ^number) the location of the first overlay element, and therefore of all overlay elements, of the ^independent:overlay:expression. If any datum mentioned in an ^independent:overlay:declaration is also mentioned in an earlier ^independent:overlay:declaration, the previously allocated location fixes the location of all overlay elements of the current ^independent:overlay:expression. If none of these constraints are placed on the locations of the data, the compiler places the entire structure determined by the relationships of the ^independent:overlay:declaration in contiguous locations it considers efficient and convenient. It does not consider, in assigning these locations, any data overlaid onto absolute locations. 1a33a7

.7 When the ^independent:overlay:declaration contains the optional ^number or ^pattern:constant in ^brackets or

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

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

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

_OVERLAY    CC,EE,DD;                            1a33a9b

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

Although their entire structural relationship cannot be described in a single "independent:overlay:declaration, they form an interrelated and contiguous block of storage--which is allocated accordingly. Figure 7-4 depicts the allocation of space to the related structure. In this diagram the origin for each of the "independent:overlay:expressions is represented by an asterisk. 1a33a10

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

illegal to attempt to relate any data declared to be independently pointed to, via "overlay:declarations, to other pointed-to data, to fixed allocation data, or to absolute locations. 1a33a11

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

.11 An effective data block, just like a declared data block, has a certain degree of permanence. The permanence of a declared data block is expressed in the optional "environmental:specifier with a default to reserved. For an effective data block the permanence derives from the permanence of the contained data structures. All the structures of an effective data block must be at the same level of permanence; i.e., they must all be in reserve or all at the privacy level of a single procedure or program. If the effective data block created by one or more "independent:overlay:declarations is to be private to some scope, at least one of the overlay elements must be declared (by an appropriate "environmental:specifier) to be private to the selected scope. Other data elements may, but need not, be declared private to that same scope. None may be declared to be pointed to. None may have a contradictory "environmental:specifier. "Contradictory" can best be explained by means of an example: 1a33a13

```
_PROC P4;                                          1a33a13a
_BEGIN                                             1a33a13b
   ...                                             1a33a13b1
   _PROC P3 IN P4;                                 1a33a13b2
   _BEGIN                                          1a33a13b3
      _ITEM Q31, Q32 F;                            1a33a13b3a
```

```
        _OVERLAY Q31  :  Q32;                                  1a33a13b3b
        ...                                                    1a33a13b3c
        _PROC P2 IN P2;                                        1a33a13b3d
        _BEGIN                                                 1a33a13b3e
           _ITEM Q21 IN P3 F;                                  1a33a13b3e1
           ...                                                 1a33a13b3e2
           _PROC P1;                                           1a33a13b3e3
           _BEGIN                                              1a33a13b3e4
              _ITEM  Q11, Q21, Q31;                            1a33a13b3e4a
              _OVERLAY Q11, Q21, Q31;                          1a33a13b3e4b
              ...                                              1a33a13b3e4c
           END                                                 1a33a13b3e5
           ...                                                 1a33a13b3e6
        END                                                    1a33a13b3f
        ...                                                    1a33a13b3g
     END                                                       1a33a13b4
     ...                                                       1a33a13b5
  END                                                          1a33a13c
  ...                                                          1a33a13d
  END                                                          1a33a13e
```

.12 Procedure _P1 contains an "independent:overlay:declaration mentioning data local to "P1, P2, and _P3, respectively. This serves to make _Q11 private to _P3, due to its relationship to _Q31. It would serve to make _Q21 also private to _P3 except that the "procedure:heading for _P2 makes all the otherwise defaulted data of _P2 private to _P2. To avoid the contradiction, _Q21 must be declared private to _P3. In _P3 there is another "independent:overlay:declaration

relating _Q32 to _Q31. This adds _Q32 to the effective data block containing _Q31, _Q21, and _Q11. Now the "environmental:specifier in the "procedure:heading for _P3 makes this effective data block private to _P4, along with all the otherwise defaulted local data of _P3. This effective data block can only be at the privacy level of the unnamed data space of _P3--because the declaration for _Q21 specifically mentions _P3 in its "environmental:specifier. 1a33a14

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

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

a. Interference is "not recognized if it arises out of two data structures being independently overlaid on absolute locations. 1a33a16a

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

Similarly, it is the programmer's responsibility, with regard to pointed-to data, to make sure that the pointer values do not cause interference. The compiler, of

course, recognizes interference among overlaid data in a pointed-to data block or table. Example: 1a33a17

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

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

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

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

7.34 "Define:Declaration 1a34

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

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

```
_ITEM FLIP U;                                        1a34a1a

_PROC TRIP ;                                         1a34a1b

_BEGIN DEFINE FLIP " END FLIP " ;                    1a34a1c

      _ALPHA = BETA ** GAMMA ;                       1a34a1d

      _FLIP = DELTA / EPSILON ;                      1a34a1e
```

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

.2 A "comment is not permitted between the "define:name

and the "definition. Since the same "ideogram is used to delimit both a "comment and a "definition, the compiler will confuse one for the other and be unable to detect the error. 1a34a3

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

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

.4 Thd trd nf ' rhnfld "exclamation:point within a "definition is restricted to the identification of "formal:define:parameters (see Section 7.34.16). However, in a way similar to that for "quotation:marks, a convention has been established to allow the use of "exclamation:points in a "definition. Each desired "exclamation:point that does not identify a "formal:define:parameter should be scripted as two. At invocation, strings of "exclamation:points are halved as they are copies. 1a34a6

.6 "Define:names have scope. The rules are, as much as possible, the same as for all other declared "names. A "define:declaration local to any scope is not effective in any outer scope. An outer scope "define:name is effective within an inner scope unless there is some

explicit ^declaration using that ^name in the inner scope. For example, if _AA is a ^define:name in an outer scope, then declaring _AA as a ^name in an ^item:declaration or a ^name:declaration in an inner scope overrides the use of _AA as a ^define:name in the inner scope; declaring _AA in a ^define:declaration in the inner scope is a ^definition of _AA effective for the inner scope only and having no effect on the ^definition of _AA for the outer scope. 1a34a7

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

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

.9 No ^define:names are permitted in a ^procedure:declaration between the ^primitive _PROC and the first terminating ^semicolon. It is not always possible to differentiate between intended ^definition:invocations and the ^names of ^formal:input:parameters and ^formal:output:parameters of the ^procedure:declarations. Therefore, to avoid such ambiguities it is necessary to make ^define:names illegal in that range. 1a34a10

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

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

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

.13 A "define:name is not considered to be a "definition:invocation where it occurs as part of another "symbol such as a "comment, "character:constant, "status:constant or "name. A "definition:invocation may occur in another "definition even before its own "define:declaration; however, it is not recognized as a "definition:invocation except when the encompassing "definition is invoked and the substituted string os "symbols is examined as source code. Then the embedded "definition:invocation invokes its own "definition. Of course, circular or recursive "definitions are not permitted. 1a34a14

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

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

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

_DEFINE BETA " GAMMA + 3 " ;                                    1a34a16b

_DELTA = ALPHA + BETA ;                                         1a34a16c
```

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

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

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

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

_INC (ABC)                                                      1a34a18b
```

is now equivalent to: 1a34a19

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

The ^semicolon within the ^definition is not required; however, it affects the proper way to invoke the ^define:name 1a34a20

```
_INC (PDQ) ;                                                    1a34a20a
```

yields: 1a34a21

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

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

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

Substitution would not be made for the _F in _FOR or for

the first and second _K's because they are not preceded by "exclamation:points. 1a34a23

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

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

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

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

The "definition:invocation 1a34a25

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

would cause the resulting string 1a34a26

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

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

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

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

```
_DEFINE STAR "*" ;                                      1a34a28a

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

cannot be used to generate 1a34a29

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

Note the distinction between this and the previous case.

^Symbols can be created by proper positioning of ^formal:define:parameters within a ^definition but ^symbols cannot be created by juxtaposition of a string of ^characters substituted for a ^definition:invocation with the immediate environment of the ^definition:invocation. 1a34a30

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

a. A copy is made of the ^definition corresponding to the ^define:name. Sequences of two or more ^quotation:marks or two or more ^exclamation:points are halved during the copying. The processing of other ^definition:invocations is postponed until step e. 1a34a31a

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

c. The temporary ^definitions corresponding to the ^formal:define:parameters replace the occurrences of those ^formal:define:parameters in the copy. Sequences of two or more ^quotation:marks or two or

more "exclamation:points in the temporary "definitions are halved during the replacement. 1a34a31c

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

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

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

```
_DEFINE ASSIGN "=" ;                                          1a34a32a

_DEFINE MAX (A, B, C)                                         1a34a32b

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

and a "definition:invocation 1a34a33

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

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

```
_!C ASSIGN !A ;                                               1a34a34a

_IF !C < !B ;                                                 1a34a34b

_!C ASSIGN !B ;                                               1a34a34c
```

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

_A is associated ith _ALPHA 1a34a35a

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

_C is associated with _F1 1a34a35c

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

```
   _F1 ASSIGN ALPHA ;                                    1a34a36a

   _IF F1 < ARCTAN(AA,BB) ;                              1a34a36b

   _F1 ASSIGN ARCTAN(AA,BB) ;                            1a34a36c
```

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

```
   _F1 = GAMMA + 3 + 2 ;                                 1a34a37a

   _IF F1 < ARCTAN(AA,BB) ;                              1a34a37b

   _F1 = ARCTAN(AA,BB) ;                                 1a34a37c
```

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

```
   _DEFINE FUNC(A, B) "(!A + !B)/2 ;                     1a34a38a

   _DEFINE ARCFUNC "FUNC(A1, ""ARCTAN(A2,A3)"")";        1a34a38b

   _XX=ARCFUNC ;                                         1a34a38c
```

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

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

and finally: 1a34a40

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

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

7.35 "Name:Declaration 1a35

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

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

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