27
7-15-81
242
SAND80-7154/1
Unlimited Release
UC-66

B 5684 (1)

MASTER

Programmer's Manual for IOSYM: An Input-Oriented Simulation Language for Continuous Systems

Volume 1

David Michael Smith
Scientific Simulation Corp.

Sandia National Laboratories

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SAND80-7154/1
Unlimited Release
Printed

PROGRAMMERS' MANUAL FOR IOSYM:
AN INPUT-ORIENTED SIMULATION LANGUAGE FOR CONTINUOUS SYSTEMS
VOLUME I

950 9891
Inc. sec: PG 129

David Michael Smith
Senior Systems Analyst
Scientific Simulation Corporation
P.O. Box 9331
Albuquerque, New Mexico 87119

Formerly of
The BDM Corporation
1801 Rancolph, SE
Albuquerque, New Mexico 87106

ABSTRACT

IOSYM is an extension of the GASP IV simulation language. It permits systems which are sequences of continuous processes to be modeled graphically. Normally the system can be described by data input only. The language permits stochastic sequencing and termination criteria for processes and allows crossing conditions for ending operations that are more general than GASP IV. Extensive capability exists for conditional branching and logical modification of the network. IOSYM has been used to model the cost of geothermal drilling where the various costly processes of drilling are represented by IOSYM operations. The language is much more general, however, since it retains more of GASP IV's discrete event capabilities and permits easy modeling of continuous processes.

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

(BDM/TAC-79-673-TR)

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

ACKNOWLEDGEMENT

The author is indebted to Joseph Polito of Pritsker and Associates formerly of Sandia National Laboratories. His careful editing and aid in organizing and preparing figures for this document have substantially improved its final form.

In addition, Dr. Polito shared in the programing and program design tasks. Without his expertise and design, this work would never have begun.

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION	1
II. STRUCTURE OF IOSYM	3
A. IOSYM SEGMENTS	3
1. The IOSYM Segment	6
2. The PARSE Segment	6
3. The ZERO Segment	6
4. The CXDATN Segment	6
5. The PROCES Segment	6
6. The ARCSTR Segment	6
7. The COLSTR Segment	6
8. The DOSTR Segment	6
9. The SAVE Segment	6
10. The ANET Segment	6
11. The DDOSIM Segment	7
12. The BXDATN Segment	7
13. The GASP Segment	7
14. The PPRCOL Segment	7
15. The DATIN Segment	7
16. The OUTPUT Segment	7
17. The PPRTAB Segment	7
18. The PPRPLT Segment	7
19. The XPLOT Segment	7
20. The UPLT Segment	7
B. IOSYM USE OF COMMON STORAGE	8
C. IOSYM SEGMENTATION DIRECTIVES	8
D. IOSYM ERROR PROCESSING	8
III. DATA STRUCTURE AND PROGRAMMING CONSIDERATION	11
A. VARIABLE IDENTIFIERS	11
1. TABLE Reference with Variable Index	11
2. Independent Variables	12
3. Secondary Variables	12
4. Primary Variables	12
5. TABLE Element With Numeric Index	12
6. Derivative Variable	13
7. Substitutable Parameter	13
8. Constant	13
9. Operator	13
10. FORTRAN Library Functions	14

TABLE OF CONTENTS (Continued)

	<u>Page</u>
11. IOSYM Probability Functions	14
12. IOSYM Menu Functions	15
13. IOSYM User Functions	15
 B. IOSYM STORAGE OF ALPHANUMERIC INFORMATION	 15
C. COMMAND IDENTIFIERS (IICMs)	17
D. UNIFORM COMMAND SEQUENCES (USCs)	17
E. COMMAND LEVELS AND SYSTEM IDENTIFICATION	22
F. STORAGE ARRAYS FOR IOSYM INFORMATION	25
 1. ACTIVATE	 25
2. ARC	25
3. COLLECT	25
4. CROSS	25
5. DEACTIVATE	25
6. DO	25
7. DUMP	26
8. ECHO	26
9. EVENT	26
10. EXIT	26
11. FALSE	26
12. FOR	26
13. GET	26
14. GLOBAL	27
15. GO	27
16. GOTO	27
17. IF	27
18. INFO	27
19. JUMP	27
20. LABEL	27
21. LOOP	27
22. NETWORK	28
23. NOCOLLECT	28
24. NODE	28
25. NODUMP	28
26. NOECHO	28
27. NOFOR	28
28. NOINFO	29
29. NOPLOT	29
30. NOREPORT	29
31. NOTABULATE	29
32. NOTRACE	29
33. OPERATION	30
34. OPR	30

TABLE OF CONTENTS (Concluded)

	<u>Page</u>
35. PICK	30
36. PLOT	30
37. PRINT	30
38. PUT	31
39. REPLACE	31
40. REPORT	31
41. RESTORE	31
42. SAVE	31
43. SET or =	31
44. STAT	31
45. SWITCH	31
46. TABLE	32
47. TABULATE	32
48. TERM	32
49. TRACE	32
50. TRUE	32
51. TYPE	32
52. VARIABLE or VAR	33
53. WITH	33
G. HOW TO EXTEND IOSYM LIMITS	33
H. IOSYM FILE USAGE	34
I. IOSYM PLOT FILE FORMAT	35
J. IOSYM TABULATE FILE FORMAT	35
K. STORAGE OF ARITHMETIC EXPRESSIONS	36
L. EXAMPLE OF IOSYM BLANK COMMON STORAGE	38
M. CROSS REFERENCE BETWEEN ROUTINE NUMBER AND NAME	57
IV. DATA DESCRIPTION	62
V. SUBPROGRAM DESCRIPTIONS (CONTAINED IN VOLUME 2 OF PROGRAMMERS' MANUAL)	1

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	The Three Functional Processors of IOSYM	2
2	IOSYM Segments and the Segmentation Tree	3
3	IOSYM COMMON Areas	9
4	IOSYM Segmentation Directives	10
5	Example NODEs and ARCs	39
6	Blank COMMON (NSET) Example Contents	41
7	CCONS Array Storage Example	46

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	IOSYM SEGMENTS	4
2	INTERNAL REPRESENTATION OF ALL CHARACTERS RECOGNIZED BY IOSYM	16
3	IOSYM COMMANDS AND COMMAND IDENTIFIERS (IICMs)	18
4	UNIFORM COMMAND SEQUENCES (UCSs)	19
5	COMMANDS AND THEIR CORRESPONDING COMMAND LEVEL NUMBERS	22
6	DEFINITIONS OF SYNTAX CODE NUMBERS	24
7	ROUTINE CROSS-REFERENCES FROM NUMBER TO NAME	58
8	ROUTINE CROSS-REFERENCES FROM NAME TO NUMBER	60

I. INTRODUCTION

This programmers' manual is intended as an aid to the IOSYM program listing, IOSYM users' manual¹, the GASP-IV Simulation Language², and the Sandia Mathematical Program Library Manual³ (for line printer plot description of XPLOT, XPMNMX, and XPSYMB). This manual is for use by an individual knowledgeable with FORTRAN and the IOSYM users' manual. When used in conjunction with the computer program listing, it will allow IOSYM to be supported, modified, or extended. Due to the complexity of the program, these functions should not be attempted without first obtaining a detailed understanding of the program structure, data flow, and functional flow of IOSYM. The program is well documented with comments indicating the operations being performed.

IOSYM is an extension of the GASP-IV simulation language. It is general enough to be used as a tool for modeling systems that are characterized by discrete and/or continuous processes. IOSYM allows the model to be defined through free form data input rather than user-written programs. The user is not normally required to do any programming. Advanced users who are familiar with GASP will find that virtually all of GASP's capabilities are also available.

For a thorough understanding of how a simulation is performed by IOSYM, an understanding of the GASP-IV language is required. The GASP-IV text is a source of information on the structure, use, and description of GASP-IV.

IOSYM can be considered as three separate processors: Input, Network Summary, and Execution. Each of the processors consists of a group of subprograms. The processors are linked together by a small group of core subprograms as shown in Figure 1. The core group is labeled IOSYM.

1. Users' Manual for IOSYM: An Input-Oriented Simulation Language for Continuous Systems, Joseph Polito, SAND80-2000, 1980.

2. The GASP-IV Simulation Language, A. Alan B. Pritsker, John Wiley and Sons, Inc. 1974.

3. Sandia Mathematical Program Library Manual, Sandia Laboratories, SAND75-0545, 1975.

Figure 1. The Three Functional Processors of IOSYM

The Input processor is invoked whenever IOSYM commands are being processed. It reads the free form IOSYM commands, parses the card image into meaningful fields, and stores the data in the appropriate areas of blank and labeled COMMON.

The Network Summary processor is invoked only to print summaries of the existing "mind set." A summary of all of the Operations, Arcs, Nodes, Variables, Tables, and Statistics is printed.

Control is passed to the Execution processor when a GO command is encountered. This processor performs all of the simulation runs that have been requested by the user and produces all of the needed output.

Section II describes the structure of IOSYM in more detail. Program flow, the use at COMMON storage, error processing, and core memory minimization are discussed.

Section III presents data storage schemes and other programming considerations. The internal representation of IOSYM commands and variables is presented along with a detailed discussion of extending internal limits on the number of Arcs, Nodes, etc. A cross reference list of subprograms by name and number is given for error processing.

Section IV is an alphabetic list of all COMMON block variables. Definitions are given for each variable, and all subprograms that define or reference the variable are listed.

Section V is an alphabetic list of all subprograms in IOSYM except the GASP-IV programs. Each entry has a discussion of the program's purpose, its parameter list, routines that it calls, and routines called by it.

II. STRUCTURE OF IOSYM

A. IOSYM SEGMENTS

Figure 2 represents the functional segments in IOSYM and their relationships in terms of program control. Table 1 lists all of the programs and subprograms in each segment. Each of the segments will be discussed briefly below.

Figure 2. IOSYM Segments and the Segmentation Tree

TABLE 1. IOSYM SEGMENTS

<u>Segment</u>	<u>Programs and Subprograms</u>
IOSYM	IOSYM, CHOICE, USRDAT, ADATA, BLOCK DATA, ERROR, PPPRNT, EEP RNT, UNCODE, UNCO1, EEROR, EEOF, IIMOV, IISSET, SSHIFT, TTAPES, VVMOV, UEEROR, SETJ
PARSE	CHOIC2, PARSE, ALPFND, ALPSRH, ARAYCK, BLDALP, BLDDLM, BLDNUM, BLDOPR, CONVRT, CONVR2, DVARFD, DVFND, FNFND, FUNC1, FUNC2, FUNC3, FUNC4A, FUNC5, FUNFND, KWFND, LIBFND, LITFND, LOLFND, NAMFND, NUMFND, NUMSTR, RESEQ, SCAN, TBLFND, TRUTH, VARFND, FUNC4
CXDATN	XXDATN, IPACK, ERRIN, DFAUT, GPLOT, HISTO, PRNTS, UERR, IMAP, BUILD, COLCT, FILEM, PRNTQ, SET, SUMRY, TIMST, CXDATN
ZERO	ZERO, ZZSET, DFAUT
PROCES	PROCES
ARCSTR	ARCSTR, DVSTR, EVTSTR, FUNSTR, GTOSTR, IFSTR, JOESTR, LBLCHK, LBLSTR, LOPSTR, NODSTR, OPRSTR, PCKCK, PRTSTR, REPSTR, RESCC, SETSTR, SRHARC, TOFSTR, TYPSTR, TRMSTR, PCKSTR, PUTSTR, GETSTR, STASTR
COLSTR	COLSTR, FORSTR, GBLSTR, PLTSTR, RPTSTR, TBTSTR, FUNSTR, RESCC, TRCSTR, COLSTR
DOSTR	DOSTR, ENDNET, WTHSTR
SAVE	SAVE, RSAVE, WSAVE, NETSUM, VVARBL, GGVAR, VVWTMN, GG TAB

TABLE 1. IOSYM SEGMENTS (Concluded)

DDOSIM	DDOSIM, CCOLC, VVARBL, GGVAR, ZZSET, VVWTMN, FFPUDJ
BXDATN	XXDATN, IPACK, ERRIN, DFAULT, GPLOT, HISTO, PRNTS, UERR, IMAP, BUILD, COLCT, FILEM, PRNTQ, SET, SUMRY, TIMST, BXDATN
PPRCOL	PPRCOL, PPRRPT, SSTWRK
GASP	GASP, HISTO, TIMST, COLCT, UINTLC, USTAT, GPLOT
DATIN	DATIN, GAMA, RLOGN, TRIAG, WEIBL, DRAND, WWRT, CLEAR, EXPON, FILEM, PRNTQ, ERLNG, NPSSN, RNORN, UNFRM, EEVAL, AAIROP, AARCST, BBLIST, DDOCMD, DDOEVT, DDOLST, DDPRT, EEVLTF, EEXP, FF, GGOTO, GGPAR, GGTAB, IIACST, NNODST, NNXNOD, PP, PPFCMD, PPIDF, PPPAR, PPTAB, PPVAR, RREPLC, RREWRK, SSWARC, PRNTS, U, UEVENT, BETA, UERR, SUMRY, STATE, TIMSA, UTERM, KROSS, PPPUT, TTABU, INTLC, XXINTL, SET, WWTHMN, UUPRNT, EVNTS, MONTR, RMOVE, SCOND, SSAVE, NFIND, UMONT, UEVNTS, USCOND, USSAVE, USTATE, GGSTAT, GGET, GGSTEP, TTRACE, PACWJ, UREAD, FFRJ, FFAJ, UINTLC
OTPUT	OTPUT
PPRTAB	PPRTAB, UOTPUT, RREPRT, PPRTAB
PPRPLT	PPRPLT
XPLOT	XPLOT, XPMNMX, YPSYMB, ARANGE
UPLOT	UPLOT
ANET	NETSUM, VVARBL, VVWTMN, GGTAB, GGVAR, DDUMP, ANET, PVECJ, PIVECJ, PKMATJ

1. The IOSYM Segment

It performs initialization of variables that will not change during the job such as maximum array sizes and certain alphanumeric information. Also, the IOSYM segment directs program control to the Input, Network Summary, and Execution processors.

2. The PARSE Segment

It reads the cards and interprets them. It identifies the IOSYM command, delineates the various fields, and calls the PROCES segment to store the information in the appropriate arrays.

3. The ZERO Segment

It initializes variables that may change whenever a NETWORK command is encountered. These include the ARC, NODE, and variable storage arrays.

4. The CXDATN Segment

It is called from the PARSE segment whenever a "GASP" command is encountered when reading a new NETWORK. CXDATN contains the GASP-IV subprograms to perform GASP free form input.

5. The PROCES Segment

It passes control to the appropriate segments farther up the tree to store information currently being processed.

6. The ARCSTR Segment

It stores information from the multiple use commands such as ARC, NODE, GOTO, IF, JUMP, EXIT, etc.

7. The COLSTR Segment

It stores information from GLOBAL commands such as PLOT, REPORT, FOR, TRACE, etc.

8. The DOSTR Segment

It stores information from simulation directive commands such as DO, WITH, and END NETWORK.

9. The SAVE Segment

It processes SAVE and RESTORE commands.

10. The ANET Segment

It prints NETWORK summaries.

11. The DDOSIM Segment

It performs initialization of statistics and other variables in preparation to execute a GO command. It controls the number of simulations to be performed and executes the READ command. The READ command is performed locally in the DDOSIM segment and is the only IOSYM input command that is not processed by the Input Processor.

12. The BXDATN Segment

It is called only when GASP input is required between simulations (which is signaled by a "GASP" command between the DO and GO commands). BXDATN contains the same subprograms as the CXDATN segment.

13. The GASP Segment

It contains subprogram GASP and controls the simulations and processes the GASP histograms, plots, statistics, and collects.

14. The PPRCOL Segment

It prints COLLECT and REPORT statistics at the end of each simulation.

15. The DATIN Segment

It contains most GASP-IV subprograms and IOSYM subprograms that interact with subroutine GASP to perform the simulation runs.

16. The OUTPUT Segment

It controls program flow to print TABULATE statistics and PLOTS.

17. The PPRTAB Segment

It prints TABULATE statistics and processes statistics at the end of runs.

18. The PPRPLT Segment

It controls standard and user plotting.

19. The XPLOT Segment

It produces IOSYM standard line printer plots.

20. The UPLLOT Segment

If used it must be user written. It is used to produce more elegant plots on site-specific devices. Section III contains a description of the data format in the PLOT file. If subprogram UPLLOT is not defined by the user, a dummy subprogram is used. This dummy subprogram does nothing but return control to the calling subprogram.

B. IOSYM USE OF COMMON STORAGE

Labeled and blank COMMON storage is the main vehicle for passing information between subprograms. The labeled and blank COMMON areas used in IOSYM are shown in Figure 3.

Blank COMMON is used to store both the GASP-IV filing structure and the IOSYM NETWORK information. The first NNSET locations (NNSET is a GASP-IV variable) in blank COMMON are reserved for the GASP-IV file structure. Subsequent locations hold IOSYM OPERATIONS, ARCs, and NODEs. Labeled COMMON is used to store alphanumeric information (ARC names, etc.) and pointers to locate specific entities in blank COMMON. A complete description of the variables in COMMON is given in Section IV.

Blank COMMON and the temporary storage array NNWRK are used to hold both real and integer data. This is accomplished by EQUIVALENCEing the integer blank COMMON array NSET with the real array QSET and the integer work array NNWRK with the real array WWRK.

The Input processor stores information from IOSYM data into the COMMON block areas. Then the Execution and Network Summary processors operate on these areas.

C. IOSYM SEGMENTATION DIRECTIVES

In order to minimize core memory, IOSYM is segmented in the Sandia/Kirtland AFB implementations. Memory segmentation is done in the same way as is the functional segmentation shown in Figure 2. The segmentation directives for the CDC 7600/CYBER 76/CYBER 176 systems at Sandia and Kirtland AFB are shown in Figure 4. The system COMMON blocks "Q8.IO." and "FCL.C." must be declared GLOBAL in addition to the standard IOSYM COMMON areas.

D. IOSYM ERROR PROCESSING

IOSYM errors are identified by error numbers. A list of IOSYM error numbers with an explanation of each is contained in the users'

DIMENSION NSET(6000)	NDIM
DIMENSION WWRK(2000)	WDIM
COMMON/ALLJ/JPET,JRD,PREC,I72	ALLJ
COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,N	GCOM1
1NAPU,NNAPT,NHAIR,NNEIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNQW,ITBEG	GCOM1
2,ITCLR,ITFIN,ITRIB(25),ITSET	GCOM1
COMMON /GCOM2/ CD(100),DDL(100),DTFUL,DINOW,ISEES,LFLAG(50),NFLAG,	GCOM2
1NNEQD,NNEQS,NNEQT,SS(100),SSL(100),TTNEX	GCOM2
COMMON /GCOM3/ AAERR,DTMAX,DTMIN,DISAV,ITES,LLERR,LLSAV,LLSEV,RRE	GCOM3
1RR,ITLAS,ITSAV	GCOM3
COMMON /GCOM4/ DTPLT(10),HLOW(25),HHWID(25),IICRD,IITAP(10),JJCEL	GCOM4
1(500),LLABC(25,2),LLABH(25,2),LLABP(11,2),LLABT(25,2),LLPHI(10),LL	GCOM4
2PLO(10),LLPLT,LLSUP(15),LLSYM(10),MMPIS,NNCEL(25),NNCLT,NNHIS,NNPL	GCOM4
3T,NNPTS(10),NNSTA,NNVAR(10),PPhi(10),PPLO(10)	GCOM4
COMMON /GCOM5/ IIEVT,IISED(6),JJBEG,JJCLR,MMNIT,MMON,NNAME(3),NNCF	GCOM5
1I,NNDAY,NNPT,NNSET,NNPRJ,NNPRM,NNRNS,NNRUN,NNSTR,NNYR,SSEED(6)	GCOM5
COMMON /GCOM6/ EENQ(100),IINN(100),KKRNC(100),MMAXQ(100),QQIIM(100	GCOM6
1),SSUBV(25,5),SSTPV(25,6),VVNQ(100)	GCOM6
COMMON/FFI/KARD(80),IFLAG(50),IVALU(50),RVALU(50),IALPH(8,50),	FFI
1 NNFLD,MORE	FFI
COMMON /ERRELG/ IERRW,IERRF,NPRT2	ERRELG
COMMON/MISC/IECH,IIPUF,IIPUS,IISUM,IIPIR,JJFIL,IIFIN,NRTUT,JBEGG	MISC
COMMON/MCHDEP/ MAXNUM,MAXEXP	MCHDEP
COMMON/ME1/ NTSFMT(133),KYWORD(62,5),LIBFUN(16,2),ICHAR(50)	ME1
1 ,ICLEVL(4),LEVEL,NUINFO,NCARD,KARDD(90),FIRST,ECHO	ME1
1 ,NDXVAL,NDX1,DUMP	ME1
COMMON/IO1/ IIRPT(50),NNCULC(50),IIPS(5),NNPLOT(7, 5),	IO1
- NNTABU(9,10)	IO1
COMMON/IO2/ NNDUM,TIME1,IUPR(10,2),LLUCSL(62),NNPTGH(5),CCONS(30	IO2
- 0),SSIN(100),ODIN(100),NNPTAC(3,50),NNPTND(4,50),	IO2
- VVAR2(150,2),ITABL(1000,2),NNPT1(2,100),NNPT2(2,150),	IO2
- NNPT0(2),NNPTB(3,50),NNPTUP(6,50)	IO2
- ,LLARY(2,20),NNGDUM,IIFPT(4,20)	IO2
COMMON/IO3/ MMUPR,IISTAT,IIDOBJ,Iiop,IiARC,LLWRK,LLCRUS,LLEVNT,	IO3
- LLIST,LLTABU,LLRPT,LLCOL,LLWITH,IISIM,IIGASP,IIGO,	IO3
- IINXAC,IINXND,NNXTB,NNV1,NNV2,NNNODE,NNARC,NNCRUS,	IO3
- NNCOP,NNCOL,NNSIM,NNWITH,NNTBS,NNRPT,NNPLTS,VVOBJ,	IO3
- NNREAD,LLXSET,LLXWRK,NNCNS,NNOP,NNARY,NNFPT,NNMNET(3),	IO3
- LLTRAC,NNWRK(2000)	IO3
COMMON/IO4/ MMSET,MMWRK,MMCNS,MMV1,MMV2,MMTAB,MMXTH,	IO4
- MMTABU,MMRPT,MMCLOC,MMPLT,PPREC,MMPAR,MMNODE,	IO4
- MMOP,MMARC,MMFUNP,MMFUNF,MMARY,MMFPT,MMLINE	IO4
COMMON QSET(6000)	QSET
EQUIVALENCE(NSET(1),QSET(1))	EQN
EQUIVALENCE (NNWRK(1),WWRK(1))	EQW

Figure 3. IOSYM COMMON Areas

```

* IOSYM SEGLOAD DIRECTIVES
*
IOSYM INCLUDE CHOICE,USRDAT,ANATA,BLKDAT,,ERROR,PPRPT,EEPRNT,UNCODE,UNC
,01
*
* GLOBAL COMMON BLOCKS
*
IOSYM GLOBAL ALLJ,GCOM1,GCOM2,GCOM3,GCOM4,GCOM5,GCOM6
IOSYM GLOBAL ME1,MCHDEP,FFI,FRRFLG,MISC
IOSYM GLOBAL I01,I02,I03,I04
*
* SYSTEM COMMON BLOCKS THAT MUST BE GLOBAL
*
IOSYM GLOBAL Q8,I0,,FCL,C.
*
* IOSYM TREE STRUCTURE
*
CXDATN INCLUDE XXDATN,IPACK,ERRIN,DFAUT,GPLT,HISTO,PRNTS,UEPR,IMAP,BUIL
,D,COLCT,FILEM,PRNTQ,SET,SUMRY,TIMST
*
BXDATN INCLUDE XXDATN,IPACK,ERRIN,DFAUT,GPLT,HISTO,PRNTS,UEPR,IMAP,BUIL
,D,COLCT,FILEM,PRNTQ,SET,SUMRY,TIMST
*
ARCSTR INCLUDE DVSTR,EVTSTR,FUNSTR,GTOSTR,IFSTR,JOESTR,LALCHK,LALSTR,LOP
,STR,NODSTR,OPRSTR,PCKCK,PRTSTR,REPSTR,RESCC,SETSTR,SRHARC,TOFSTR,TYPSTR
,TRMSTR,PCKSTR,PUTSTR,GETSTR,STASTR
*
COLSTR INCLUDE FORSTR,GHLSTR,PLTSTR,RPTSTR,TBTSTR,FUNSTR,RESCC,TRCSTR
*
DOSTR INCLUDE ENDNET,WTHSTR
*
SAVE INCLUDE RSAVIN,WSAVIN,NETSUM,VVARBL,GGVAR,VVWTMN,GGTAH
*
PARSE INCLUDE CHOIC2
*
DDOSIM INCLUDE CCOLC,VVARBL,GGVAR,ZZSET,VVWTMN
*
PPRCOL INCLUDE PPRPT,SSTWRK
*
DATN INCLUDE GAMA,RLOGN,TRIAG,WFIHL,DRAND,WWRT,CLEAR,EXPON,FILEM,PRNTQ
,,ERLNG,GAM,NPSSN,RNOPM,UNFRM,EEVAL,AAIHOP,AARCST,BRLIST,DDOCMD,DDOEV,D
,DOI,ST,DDPRNT,EEVLTF,EEEXP,FF,GGOTO,GGPAR,GGTAH,IIACST,NNODST,NNXNOD,PP,P
,PFCMD,PPIDF,PPPAR,PPTAB,PPVAR,PPREPLC,RREWRK,SSWARC,PRNTS,U,UEVENT,USTAT
,,RFTA,UEPR,SUMRY,STATE,TIMSA,GPLT,UTERM,KROSS,PPRPT,TTARU,INTLC,XXINTL
,,SET,UITLTC,WWTMMN,UUPRNT,EVNTS,MONTM,MOVE,SCOND,SSAVE,NFIND,IMONT,UEV
,NTS,USCOND,USSAVE,USTATE,GGSTAT,GGGET,GGSTEP,TTRACE,PACWJ,URFAD,FFRJ,FF
,AJ
*
PPRTAB INCLUDE UOUPUT,PREPRT
*
XPLOT INCLUDE XPMNMX,XPSYMB,ARANGE
*
ZERO INCLUDE ZZSET,DFAUT
*
ANET INCLUDE VVARBL,VVWTMN,NETSUM,GGTAH,GGVAR,DDUMP
*
GASP INCLUDE HISTO,TIMST,COLCT
*
* TREE STRUCTURE
*
FTREE TREE PPRPT-(XPLOT,UPLT)
GTREE TREE OUPUT-(FTREE,PPRTAB)
DTREE TREE PROCES-(ARCSTR,COLSTR,DOSTR,SAVE)
RTREE TREE GASP-(GTREE,DATN)
ATREE TREE DDOSIM-(BXDATN,HTREE,PPRCOL)
CTREE TREE PARSE-(DTREE,ZERO,CXDATN)
MAIN TREE IOSYM-(CTREE,ANET,ATREE)
*
* END OF IOSYM SEGMENTATION
*
END IOSYM

```

Figure 4. IOSYM Segmentation Directives

manual. Odd error numbers indicate fatal errors and even numbers are warnings. Error messages are printed that indicate the error number and subprogram number in which the error occurred. A cross reference listing of subprogram numbers and names is given in Section III.

IOSYM subprogram EEROR processes all IOSYM errors and warnings. (Subprogram ERROR processes all GASP-IV errors.) Fatal errors that occur during execution of a simulation run will cause the job to be terminated. Fatal errors that occur during the reading of a new NETWORK will not cause immediate termination. IOSYM will attempt to finish reading the NETWORK before aborting the job. Warning messages may be suppressed by using the NOINFO command. See subprogram EEROR in section V for a description of the calling sequence.

III. DATA STRUCTURE AND PROGRAMMING CONSIDERATION

A. VARIABLE IDENTIFIERS

Variables, operations, functions, and some other entities are identified internally by a unique number called a Variable Identifier (IIDV). For example, the first primary variable has IIDV = 1001, the second has IIDV = 1002, etc. There are thirteen entities identified by IIDVs. The way in which the IIDVs is computed is described below.

1. TABLE Reference with Variable Index

If an IIDV is negative, it indicates that it is a TABLE reference with a variable index (e.g. GV(I)). The TABLE number (H) is found by taking the negative of the IIDV divided by ten. There are only six secondary variables that can be used as TABLE indices. They are I, J, K, L, M, and N. The index number is found by subtracting $10 \times H$ from the $\text{ABS}(\text{IIDV})$. If the result is one, then the variable used was I. If two is the result, then J was used, etc.

Given an IIDV that is negative the:

TABLE number H is: $\text{INT}(-\text{IIDV}/10)$

index i is: $-\text{IIDV} - 10 * H$

where i corresponds to one of the variables I, J, K, L, M, or N.

The value of the TABLE reference can be found by:

```
value1 = NNPTB(3,H)+i-1  
value = TTABL(value1)
```

2. Independent Variables

If an IIDV is zero, it indicates the independent variable is to be used. Its current value is the value of the GASP variable TNOW.

3. Secondary Variables

If an IIDV is greater than zero, but less than 1000, it indicates a secondary variable. Each secondary variable is identified by a number from 1 to 999. For instance, variable I is the first variable, M is fifth, and variable PI is eighth.

The current value of the secondary variable IIDV is found by:

```
value = VVAR2(IIDV, 1)
```

4. Primary Variables

If an IIDV is greater than 1000, but less than 2000, it indicates a primary variable. Each primary variable is identified by a number from 1 to 999. By subtracting 1000 from the IIDV, the primary variable can be identified. The current value of a primary variable can be found by:

```
value = SS(IIDV - 1000).
```

5. TABLE Element With Numeric Index

If an IIDV is greater than 2000, but less than 6000, it indicates a TABLE element with a numeric index. The TABLE element value identified is found by subtracting 2000 from the IIDV. This points to the location in the TTABL array where the value is stored. The current value of a TABLE element can be found by:

```
value = TTABL(IIDV - 2000, 1).
```

The IIDV for the sixth element in TABLE N is calculated by:

$$\text{IIDV} = 2000 + \text{NNPTB}(3, \text{N}) + 6 - 1.$$

6. Derivative Variable

If an IIDV is greater than 6000, but less than 7000, it indicates a derivative variable. By subtracting 6000 from the IIDV, the derivative variable can be identified. The current value of a derivative variable can be found by:

$$\text{value} = \text{DD}(\text{IIDV} - 6000)$$

7. Substitutable Parameter

If an IIDV is greater than 7000, but less than 8000, it indicates a substitutable parameter is to be used. The IIOPR array contains the substitutable parameters. The IIDV of the parameter that is being substituted can be found by:

$$\text{IIDV new} = \text{IIOPR}(\text{IIDV} - 7000, 1).$$

8. Constant

If an IIDV is greater than 8000, but less than 9000, it indicates a constant. Each constant is identified by a location in the CCONS array. By subtracting 8000 from the IIDV, the location in the CCONS array is identified. To find the value of the constant:

$$\text{value} = \text{CCONS}(\text{IIDV} - 8000).$$

9. Operator

If an IIDV is greater than 9000, but less than 9007, it indicates an operator is to be used. Each operator has its own IIDV.

<u>IIDV</u>	<u>OPERATOR</u>
9001	**
9002	*
9003	/
9004	+
9005	-
9006	=

10. FORTTRAN Library Functions

If an IIDV is greater than 9006, but less than 9023, it indicates a library function is to be used. Each function has its own IIDV.

<u>IIDV</u>	<u>LIBRARY FUNCTION</u>
9007	ATAN
9008	CMPLX (not currently implemented)
9009	AMAX1
9010	AMIN1
9011	AMOD
9012	SIGN
9013	ABS
9014	ATAN2
9015	COS
9016	EXP
9017	ALOG
9018	SIN
9019	SQRT
9020	TAN
9021	CONJG (not currently implemented)
9022	IFIX

11. IOSYM Probability Functions

If an IIDV is greater than 9022, but less than 9100, it indicates an IOSYM probability function is to be used. Each probability function is identified by a number. By subtracting 9100 from the IIDV, the negative of the function number is identified.

Probability function number = $-(\text{IIDV} - 9100)$.

Twelve probability functions are defined. (See users' manual for their descriptions.)

12. IOSYM Menu Functions

If an IIDV is greater than 9100, but less than 9100 plus MMFUNF, it indicates an IOSYM menu function is to be used. Each menu function is identified by a number. By subtracting 9100 from the IIDV, the function number is identified.

Menu function number = $\text{IIDV} - 9100$.

Twenty-nine menu functions are defined. (See users' manual for their descriptions.)

13. IOSYM User Functions

If an IIDV is greater than 9100 plus MMFUNF, but less than 10,000, it indicates an IOSYM User function is to be used. Each user function is identified by a number. By subtracting 9100 plus MMFUNF from the IIDV, the function number is identified.

User function number = $\text{IIDV} - 9100 - \text{MMFUNF}$.

The user must supply subroutine U to do the user function processing.

B. IOSYM STORAGE OF ALPHANUMERIC INFORMATION

Alpha fields read by IOSYM are stored in two words with the first word containing the first four characters of the alpha field, and the second word containing the next three characters plus a blank. If the alpha field isn't seven characters long, the alpha string is assumed to be filled with blanks to seven characters plus an extra trailing blank. There are two times when the alpha string can be longer than seven characters without generating an IOSYM error. If the first character in the string is a "D", then the string could be a derivative

variable, in which case the maximum number of characters in the alpha string is eight. The other case is when processing the first field on a card. The first field in all cases, except the assignment statements, is the command name. The command can be abbreviated to seven characters, or it can be the entire command. The characters in excess of seven will be ignored, and no error message printed.

The IOSYM value of the alpha field is calculated by taking the IOSYM representation of the first character, multiplying it by 64, and adding the value of the second character and then multiplying by 64, adding the third character, multiply by 64 and adding the fourth character. The result is stored in the first word. The second word is calculated in the same way for characters five through seven. Then a blank character is added. Table 2 shows the internal representation of all the IOSYM recognized characters.

TABLE 2. INTERNAL REPRESENTATION OF ALL CHARACTERS RECOGNIZED BY IOSYM

A = 1	Q = 17	6 = 33
B = 2	R = 18	7 = 34
C = 3	S = 19	8 = 35
D = 4	T = 20	9 = 36
E = 5	U = 21	. = 37
F = 6	V = 22	** = 38
G = 7	W = 23	* = 39
H = 8	X = 24	/ = 40
I = 9	Y = 25	+ = 41
J = 10	Z = 26	- = 42
K = 11	0 = 27	= = 43
L = 12	1 = 28	) = 44
M = 13	2 = 29	(= 45
N = 14	3 = 30	, = 46
O = 15	4 = 31	(blank) = 47
P = 16	5 = 32	\$ = 48

NOTE: If "***" occurs in consecutive fields, then it is a 38, to represent an exponential. If only one "*" occurs, then it is a 39, to represent a multiply.

For example, the alpha string "NETWORK" would be represented in two words as:

N = 14	O = 15
E = 5	R = 18
T = 20	K = 11
W = 23	blank = 47

word one = $((14 \times 64 + 5) \times 64 + 20) \times 64 + 23 = 3691799$

word two = $((15 \times 64 + 18) \times 64 + 11) \times 64 + 47 = 4006639$

Likewise, the alpha string "A" would be represented as

A = 1
blank = 47

word one = $((1 \times 64 + 47) \times 64 + 47) \times 64 + 47 = 457711$

word two = $((47 \times 64 + 47) \times 64 + 47) \times 64 + 47 = 12516335$

C. COMMAND IDENTIFIERS (IICMs)

Each IOSYM command is associated with a unique number called its Command Identifier (IICM). The commands and their associated IICMs are given in Table 3.

D. UNIFORM COMMAND SEQUENCES (USCs)

A UCS is a uniform method for representing a command in the link listed array, NSET. A UCS is contained in consecutive words in the NSET array as follows

The command parameters hold specific information for command IICM. IINX is a successor pointer. If IINX equals zero, no command follows. If a string of commands is being executed, NSET(IINX) is the location of the IICM of the next command. The format of the UCSs is given in Table 4.

TABLE 3. IOSYM COMMANDS AND COMMAND IDENTIFIERS (IICMs)

<u>IICM</u>	<u>COMMAND</u>	<u>IICM</u>	<u>COMMAND</u>
1.	ACTIVATE	32.	REPORT
2.	ARC	33.	RESTORE
3.	CHANGE	34.	SAVE
4.	CLEAR	35.	SET or =
5.	COLLECT	36.	SWITCH
6.	CONSTRAINT (not implemented)	37.	TABLE
7.	DEACTIVATE	38.	TABULATE
8.	DO	39.	TERM
9.	DUMP	40.	VARIABLE or VAR
10.	ECHO	41.	WITH
11.	EDIT (not implemented)	42.	EVENT
12.	END	43.	REPLACE
13.	GASP	44.	CROSS
14.	GLOBAL	45.	FOR
15.	GO	46.	LOOP
16.	GOTO	47.	LABEL
17.	TYPE	48.	IF
18.	MINIMIZE	49.	TRUE
19.	NETWORK	50.	FALSE
20.	NOCOLLECT	51.	QUIT
21.	NODE	52.	NOFOR
22.	NODUMP	53.	JUMP
23.	NOECHO	54.	EXIT
24.	NO PLOT	55.	PRINT
25.	NOREPORT	56.	PUT
26.	NOTABULATE	57.	GET
27.	OPERATION	58.	STAT
28.	OPR	59.	TRACE
29.	PICK	60.	NOTRACE
30.	PLOT	61.	INFO
31.	READ	62.	NOINFO

TABLE 4. UNIFORM COMMAND SEQUENCES (UCSSs)

- (1) SET
 - (a) IICM
 - (b) IIDV for variable to be set
 - (c) Pointer to expression location
 - (d) Truth function indicator
 - (e) IINX
- (2) ACTIVATE, DEACTIVATE, SWITCH
 - (a) IICM
 - (b) ARC number
 - (c) IINX
- (3) GOTO
 - (a) IICM
 - (b) ARC or NODE number $\left\{ \begin{array}{l} < 0 \text{ negative of NODE number} \\ > 0 \text{ ARC number} \end{array} \right.$
 - (c) IINX
- (4) PICK
 - (a) IICM
 - (b) PICK number
 - (c) Pointer to expression location
 - (d) Truth condition
 - (e) Pointer to next PICK for the appropriate NODE
- (5) TERM, CROSS
 - (a) IICM
 - (b) TERM or CROSS number
 - (c) Pointer to expression location
 - (d) CROSSing tolerance (if TERM command, it is equal to zero and isn't used).
 - (e) CROSSing direction (not used for TERM)
 - (f) IINX for commands on the TERM or CROSS
 - (g) IINX for next TERM or CROSS on current ARC
- (6) LABEL
 - (a) IICM
 - (b) First four characters of LABEL
 - (c) Next three characters of LABEL
 - (d) IINX
- (7) PRINT
 - (a) IICM
 - (b) TABULATE number
 - (c) NODE or ARC indicator

TABLE 4. UNIFORM COMMAND SEQUENCES (UCSs) (Continued)

- (d) IIDV of 1st variable to PRINT
- (e) IIDV of 2nd variable to PRINT
- (f) IIDV of 3rd variable to PRINT
- (g) IIDV of 4th variable to PRINT
- (h) IINX

- (8) PUT
 - (a) IICM
 - (b) IIDV of GASP file number
 - (c) IIDV of TABLE element
 - (d) Table number
 - (e) IINX

- (9) GET
 - (a) IICM
 - (b) IIDV of GASP file number
 - (c) IIDV of error indicator
 - (d) IIDV of TABLE element
 - (e) Table number
 - (f) IINX

- (10) STAT
 - (a) IICM
 - (b) Statistic type indicator
 - (c) Statistic number
 - (d) Pointer to expression location
 - (e) IIDV of T variable
 - (f) IINX

- (11) REPLACE
 - (a) IICM
 - (b) NODE number of NODE to REPLACE
 - (c) NODE number of REPLACEMENT NODE
 - (d) IINX

Seven Commands with Modified UCSs.

- (1) NODE
 - (a) Number of outgoing ARCs
 - (b) Pointer to location of first PICK
 - (c) Pointer to other commands on the NODE

- (2) EVENT
 - (a) IICM
 - (b) EVENT number
 - (c) Number of parameters
 - (d) List of IIDVs of parameters-as many words as specified in (c)
 - (e) IINX

TABLE 4. UNIFORM COMMAND SEQUENCES (UCSs) (Concluded)

- (3) LOOP
 - (a) IICM
 - (b) IIDV of LOOP index
 - (c) IIDV of LOOP initial index value
 - (d) IIDV of LOOP final index value
 - (e) IIDV of LOOP increment
 - (f) Pointer to first command of LOOP
 - (g) Pointer to LOOP LABEL location
- (4) OPR
 - (a) IICM
 - (b) OPERATION number
 - (c) Number of substitutable parameters
 - (d) List of IIDVs of substitutable parameters-as many words as specified in (c)
- (5) IF
 - (a) IICM
 - (b) IIDV for the variable on the left hand side of the truth condition
 - (c) Truth condition
 - (d) Pointer to the location of expression (RHS of truth condition)
 - (e) Pointer to the location of TRUE side
 - (f) Pointer to the location of FALSE side
 - (g) Pointer to the location of the LABEL
- (6) JUMP, EXIT
 - (a) IICM
 - (b) First four characters of LABEL name
 - (c) Next three characters of LABEL name
 - (d) Zero-to indicate end
- (7) ARC
 - (a) Start NODE
 - (b) End NODE
 - (c) Pointer to UCS for OPR
 - (d) Pointer to number of TERMS and CROSSes (see (1) below)
 - (e) Initial ARC status - ARC number implies inactive
+ ARC number implies active
 - (f) Current ARC status - number implies inactive
+ number implies active
 - (g) UCS for PICK (as many words as needed)
 - (h) UCS for OPR (as many words as needed)
 - (i) Pointer to UCS for type S commands
 - (j) Pointer to UCS for type I commands
 - (k) Pointer to UCS for type E commands
 - (l) Number of TERMS and CROSSes
 - (m) IINX to UCSs for first TERM or CROSS

E. COMMAND LEVELS AND SYSTEM IDENTIFICATION

To determine the syntax of a command, two arrays need to be discussed. The KYWORD array contains the command information.

KYWORD(I,1) contains the first four characters stored as an IOSYM alpha string.

KYWORD(I,2) contains the next three characters of the command.

KYWORD(I,3) contains the command level number. If this number is zero for a command, the command doesn't change the command level hierarchy. If the number is greater than zero, then 2 to the power of this number, minus one is the command level number to store in the command level hierarchy. Table 5 shows the commands that have command level numbers.

TABLE 5. COMMANDS AND THEIR CORRESPONDING COMMAND LEVEL NUMBERS

<u>COMMAND</u>	<u>COMMAND LEVEL NUMBER</u>
ARC	= 2
CHANGE	= 4
CONSTRAINT	= 8
DO	= 16
EDIT	= 32
GLOBAL	= 64
TYPE	= 128
MINIMIZE	= 256
NETWORK	= 512
OPERATION	= 1024
TERM OR CROSS	= 2048
NODE	= 4096

KYWORD(I,4) contains a number that represents on which command level numbers the command is valid. For instance, a SET command is valid on the following command levels:

NODE (4096), TERM OR CROSS (2048), TYPE (128), and CONSTRAINT (8).

The sum of the corresponding command level numbers is 6280. This is the value stored in the fourth word of array KEYWORD for command SET. When a command is processed, a check is done to determine whether the current command is valid under the current level.

KEYWORD(I,5) contains the pointer to the NTSFMT array where the syntax for the specific command is located. If the (I,5) value is zero, then that command has no syntax other than the command name.

The NTSFMT array contains the syntax for all the commands as coded numbers. The KEYWORD(I,5) value points to the location in the NTSFMT array where the number of elements in the syntax for that command is stored. This word in the NTSFMT array tells how many of the words following that word, indicate the syntax for the given command. For example the KEYWORD(I,5) value for command ARC is 92. In the NTSFMT array positions 90 to 101 are:

<u>(I)</u>	<u>NTSFMT(I)</u>
90	1
91	-12
92	4
93	-1
94	-2
95	-2
96	-16
97	4
98	-8
99	-16
100	-8
101	-12

In position 92 is a 4. This says that the next four words contain the syntax for the ARC command. Table 6 shows the syntax code numbers and their definitions.

TABLE 6. DEFINITIONS OF SYNTAX CODE NUMBERS

SYNTAX CODE NUMBER	DEFINITION
-1	ARC Name
-2	NODE Name
-3	OPERATION Name
-4	Process TABLE
-5	Process VARIABLE
-6	List of Numbers
-7	Defined Variable
-8	Number
-9	Function
-10	List of Variables
-12	List of Literals
-13	NODE or ARC Name
-14	Task Name
-15	Mind Set Name
-16	Alphanumeric String
-17	List of OPERATIONS
-18	LABEL

So the syntax for the ARC command is:

ARC, ARC Name, NODE Name, NODE Name, Alphanumeric string

According to the users' manual this is the serial syntax of the ARC command. The other commands are stored in a similar fashion.

During input processing, when an ARC command is encountered the first field after ARC will be analyzed to see if it is a valid ARC name. The second field is analyzed to see if it is a valid NODE name, as is the third field. The last field is analyzed to see if it is an alpha field. Once these fields have been identified, the appropriate information is

stored in Labeled and Blank COMMON areas. This is done in a similar fashion for all the IOSYM input cards.

F. STORAGE ARRAYS FOR IOSYM INFORMATION

This section explains how and where the IOSYM information is stored. For detailed descriptions of the ARRAYS and VARIABLES discussed in this section, see DATA DESCRIPTION in section IV. All "positions" refer to positions in the NSET/QSET array.

1. ACTIVATE

The pointer to the UCS of this command is the command that preceeds the ACTIVATE command. The pointer points to the location in the NSET array where the ACTIVATE UCS begins.

2. ARC

The ARC name and the location, in the NSET array where the ARC UCS is stored, is in the NNPTAC array.

3. COLLECT

The variables to be COLLECTed are stored in the NNCOLC array. The variables are stored as variable identifiers, (described in Section III-A). Variable NNCOL is set to the number of variables to be COLLECTed.

4. CROSS

The pointer to the CROSS command is stored in the UCS for the ARC command.

5. DEACTIVATE

The pointer to the UCS of this command is stored in the UCS for the command that preceeds the DEACTIVATE command.

6. DO

The number of simulations specified on the DO command is stored in the variable NNSIM.

7. DUMP

If a DUMP command is specified, then the logical variable DUMP is set to TRUE.

8. ECHO

If an ECHO command is specified, then the logical variable ECHO is set to TRUE.

9. EVENT

The pointer to the UCS of this command is stored in the UCS for the command that preceeds the EVENT command.

10. EXIT

The pointer to the UCS of this command is stored in the UCS for the command that preceeds the EXIT command. The pointer points to the location in the NSET array where the EVENT UCS begins.

11. FALSE

If a FALSE command is encountered after processing an IF command, the appropriate element of the UCS of the IF command is set to point to the next command after the FALSE. This is where processing is to continue if the IF expression is FALSE.

12. FOR

The OPERATIONS specified on the FOR command, are stored in the NNPTOP array. Variable NNCOP is set to the number of OPERATIONS to COLLECT on.

13. GET

The pointer to the UCS of this command is stored in the UCS of the command that preceeds the GET command.

14. GLOBAL

If the GLOBAL command is specified with a list of substitutable parameters, the substitutable parameters are stored in the IIOPR array. The number of substitutable parameters is stored in the variable NNGDUM.

15. GO

The number of the NODE specified on the GO command is stored in the variable IIGO.

16. GOTO

The pointer to the UCS of this command is stored in the UCS of the preceeding command.

17. IF

The pointer to the UCS of this command is stored in the UCS for the command that preceeds the IF command.

18. INFO

If the INFO command is specified, then the logical variable NOINFO is set to FALSE.

19. JUMP

The pointer to the UCS of this command is stored in the UCS for the command that preceeds the JUMP command.

20. LABEL

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the LABEL command.

21. LOOP

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the LOOP command.

22. NETWORK

The NETWORK name is stored in the NNMNET array.

23. NOCOLLECT

If there are no parameters specified on the NOCOLLECT card, then the variable NNCOL is set to zero to indicate there are no COLLECT variables, and NNCOP is set to zero.

If there are parameters on the NOCOLLECT card, then these variables are deleted from the NNCOLC array, and the variable NNCOL is decreased by one for each variable eliminated.

24. NODE

The NODE name and location in the NSET array where the other NODE informaton is stored is stored in the NNPTND array.

25. NODUMP

If a NODUMP command is specified, then the logical variable DUMP is set to FALSE.

26. NOECHO

If a NOECHO command is specified, then the logical variable ECHO is set to FALSE.

27. NOFOR

If there are no parameters specified on the NOFOR command, then the variable NNCOP is set to zero, and the array NNPTOP is set so that none of the OPERATIONS are COLLECTed on.

If there are OPERATIONS on the NOFOR card, then the NNPTOP array is set so they are not COLLECTed on. Variable NNCOP is also decreased by one for each OPERATION deleted.

28. NOINFO

If the NOINFO command is specified, then the logical variable NOINFO is set to TRUE.

29. NO PLOT

The array NNPLT is reset so the specified plot is made inactive. Variable NNPLTS is decreased by one for each of the PLOTS made inactive.

30. NOREPORT

If no parameters are specified on the NOREPORT command, then all the variables stored in the IIRPT array are deleted. The variable NNRPT is set to zero to indicate that there are no REPORT variables.

If there are parameters on the NOREPORT command, then the variables specified are deleted from the IIRPT array, and the variable NNRPT is decreased by one for each variable deleted.

31. NOTABULATE

The TABULATE number specified on the NOTABULATE command is reset to -1 in the NNTABU array.

32. NOTRACE

If no parameters are specified on the NOTRACE command, then the variable LLTRAC is set to the negative of the pointer to the NSET array where the TRACE information is stored. The number of TRACE variables is set to zero.

If parameters are specified on the NOTRACE command, those variables are deleted from the TRACE information, and the number of variables to TRACE is decreased by one for each variable deleted. Variable LLTRAC points to the location in the NSET array where the TRACE information is stored. These five locations contain:

- (1) Number of TRACE variables
- (2) IIDV of the 1st variable
- (3) IIDV of the 2nd variable
- (4) IIDV of the 3rd variable
- (5) IIDV of the 4th variable

33. OPERATION

The number of the current OPERATION is stored in IIOP, which is used as a pointer to the NNPTOP array.

34. OPR

The location of the start of the UCS for the OPR command is stored in the UCS for the ARC the OPR command is being processed on. The pointer from the ARC UCS points to the location in the NSET array where the OPR command UCS is stored.

35. PICK

If the current ARC is the first ARC encountered which leaves the NODE, the pointer to this PICK command is stored in the location of the NODE UCS. Subsequent ARCs which leave the NODE have their PICK UCS's pointed to by the (e) element of the previous PICK. When the END NETWORK is encountered, the chain of PICK commands is re-ordered in order of PICK number - lowest to highest.

36. PLOT

Array NNPLOT is used to determine if a given PLOT is active or inactive and is used to store all the PLOT variable IIDVs. Variable NNPLTS is used to store the number of PLOTS that are active.

37. PRINT

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the PRINT command.

38. PUT

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the PUT command.

39. REPLACE

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the REPLACE command.

40. REPORT

The variables specified on the REPORT command are stored in the IIRPT array as IIDVs. Variable NNRPT contains the number of variables that are stored in the IIRPT array.

41. RESTORE

The NETWORK name is stored in the NNMNET array. The print indicator is also stored in the NNMNET array.

42. SAVE

The NETWORK name is stored in the NNMNET array. The print indicator is also stored in the NNMNET array.

43. SET or =

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the SET command.

44. STAT

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the STAT command.

45. SWITCH

The pointer to the UCS for this command is stored in the UCS for the command that preceeds the SWITCH command.

46. TABLE

The TABLE names are stored in the NNPTB array. The TABLE values are stored in the TTABL array. The variable LLTAB indicates the last position in the TTABL array that has been used. The third element in the NNPTB array points to the location in the TTABL array where the appropriate TABLE values begin.

47. TABULATE

The TABULATE time option and the variable identifiers are stored in the NNTABU array.

48. TERM

The pointer to the UCS for the TERM command is stored in the UCS for the associated ARC.

49. TRACE

Variable LLTRAC points to the location in the NSET array where the TRACE information is stored. Five words are used to specify TRACE information as shown below:

- (1) Number of TRACE variables
- (2) IIDV of 1st variable
- (3) IIDV of 2nd variable
- (4) IIDV of 3rd variable
- (5) IIDV of 4th variable

50. TRUE

If a TRUE command is processed after encountering an IF command, the (e) element of the UCS of the IF command is changed to point to the the next command after the TRUE. This is the point at which processing is to continue if the IF expression is TRUE.

51. TYPE

If processing an ARC command, then the ARC UCS is updated to show where the specified TYPE commands start in the NSET array. If

processing a GLOBAL command, then the NNPTGB array is updated to show where the TYPE commands start. If processing an OPERATION command then the NNPTOP array is updated to indicate where the TYPE commands start.

52. VARIABLE or VAR

If processing the independent variable, then the variable name is stored in the NNPT0 array, and the initial value is stored in variable TTIME1. If processing a primary variable, then the name is stored in the NNPT1 array; the initial value is stored in the SSIN array, and the variable NNV1 is increased by one to show there is one more primary variable. If processing a secondary variable, then the name is stored in the NNPT2 array, the initial value is stored in VVAR2 array, and the variable NNV2 is increased by one to show there is one more secondary variable.

53. WITH

The WITH variable IIDVs are stored in the NNWRK array starting at the location specified by the variable LLWITH. The variable NNWITH is the number of WITH variables.

G. HOW TO EXTEND ISOYM LIMITS

For detailed descriptions of the ARRAYS and VARIABLES discussed in this section see DATA DESCRIPTION in section IV.

Subroutine USRDAT contains the variables necessary to change the dimensions of IOSYM arrays. The variables are: MMSET, MMOPR, MMWRK, MMCNS, MMV2, MMTAB, MMXTB, MMTABU, MMRPT, MMCOLC, MMPLT, MMNODE, MMOP, MMARC, MMFPT, and MMARY. The dimensions of the appropriate arrays should agree with the above variables. The arrays are: NSET, IIOPR, NNWRK/WWRK, CCONS, VVAR2, TTABL, NNPTB, NNTABU, IIRPT, NNCOLC, NNPLT, NNPTND, NNPTOP, NNPTAC, IIFPT, and LLARY, respectively.

Changes in the above arrays allow the user to change the amount of NETWORK storage, number of substitutable parameters, amount of working

storage, number of constants, number of secondary variables, amount of TABLE storage, number of TABLEs, number of TABULATEs, number of REPORT variables, number of COLLECT variables, number of PLOTs, number of NODEs, number of OPERATIONs, number of ARCs, number of nested IFs and LOOPs, and number of LABELs.

Variable MMLINE sets the maximum number of lines on each computer page. Variable PPREC sets the computer machine precision. Variable MAXNUM sets the maximum number of digits in a number. Variable MAXEXP sets the maximum exponent.

H. ISOYM FILE USAGE

There are eight files used by IOSYM. They are TAPE5, TAPE6, TAPE8, TAPE9, TAPE10, TAPE11, TAPE12, and TAPE98. All files are sequential files.

TAPE5 is the input file.

TAPE6 is the output file.

TAPE8 contains the information that is used to RESTORE a NETWORK or to SAVE a NETWORK. If SAVE and RESTORE are used TAPE8 should be stored on magnetic tapes or permanent files between jobs.

TAPE9 contains GASP storage information. It is used to restore the GASP information when starting each simulation. TAPE9 is a working file and must not be retained between jobs.

TAPE10 contains the IOSYM PLOT information. It is a working file.

TAPE11 contains the IOSYM TABULATE information, and it is a working file.

TAPE12 is a working file used when processing a SAVE command.

TAPE98 contains the PLOT output generated by the user-written subroutine UPLLOT. This file is for use by a post-processor to generate specialized plots.

I. IOSYM PLOT FILE FORMAT

TAPE10 is used to store the PLOT information to be processed at the end of each simulation. The PLOT file has the following format.

Each record on the PLOT file contains a variable number of items, these items are:

INUM - PLOT number.

T - Current value of the independent variable.

NV - Number of independent variables.

VAL(K), K=1, NV - Value of all NV variables.

If INUM is equal to -191919 and T is equal to -191919., then the end of the file is indicated.

J. IOSYM TABULATE FILE FORMAT

TAPE11 is used to store the TABULATE and PRINT information to be processed for TABULATE statistics. The TABULATE file has the following format.

There are two records in each group. The first record contains two items:

IA - PRINT or TABULATE indicator. If IA is equal to 999, the next record contains PRINT information. If IA is any other value, the next record contains TABULATE information.

INUM - Contains the TABULATE number on which the PRINT or TABULATE is to occur.

If IA is equal to -191919 and INUM is equal to -191919, the end of the file is indicated.

If the next record is TABULATE information, it contains a variable number of items:

INUM - TABULATE number.

T - Simulation time when this record was written.

NV - Number of TABULATE variables.
 VAL(J),J=1,NV - Value of all NV variables.
 IOPNO1 - OPERATION number of the OPERATION on which the TABULATE was collected.
 IOPNO2 - ARC number of the ARC on which the TABULATE was collected.

If the next record is PRINT information, it contains a variable number of items:

INUM - TABULATE number on which the PRINT is to occur.
 T - Simulation time when this record was written.
 NV - Number of PRINT variables.
 VAL(J),J=1,NV - Value of all NV variables.
 IOPNO1 - If the PRINT was processed on an OPERATION, this contains the OPERATION number.
 IOPNO2 - If IOPNO2 is greater than zero, then IOPNO2 is the ARC number of the ARC on which the PRINT was performed.
 If IOPNO2 is less than zero, then the negative of IOPNO2 is the NODE number of the NODE on which the PRINT was performed.
 IVAL(J),J=1,NV - Contains the IIDVs of all NV variables.

K. STORAGE OF ARITHMETIC EXPRESSIONS

Each expression is stored in the NSET array. The expressions are stored in forward polish notation. They are stored so that when an operator is found, the last two operands encountered are processed by the operator. When a FORTRAN function or an IOSYM function is encountered, the function number is identified, the arguments are calculated, and the result is used as an operand for the next operation. Two examples of IOSYM expressions are presented here:

CASE 1 - RIGRATE + F29 (5, 8, 1/GV)

	<u>CODE</u>	<u>NOTE</u>
1.	14	
2.	10	RIGRATE
3.	10003	
4.	3	
5.	8003	5
6.	8004	8
7.	8005	1
8.	9129	F29
9.	1	
10.	1	GV
11.	0	
12.	0	
13.	0	
14.	9004	+
15.	9006	=

The first item says that there are fourteen words in this expression. Item 2 is the IIDV of RIGRATE. Item 3 is an end-of-function indicator. Item 4 indicates that there are three items in the first argument in the function. Items 5, 6, and 7 are the IIDVs of the argument and represent the number 5, 8, and 1, respectively. Item 8 is the function indicator, in this case menu function 29. Item 9 indicates that there is one TABLE in the second argument, and item 10 is the number of the GV table. Items 11, 12, and 13 indicate that there are no parameter sets, accessible entities, or TRUTH condition. Item 14 says to add the last two operands. The first operand is the value of the function calculation. The second operand is the variable RIGRATE. Item 15 says to store the result of the expression in the appropriate location.

CASE 2 - GUAGE + GV(47)*(CBP - TREAM(K))

	<u>CODE</u>	<u>NOTE</u>
1.	8	
2.	48	GUAGE
3.	2047	GV(47)
4.	1011	CBP
5.	-223	TREAM(K)
6.	9005	-
7.	9002	*
8.	9004	+
9.	9006	=

Item 1 indicates that there are eight words in this expression. Item 2 is the IIDV of GUAGE; Item 3 is the IIDV of TABLE element GV(47). Item 4 is the IIDV of CBP, and Item 5 is the TABLE element TREAM(K). Item 6 is a negative operator, so the last two operands, TREAM(K) and CBP, are subtracted. Item 7 indicates a multiplication of the last two operands, last result and GV(47). Item 8 indicates an addition of the last two operands, last result and GUAGE. Item 9 indicates that the last result should be stored in the appropriate location. It should be noted that in this example the equation was reordered to do the calculations in the proper sequence, taking into account the parenthesis and operators.

L. EXAMPLE OF IOSYM BLANK COMMON STORAGE

Figure 5 shows a number of NODEs and ARCs taken from an IOSYM NETWORK. Figure 6 shows the corresponding locations in the BLANK COMMON array NSET where the NODE and ARC information is stored. ARC A is stored starting in location 2189. NODE 1 is stored starting in location 1759, NODE 2 in location 1587. Figure 7 shows the CCONS array where all the constants are stored (the 8XXX codes).


```

$
NODE 2
  DEACTIVATE M
  KODE1=IFIX(KODE/100+0.001)
  KODE2=KODE-KODE1*100
  IF 21 KODE GE 0
    TRUE
    IF 22 CBP GT 0
      TRUE
      ACTIVATE M
      GOTC A M
      LABEL 22
    FALSE
    KODE1=-KODE1
    KODE2=-KODE2
    LABEL 21
  J=9
  IF 23 J EQ ABS(KODE1)
    TRUE
    J=1
    IF 23 J LE FI(CBP////LE)+FI(KODE ////LT)
      TRUE
      GOTO N -
      VV(1)=UR(KODE2,DEPTH,COST///INTVCST,INTVTIM)
    LABEL 23
  END
$
NODE 1
  END
$

```

Figure 5. Example NODEs and ARCs

```

$
ARC A 1 2
  OPR DRLING
  CROSS 1 GE 1 DEPTH-WELL(I)
  KODE=KODES(I)
  END
  CROSS 2 GE 5 ARCLN-NXSURV
  GOTO N 3
  END
  TERM 3 PULL
  GOTO N 4
  IF T31 DEPTH GE WELL(I)
  TRUE
    GOTO N 2
    KODE=KODES(I)
    LABEL T31
  END
  CROSS 4 GE 1 CBP-NXCON
  GOTO N 13
  IF C41 PULL LT TIME+CONCTIM
  TRUE
    GOTO N 4
  FALSE
    PULL=PULL+CONCTIM
    LABEL C41
  END
  CROSS 5 GE 5 DEPTH-FDEPTH-.01
  GOTO N 16
  M=3
  END
  CROSS 6 GE 5 CBP-OUTGAUG-.01
  GOTO N 16
  M=4
  CBP=OUTGAUG
  END
  CROSS 7 GE 5 CBP-GAUGCBP-.01
  GOTO N 4
  CBP=GAUGCBP
  END
END ARC

```

Figure 5. Example NODEs and ARCs (Concluded)

I	NSET (U)		I	NSET (U)	
1587=	10	(a)	1634=	3	(f) FALSE
1588=	NODE 2	2414	(b)	1635=	1645
1589=	1590	(c)	1636=	2	(g) LABEL
1590=	7	(a)	1637=	Expression	8124
1591=	DEACTIVATE	1	(b)	1638=	9035
1592=	1593	(c)	1639=	START OF TRUE	1
1593=	35	(a)	1640=	ACTIVATE	1
1594=	41	(b) CODE 1	1641=	1642	(c)
1595=	SET	1598	(c)	1642=	16
1596=	0	(d)	1643=	GO TO	1
1597=	1607	(e)	1644=	J	(c)
1598=	8		1645=	47	(a)
1599=	10003		1646=	LABEL	7724013
1600=	40	CODE	1647=	22	12516335
1601=	8120	CCONS (120)	1648=	J	(d)
1602=	Expression	9005	/	1649=	START OF FALSE
1603=	3121	CCONS (121)	1650=	41	(b) CODE 1
1604=	9004	+	1651=	SET	1654
1605=	9022	IFIX	1652=	0	(d)
1606=	9005	=	1653=	1659	(e)
1607=	35	(a)	1654=	4	
1608=	42	(b) CODE 2	1655=	8125	CCONS (125)
1609=	SET	1612	(c)	1656=	Expression
1610=	0	(d)	1657=	9005	-
1611=	1619	(e)	1658=	9006	=
1612=	6		1659=	35	(a)
1613=	40	CODE	1660=	42	(b) CODE 2
1614=	41	CODE 1	1661=	SET	1664
1615=	Expression	8122	CCONS (122)	1662=	0
1616=	9002	*	1663=	0	(c)
1617=	9005	-	1664=	4	
1618=	9006	=	1665=	8126	CCONS (126)
1619=	48	(a)	1666=	Expression	42
1620=	40	(b) CODE	1667=	9005	-
1621=	5	(c) GT	1668=	9006	=
1622=	IF 21	1626	(d) Expression	1669=	47
1623=	1629	(e) TRUE	1670=	LABEL 21	7719919
1624=	1649	(f) FALSE	1671=	12516335	(c)
1625=	1659	(g) LABEL	1672=	1673	(d)
1626=	2		1673=	35	(a)
1627=	Expression	8123	CCONS (123)	1674=	2
1628=	9006	=	1675=	SET	1678
1629=	START OF TRUE	48	(a)	1676=	0
1630=	1011	(b) CBP	1677=	1631	(e)
1631=	4	(c) GT	1678=	2	
1632=	IF 22	1635	(d) Expression	1679=	Expression
1633=	1639	(e) TRUE	1680=	9006	=
			1681=	IF 23	48
			1682=	2	(b) J

Figure 6. Blank COMMON (NSET) Example Contents

I		NSET (I)		I		NSET (I)	
1683=		3	(c) EQ	1723=		1	
1684=		1688	(d) Expression	1724=		0	
1685=	IF 23	1693	(a) TRUE	1725=		1	
1686=		0	(f) FALSE	1726=	Expression	0	
1687=		1755	(g) LABEL	1727=		0	
1688=		4		1728=		2	LT
1689=		10003		1729=		9004	+
1690=	Expression	41	KODE 1	1730=		9006	=
1691=		9013	ABS	1731=	START OF TRUE	16	(a)
1692=		9006	=	1732=	GO TO	-3	(b)
1693=	START OF TRUE	35	(a)	1733=		1734	(c)
1694=		2	(b) J	1734=		35	(a)
1695=	SET	1698	(c)	1735=		2051	(b) NV (I)
1696=		0	(d)	1736=	SET	1739	(c)
1697=		1701	(e)	1737=		0	(d)
1698=		2		1738=		0	(e)
1699=	Expression	8128	CCONS (128)	1739=		15	
1700=		9006	=	1740=		10003	
1701=		48	(a)	1741=		3	
1702=		2	(b) J	1742=		42	KODE 2
1703=		1	(c) LE	1743=		1006	DEPTH
1704=	IF 23	1708	(d) Expression	1744=		1001	COST
1705=		1731	(a) TRUE	1745=		9131	U2
1706=		0	(f) FALSE	1746=	Expression	1	
1707=		1755	(g) LABEL	1747=		0	
1708=		22		1748=		1	
1709=		10003		1749=		0	
1710=		1		1750=		2	
1711=		1011	CBP	1751=		54	INTVCST
1712=		9101	FI	1752=		55	INTVTIM
1713=		1		1753=		0	
1714=		0		1754=		9006	=
1715=	Expression	1		1755=		47	
1716=		0		1756=	LABEL 23	7728111	
1717=		0		1757=		12516335	
1718=		1	LE	1758=		0	
1719=		10003		1759=		1	(a)
1720=		1		1760=	NODE 1	2195	(b)
1721=		40	KODE	1761=		0	(c)
1722=		9101	FI				

Figure 6. Blank COMMON (NSET) Example Contents (Continued)

I	NSET (I)		I	NSET (I)	
2189=	4	(a) START NODE	2238=	9005	-
2190=	2	(b) STOP NODE	2239=	9006	-
2191=	2200	(c) OPR	2240=	16	
2192=	2206	(d)	2241=	-5	GO TO
2193=	2	(e) INITIAL STATUS	2242=	0	
2194=	1	(f) CURRENT STATUS	2243=	39	(a)
2195=	29	(g) (a)	2244=	3	(b)
2196=	0	(b)	2245=	2250	(c)
2197=	-1	(c)	2246=	0	(d) TERM 3
2198=	0	(d)	2247=	0	(e)
2199=	0	(e)	2248=	2253	(f)
2200=	28	(h) (a)	2249=	2281	(g)
2201=	1	(b)	2250=	2	
2202=	0	(c)	2251=	36	PULL
2203=	-1	(i)	2252=	9006	-
2204=	-1	(j)	2253=	16	(a)
2205=	-1	(h)	2254=	-6	(b) GO TO
2206=	7	(i)	2255=	2256	(c)
2207=	2208	(m)	2256=	48	(a)
2208=	44	(a)	2257=	1006	(b) DEPTH
2209=	1	(b)	2258=	5	(c) GE
2210=	2215	(c)	2259=	2263	(d) IF T3I Expression
2211=	*****	(d)	2260=	2266	(e) TRUE
2212=	5	(e) GE	2261=	0	(f) FALSE
2213=	2220	(f)	2262=	2277	(g) LABEL
2214=	2228	(g)	2263=	2	
2215=	4		2264=	-51	WELL (I) Expression
2216=	1006	DEPTH	2265=	9006	-
2217=	-51	WELL (I)	2266=	16	START OF TRUE
2218=	9005	-	2267=	-2	GO TO
2219=	9006	-	2268=	2269	
2220=	35	(a)	2269=	35	(a)
2221=	40	(b) CODE	2270=	40	(b) CODE
2222=	2225	(c)	2271=	2274	(c) SET
2223=	0	(d)	2272=	0	(d)
2224=	0	(e)	2273=	0	(e)
2225=	2		2274=	2	
2226=	-61	KODES (I) Expression	2275=	-61	KODES (I)
2227=	9006	-	2276=	9006	-
2228=	44	(a)	2277=	47	(a)
2229=	2	(b)	2278=	5367599	(b) LABEL T3I
2230=	2235	(c)	2279=	12516335	(c)
2231=	*****	(d)	2280=	0	(d)
2232=	5	(e) GE	2281=	44	(a)
2233=	2240	(f)	2282=	4	(b)
2234=	2243	(g)	2283=	2288	(c) CROSS 4
2235=	4		2284=	*****	(d)
2236=	1005	ARCLEN	2285=	5	(e) GE
2237=	43	NXSURV	2286=	2293	(f)

Figure 6. Blank COMMON (NSET) Example Contents (Continued)

I	NSET (I)	I	NSET (I)
2287= CROSS 4	2325 (g)	2335=	9005 -
2288=	4	2336=	8154 CCONS (154)
2289=	1011 CBP	2337= Expression	9005 -
2290= Expression	16 NXCON	2338=	9006 =
2291=	9005 -	2339=	16 (a)
2292=	9006 =	2340= GO TO	-15 (b)
2293=	16 (a)	2341=	2342 (c)
2294= GO TO	-13 (b)	2342=	35 (a)
2295=	2296 (c)	2343=	5 (b) M
2296=	48 (a)	2344= SET	2347 (c)
2297=	36 (b) PULL	2345=	0 (d)
2298=	2 (c) LT	2346=	0 (e)
2299= IF C41	2303 (d) Expression	2347=	2
2300=	2308 (e) TRUE	2348= Expression	8155 CCONS (155)
2301=	2311 (f) FALSE	2349=	9006 =
2302=	2321 (g) LABEL	2350=	44 (a)
2303=	4	2351=	6 (b)
2304=	0 TIME	2352= CROSS 6	2357 (c)
2305= Expression	17 CONCTIM	2353= *****	(d)
2306=	9004 +	2354=	5 (e) GE
2307=	9006 =	2355=	2364 (f)
2308= START OF TRUE	16 (a)	2356=	2383 (g)
2309= GO TO	-6 (b)	2357=	6
2310=	0 (c)	2358=	1011 CBP
2311= START OF FALSE	35 (a)	2359=	49 OUTGUAG
2312=	36 (b) PULL	2360= Expression	9005 -
2313= SET	2316 (c)	2361=	8156 CCONS (156)
2314=	0 (d)	2362=	9005 -
2315=	0 (e)	2363=	9006 =
2316=	4	2364=	16 (a)
2317=	36 PULL	2365= GO TO	-15 (b)
2318= Expression	17 CONCTIM	2366=	2367 (c)
2319=	9004 +	2367=	35 (a)
2320=	9006 =	2368=	5 (b) M
2321=	47 (a)	2369= SET	2372 (c)
2322=	915247 (b)	2370=	0 (d)
2323= LABEL C41 12515335	(c)	2371=	2375 (e)
2324=	0 (d)	2372=	2
2325=	44 (a)	2373= Expression	8157 CCONS (157)
2326=	5 (b)	2374=	9006 =
2327= CROSS 5	2332 (c)	2375=	35 (a)
2328= *****	(d)	2376=	1011 (b) CBP
2329=	5 (e) GE	2377= SET	2380 (c)
2330=	2339 (f)	2378=	0 (d)
2331=	2350 (g)	2379=	0 (e)
2332=	6	2380=	2
2333= Expression	1006 DEPTH	2381= Expression	49 OUTGUAG
2334=	51 FDEPTH	2382=	9006 =

Figure 6. Blank COMMON (NSET) Example Contents (Continued)

<u>I</u>	<u>NSET (I)</u>	
<u>2383=</u>	44	(a)
2384=	7	(b)
2385= CROSS 7	2390	(c)
2386=*****		(d)
2387=	5	(e) GE
2388=	2397	(f)
<u>2389=</u>	0	(g)
<u>2390=</u>	6	
2391=	1011	CBP
2392=	50	GUAGCBP
2393= Expression	9005	-
2394=	8158	CCONS (158)
2395=	9005	-
<u>2396=</u>	9006	=
2397=	16	(a)
2398= GO TO	-6	(b)
<u>2399=</u>	2400	(c)
<u>2400=</u>	35	(a)
2401=	1011	(b) CBP
2402= SET	2405	(c)
2403=	0	(d)
<u>2404=</u>	0	(e)
2405=	2	
2406= Expression	50	GUAGCBP
<u>2407=</u>	9006	=

Figure 6. Blank COMMON (NSET) Example Contents (Concluded)

<u>I</u>	<u>CCONS(I)</u>
120 =	100.0000
121 =	.1000000E-2
122 =	100.0000
123 =	0.
124 =	0.
125 =	0.
126 =	0.
127 =	9.000000
128 =	1.000000
154 =	.1000000E-01
155 =	3.000000
156 =	.1000000E-01
157 =	4.000000
158 =	.1000000-01

Figure 7. CCONS Array Storage Example

NODE 2. Location 1587 is the first word of NODE 2, and it indicates that ten ARCs originate from this NODE. Location 1588 is a pointer to the PICK command on the first ARC leaving this NODE. Location 1589 is a pointer to the other commands on NODE 2 and points to location 1590. Location 1590 is a command for DEACTIVATE, and 1591 says that ARC M is to be DEACTIVATEed. Location 1592 is a pointer to the next command and points to location 1593 which is a SET command. Variable KODE1 (NSET(1594)=41) is the variable to be SET. Location 1595 is a pointer to where the expression is located, location 1596 is a truth condition indicator, and since it is zero there is no truth condition. Location 1597 is a pointer to the next command. Location 1598 indicates that there are eight items in the expression. The next eight items are the expression; it starts with an end-of-function indicator (NSET(1599)=10003) and continues with the variable KODE (NSET(1600)=40), constant of 100 (NSET(1601)=8120), a divide sign (NSET(1602)=9003), another constant, 0.001 (NSET(1603)= 8121), a plus sign (NSET(1604)=9004), the function IFIX (NSET(1605)= 9022), and an equal sign (NSET(1606)=9006).

The next command is another SET command. Variable KODE2 (NSET(1608)=42) is to be SET. Location 1609 is a pointer to the expression; there is no truth condition (NSET(1610)=0), and the next command starts at location 1619 (NSET(1611)=1619). There are six items in this expression (NSET(1612)=6): variable KODE (NSET(1613)=40), variable KODE1 (NSET(1614)=41), constant 100 (NSET(1615)=8122), multiply sign (NSET(1616)= 9002), subtraction sign (NSET(1617)=9005), and an equal sign (NSET(1618)= 9006).

The next command is an IF (NSET(1619)=48). The variable KODE (NSET(1620)=40) is tested to see if it is greater than (NSET(1621)=5) the expression at location 1626 (NSET(1622)=1626). Locations 1623 through 1625 point to the locations of the TRUE (NSET(1623)=1629), the FALSE (NSET(1624)=1649), and the LABEL (NSET(1625)=1669), respectively. The expression contains two items (NSET(1626)=2), and they are a constant of zero (NSET(1627)=8123) and an equal sign (NSET(1628)=9006).

The first command of the TRUE side is another IF (NSET(1629)=48), and variable CBP (NSET(1630)=1011) is tested to see if it is greater than (NSET(1631)=4) the expression at location 1636 (NSET(1632)=1636). The pointers to the TRUE (NSET(1633)=1639), FALSE (NSET(1634)=0), and LABEL (NSET(1635)=1645) are the next three items. Since the FALSE pointer is zero it indicates that there is no FALSE for this IF. The expression contains two items, and they are a constant of zero (NSET(1637)=8124) and an equal sign (NSET(1638)=9006). The first command on this TRUE is an ACTIVATE (NSET(1639)=1), and ARC M (NSET(1640)=1) is ACTIVATED. Location 1641 points to the next command which is a GOTO (NSET(1642)=16), and location 1643 indicates where to go, which is ARC M (NSET(1643)=1). Location 1644 is a pointer to the next command, and since it is zero (NSET(1644)=0) the end of the TRUE branch has been reached. Location 1645 is a LABEL (NSET(1645)=47), and locations 1646 and 1647 are the LABEL name of 22 (NSET(1646)=7724015) and (NSET(1647)=12516335). Location 1648 points to the next command, and since it is zero it indicates the end of the TRUE side of the outside IF.

Location 1649 is the start of the commands for the FALSE side of the outside IF and starts with a SET command (NSET(1649)=35). The variable KODE1 (NSET(1650)=41) is SET to the value of the expression at location 1654. There is no truth condition (NSET(1652)=0), but the next command starts at location 1659 (NSET(1653)=1659). There are four items (NSET(1654)=4) for this expression, and they are a constant of zero (NSET(1655)=8125), the variable KODE1 (NSET(1656)=41), a negative sign (NSET(1657)=9005), and an equal sign (NSET(1658)=9006).

The next command is a SET command, and the variable KODE2 (NSET(1660)=42) is SET to the value of the expression at 1664. There is no truth condition (NSET(1662)=0), and no other commands follow this one (NSET(1663)=0). There are four (NSET(1664)=4) items in this expression: a zero (NSET(1665)=8126), the variable KODE2 (NSET(1666)=42), a negative sign (NSET(1667)=9005), and an equal sign (NSET(1668)=9006). This finishes all of the commands on the FALSE side of the IF leg.

The LABEL command (NSET(1669)=47) is next and terminates the outside IF command. The LABEL name is 21 (NSET(1670)=7719919, NSET(1671)=12516335). The LABEL command points to the next command following the LABEL (NSET(1672)=1673).

This command is a SET command (NSET(1673)=35), and variable J (NSET(1674)=2) is SET to the value of the expression at location 1678 (NSET(1675)=1678). There is no truth condition (NSET(1676)=0). The next command starts at location 1681 (NSET(1677)=1681). There are two items (NSET(1678)=2) in the expression: a number 9 (NSET(1679)=8127) and an equal sign (NSET(1680)=9006).

The next command is an IF command (NSET(1681)=48). The variable J (NSET(1682)=2) is tested to see if it is equal to (NSET(1683)=3) the expression at location 1688 (NSET(1684)=1688). Location 1685 points to the location of the TRUE side (NSET(1685)=1693). There is no FALSE side (NSET(1686)=0), and the IF LABEL is at location 1755 (NSET(1687)=1755). The expression contains four items (NSET(1688)=4): an end of function indicator (NSET(1689)=10003), the variable KODE1 (NSET(1690)=41), the function ABS (NSET(1691)=9013), and an equal sign (NSET(1692)=9006).

The first command of the TRUE side is a SET command (NSET(1693)=35). The variable J (NSET(1694)=2) is set to the value of the expression at location 1698 (NSET(1695)=1698). There is no truth condition (NSET(1696)=0), and the next command starts at location 1701 (NSET(1697)=1701). There are two items in the expression: the number one (NSET(1699)=8128) and an equal sign (NSET(1700)=9006).

The next command is another IF command (NSET(1701)=48). The variable J (NSET(1702)=2) is tested to see if it is less than or equal to (NSET(1703)=1), the expression at location 1708 (NSET(1704)=1708). The TRUE side of the IF command starts at 1731 (NSET(1705)=1731), and again there is no FALSE side (NSET(1706)=0). The LABEL 23 starts at location 1755 (NSET(1707)=1755). There are twenty-two items in the expression (NSET(1708)=22): an end of function indicator (NSET(1709)=10003), the number of arguments in the function (NSET(1710)=1), the variable CBP (NSET(1711)=1011), the function F1 (NSET(1712)=9101), the number of TABLEs (NSET(1713)=1), a null TABLE (NSET(1714)=0), the number of parameter sets (NSET(1715)=1), a null parameter set (NSET(1716)=0), the number of entities (NSET(1717)=0), a truth condition of less than or equal (NSET(1718)=1), an end of function indicator (NSET(1719)=10003), the number of arguments in the function (NSET(1720)=1), the variable KODE

(NSET(1721)=40), the function F1 (NSET(1722)=9101), the number of TABLEs in the function (NSET(1723)=1), a null TABLE (NSET(1724)=0), the number of parameter sets (NSET(1725)=1), a null parameter set (NSET(1726)=0), the number of entities (NSET(1727)=0), a truth condition of less than (NSET(1728)=2), a plus sign (NSET(1729)=9004), and an equal sign (NSET(1730)=9006).

The first command on the TRUE side is a GOTO (NSET(1731)=16). Since the item in location 1732 is negative (NSET(1732)=-3) a NODE is where control is going to, and the NODE is the third NODE which is NODE 8. The next command starts at location 1734 (NSET(1733)=1734).

This command is a SET command (NSET(1734)=35). The variable NV(1) (NSET(1735)=2051) is set to the value of the expression at location 1739 (NSET(1736)=1739). There is no truth condition (NSET(1737)=0) and no other commands on this TRUE leg (NSET(1738)=0). There are fifteen items in the expression NSET(1739)=15: an end of function indicator (NSET(1740)=10003), the number of arguments in function (NSET(1741)=3), three variables KODE2 (NSET(1742)=42), DEPTH (NSET(1743)=1006), COST (NSET(1744)=1001), the function U2 (NSET(1745)=9131), the number of TABLEs (NSET(1746)=1), a null TABLE (NSET(1747)=0), the number of parameter sets (NSET(1748)=1), a null parameter set (NSET(1749)=0), the number of entities (NSET(1750)=2), two entities INTVCST (NSET(1751)=54), INTVTIM (NSET(1752)=55), no truth condition (NSET(1753)=0), and an equal sign (NSET(1754)=9006). This is the last command on the TRUE side of both the inner and outer IFs.

Location 1755 is the LABEL 23 which is the end of both IF statements. This is also the last command on NODE 2 (NSET(1758)=0).

NODE 1. Location 1759 is the first word of NODE 1 and it indicates that one ARC originates from this NODE (NSET(1759)=1). Location 1760 is a pointer to the location of the PICK command on that ARC (NSET(1760)=2195). Location 1761 is a pointer to the other commands, and since it is zero (NSET(1761)=0) there are no other commands and the end of NODE 1 has been reached.

ARC A. Location 2189 is the first word of ARC A, and it indicates that this ARC starts at NODE 1 (NSET(2189)=4). Location 2190

indicates that NODE 2 (NSET(2190)=2) is the normal termination of this ARC. Location 2191 contains the pointer to the UCS for the OPR command (NSET(2191)=2200). Location 2192 points to the location where the number of TERM and CROSS commands are stored (NSET(2192)=2206). Location 2193 is positive; so the initial status of this ARC is active. Location 2194 is also positive; so the current status of this ARC is active.

The PICK command information is next (NSET(2195)=29). Location 2196 is zero; this indicates that ARC A is the only ARC leaving NODE 1. Location 2197 is negative which indicates that there is no expression on this PICK, location 2198 is zero which indicates that there is no truth condition, and location 2199 is zero to show that there is no other PICK command to point toward.

Location 2200 is the start of the USC for the OPR command (NSET(2200)=28). The OPERATION to perform is the DRLING OPERATION (NSET(2201)=1), and there are no substitutable parameters (NSET(2202)=0). Location 2203 indicates that there is no TYPE S commands on the ARC (NSET(2203)=-1), location 2204 indicates there is no TYPE I commands (NSET(2204)=-1), and location 2205 indicates that there is no TYPE S commands (NSET(2205)=-1).

Location 2206 contains the number of TERM and CROSS commands which is seven (NSET(2206)=7). Location 2207 points to the location of the beginning of the first TERM or CROSS command (NSET(2207)=2208).

The first TERMination of the ARC is a CROSS command (NSET(2208)=44). This is CROSS number 1 (NSET(2209)=1), and location 2210 points to the location of the expression (NSET(2210)=2215). Location 2211 is the CROSSing tolerance, but since the NSET array was printed out in integer format, and the tolerance is a real number, the value isn't printed (NSET(2211)=*). The CROSSing condition is greater than or equal to (NSET(2212)=5). Location 2213 points to the location of the other commands on the CROSS (NSET(2213)=2220), and location 2214 points to the location of the start of the next TERM or CROSS (NSET(2214)=2228). There are four items in the expression: the variable DEPTH (NSET(2216)=1006),

the variable WELL(I) (NSET(2217)=-51), a negative sign (NSET(2218)=9005), and an equal sign (NSET(2219)=9006).

The first command on this CROSS is a SET command (NSET(2220)=35). The variable KODE (NSET(2221)=40) is SET to the value of the expression at location 2225 (NSET(2222)=2225). There is no truth condition (NSET(2223)=0), and there is no command following this one (NSET(2224)=0). There are two items in the expression: the variable KODES(I) (NSET(2226)=-61) and an equal sign (NSET(2227)=9006). Since there are no other commands on the CROSS the end of this CROSS has been reached.

The next TERMINation of this ARC is another CROSS command NSET(2228)=44) and is CROSS number 2 (NSET(2229)=2). Location 2230 (NSET(2230)=2235) is the pointer to the expression, location 2231 is the CROSSing tolerance (again a real number in integer format), location 2232 is the CROSSing condition of greater than or equal to (NSET(2232)=5), location 2233 points to the other commands on this CROSS (NSET(2233)=2240), and location 2234 points to the next TERM or CROSS (NSET(2234)=2243). There are four (NSET(2235)=4) items in the expression: the variable ARCLen (NSET(2236)=1005), variable NXSURV (NSET(2237)=43), a negative sign (NSET(2238)=9005), and an equal sign (NSET(2239)=9006).

The first command on this CROSS is a GOTO (NSET(2240)=16) command. Since location 2241 is negative, a NODE is where control is going, and the NODE is NODE 3 (NSET(2241)=-5). Location 2242 is a pointer to the next command, and since it is zero (NSET(2242)=0) there are no other commands; so this CROSS command is finished.

The next TERMINation of the ARC is a TERM command (NSET(2243)=39). This is TERM number 3 (NSET(2244)=3). Location 2245 points to the location of the expression (NSET(2245)=2250); locations 2246 and 2247 are not used in the TERM. Location 2248 points to the other commands on the TERM (NSET(2248)=2253), and location 2249 points to the next TERM or CROSS (NSET(2249)=2281). The expression contains two (NSET(2250)=2) items: the variable PULL (NSET(2251)=36) and an equal sign (NSET(2252)=9006).

The first command on the TERM is a GOTO command (NSET(2253)=16). Since location 2254 is negative, control is being transferred to a NODE, and the NODE is NODE 4 (NSET(2254)=-6). Location 2255 points to the location of the next command (NSET(2255)=2256).

The next command is an IF command (NSET(2256)=48). The variable DEPTH (NSET(2257)=1006) is tested to see if it is greater than or equal to (NSET(2258)=5) the expression pointed to by location 2259 (NSET(2259)=2263). Location 2260 points to the start of the TRUE side of the IF (NSET(2260)=2266), location 2261 is zero so there is no FALSE side of the IF, and location 2262 points to the terminating LABEL of the IF (NSET(2262)=2277). There are two (NSET(2263)=2) items in the expression: the variable WELL(I) (NSET(2264)=-51) and an equal sign (NSET(2265)=9006).

The first command on the TRUE side is a GOTO command (NSET(2266)=16). Since location 2267 is negative, a NODE is where control is going, and the NODE is NODE 2 (NSET(2267)=-2). Location 2268 points to the location of the next command (NSET(2268)=2269).

This command is a SET command (NSET(2269)=35). The variable KODE (NSET(2270)=40) is SET to the value of the expression pointed to by location 2271 (NSET(2271)=2274). There is no truth condition (NSET(2272)=0), and there is no command following this command (NSET(2273)=0). There are two (NSET(2274)=2) items in the expression: the variable KODES(I) (NSET(2275)=-61) and an equal sign (NSET(2276)=9006).

The next command is the LABEL command (NSET(2277)=47) which terminates the IF command. Since this command doesn't point to any other command, (NSET(2280)=0) this is also the end of this TERM.

The next TERMINATION of the ARC is a CROSS command (NSET(2281)=44). This is CROSS number four (NSET(2282)=4). Location 2283 points to the location of the expression (NSET(2283)=2288); location 2284 is the CROSSING tolerance (again a real number); location 2285 is the CROSSING condition of greater than or equal to (NSET(2285)=5); location 2286 points to the other commands on this CROSS (NSET(2286)=2293), and location 2287 points to the next TERM or CROSS command (NSET(2287)=2325).

There are four (NSET(2288)=4) items in this expression: the variable CBP (NSET(2289)=1011), the variable NXCON (NSET(2290)=16), a negative sign (NSET(2291)=9005), and an equal sign (NSET(2292)=9006).

The first command on this CROSS is a GOTO command (NSET(2293)=16). Since location 2294 is negative, a NODE is where control is going, and the NODE is NODE 13 (NSET(2294)=-13). Location 2295 is a pointer to the next command (NSET(2295)=2296).

The next command on this CROSS is an IF command (NSET(2296)=48). The variable PULL (NSET(2297)=36) is tested to see if it is less than (NSET(2298)=2) the expression pointed to by location 2299 (NSET(2299)=2303). Location 2300 points to the TRUE side of the IF command (NSET(2300)=2308), location 2301 points to the FALSE side (NSET(2301)=2311), and location 2302 points to the location of the LABEL (NSET(2302)=2321). There are four items in the expression (NSET(2303)=4): the variable TIME (NSET(2304)=0), the variable CONCTIM (NSET(2305)=17), a plus sign (NSET(2306)=9004), and an equal sign (NSET(2307)=9006).

The first command on the TRUE side is a GOTO command (NSET(2308)=16). Since location 2309 is negative, a NODE is where control is going, and the NODE is NODE 4 (NSET(2309)=-6). Since location 2710 is zero, it doesn't point to another command; so this is the end of the TRUE side of the IF command.

The first command on the FALSE side is a SET command (NSET(2311)=35). The variable PULL (NSET(2312)=36) is SET to the value of the expression pointed to by location 2313 (NSET(2313)=2316). There is no truth condition (NSET(2314)=0), and this command doesn't point to any other; so this command is the last command on the FALSE side of the IF command (NSET(2315)=0). There are four items in the expression (NSET(2316)=4): the variable PULL (NSET(2317)=36), the variable CONCTIM (NSET(2318)=17), a plus sign (NSET(2319)=9004), and an equal sign (NSET(2320)=9006).

Location 2321 contains the LABEL command (NSET(2321)=47), and the LABEL is C41 (NSET(2322)=915247, NSET(2323)=12516335). Since location 2324 is zero, no other commands are pointed to; so this is the end of this CROSS command (NSET(2324)=0).

The next TERMination of the ARC is another CROSS command (NSET(2325)=44). This is CROSS number five (NSET(2326)=5). Location 2327 points to the location of the expression (NSET(2327)=2332), location 2728 is the CROSSing tolerance (again a real number), location 2329 is the CROSSing condition of greater than or equal to (NSET(2329)=5), location 2330 points to the other commands on this CROSS command (NSET(2330)=2339), and location 2331 points to the next TERM or CROSS command (NSET(2331)=2350). The expression contains six (NSET(2332)=6) items: the variable DEPTH (NSET(2333)=1006), the variable FDEPTH (NSET(2334)=51) a negative sign (NSET(2335)=9005), the constant 0.01 (NSET(2336)=8154), a negative sign (NSET(2337)=9005), and an equal sign (NSET(2338)=9006).

The first command in this CROSSing is a GOTO command (NSET(2339)=16). Since location 2340 is negative, a NODE is where control is going, and the NODE is NODE 16 (NSET(2340)=-15). Location 2341 is a pointer to the next command (NSET(2341)=2342).

The next command is a SET command (NSET(2342)=35). The variable M (NSET(2343)=5) is SET to the value of the expression at the location pointed to by location 2344 (NSET(2344)=2347). There is no truth condition (NSET(2345)=0), and no other commands follow this command (NSET(2346)=0). The expression contains two (NSET(2347)=2) items: the number four (NSET(2348)=8155) and an equal sign (NSET(2349)=9006). Since no other commands follow this command, this is the end of this CROSSing.

The next card of this ARC is a CROSS command (NSET(2350)=44); this is CROSS number 6 (NSET(2351)=6). Location 2352 points to the location of the expression (NSET(2352)=2357), location 2353 is the CROSSing tolerance (again a real number), location 2354 is the CROSSing condition of greater than or equal to (NSET(2354)=5), location 2355 points to the other commands on this CROSS (NSET(2355)=2364), and location 2356 points to the next TERM or CROSS (NSET(2356)=2380). There are six items in the expression (NSET(2357)=6): the variable CBP (NSET(2358)=1011), the variable OUTGUAG (NSET(2359)=49), a negative sign (NSET(2360)=9005), the number 0.01 (NSET(2361)=8156), a negative sign (NSET(2362)=9005), and an equal sign (NSET(2363)=9006).

The first command on this CROSS is a GOTO command (NSET(2364)=16). Since location 2365 is negative, a NODE is where control is going, and the NODE is NODE 16 (NSET(2365)=-15). Location 2366 is a pointer to the next command (NSET(2366)=2367).

The next command is a SET command (NSET(2367)=35). The variable M (NSET(2368)=5) is set to the value of the expression pointed to by location 2369 (NSET(2369)=2372). There is no truth condition (NSET(2370)=0), and location 2371 points to the next command (NSET(2371)=2375). The expression contains two (NSET(2372)=2) items: the number four (NSET(2373)=8157) and an equal sign (NSET(2374)=9006).

The next command is another SET command (NSET(2375)=35). The variable CBP (NSET(2376)=1011) is set to the value of the expression pointed to by location 2377 (NSET(2377)=2380). There is no truth condition (NSET(2378)=0) and no other commands (NSET(2379)=0). There are two items in the expression (NSET(2380)=2): the variable OUTGUAG (NSET(2381)=49) and an equal sign (NSET(2382)=9006). Since no other commands follow this command, the end of this CROSSing has been reached.

The last TERMination of this ARC is a CROSS command (NSET(2383)=44). This is CROSS number seven (NSET(2384)=7). Location 2385 points to the location of the expression (NSET(2385)=2390), location 2386 is the CROSSing tolerance (again a real number in integer format), the truth condition is greater than or equal to (NSET(2387)=5), and the first command is pointed to by location 2388 (NSET(2388)=2397). Since location 2389 is zero, this is the last CROSS or TERM. The expression contains six items (NSET(2390)=6): the variables CBP (NSET(2391)=1011) and GUAGCBP (NSET(2392)=50), a negative sign (NSET(2393)=9005), the number 0.01 (NSET(2394)=8158), a negative sign (NSET(2395)=9005), and an equal sign (NSET(2396)=9006).

The first command on this CROSSing is a GOTO command (NSET(2397)=16). Since location 2398 is negative, a NODE is where control is going, and the NODE is NODE 4 (NSET(2398)=-6). Location 2399 is a pointer to the next command (NSET(2399)=2400).

This command is a SET command (NSET(2400)=35). The variable CBP (NSET(2401)=1011) is SET to the value of the expression pointed to by location 2402 (NSET(2402)=2405). There is no truth condition (NSET(2403)=0), and no commands follow this one (NSET(2404)=0). The expression contains two items (NSET(2405)=2): the variable GUAGCBP (NSET(2406)=50) and an equal sign (NSET(2407)=9006). Since this is the last command of the CROSS, this is the end of this CROSSing. Since this CROSS doesn't point to any other CROSS or TERM commands, this is the end of ARC A.

M. CROSS REFERENCE BETWEEN ROUTINE NUMBER AND NAME

Table 7 shows a cross-reference of all the IOSYM routines from number to name. Table 8 shows a cross-reference of all the IOSYM routines from name to number.

TABLE 7. ROUTINE CROSS-REFERENCES FROM NUMBER TO NAME

1 = PARSE	31 = SRHARC	61 = LBLCHK
2 = DVFND	32 = TRMSTR	62 = NOT USED
3 = NUMFND	33 = ZERO	63 = UNCODE
4 = FNFND	34 = NODSTR	64 = TYPSTR
5 = FUNC1	35 = PLTSTR	65 = GBLSTR
6 = FUNC4	36 = RPTSTR	66 = TOFSTR
7 = FUNC5	37 = TBTSTR	67 = CHOICE
8 = FUNC6	38 = TBLFND	68 = CHOIC2
9 = LOLFND	39 = ADATA	69 = CXDATN
10 = FUNC2	40 = CONVRT	70 = JOESTR
11 = FUNC3	41 = EEROR	71 = PRTSTR
12 = ALPFND	42 = KWFND	72 = DDPRT
13 = CONVR2	43 = PROCES	73 = RESCC
14 = ARAYCK	44 = PCKCK	74 = SAVE
15 = BLDDLML	45 = REPSTR	75 = EEOF
16 = BLDNUM	46 = DVSTR	76 = NETSUM
17 = BLDALP	47 = EVTSTR	77 = UNC01
18 = LITFND	48 = GTOSTR	78 = PPRPLT
19 = SCAN	49 = VARFND	79 = RSAVIN
20 = TRUTH	50 = DVARFD	80 = WSAVIN
21 = OPRSTR	51 = RESEQ	81 = FUNC4A
22 = FUNSTR	52 = FORSTR	82 = ARANGE
23 = BLDOPR	53 = COLSTR	83 = USRDAT
24 = FUNFND	54 = EEVAL	84 = PUTSTR
25 = LIBFUN	55 = AAIRP	85 = STASTR
26 = NAMFND	56 = ENDNET	86 = GETSTR
27 = NUMSTR	57 = LOPSTR	87 = GGGET
28 = ARCSTR	58 = LBLSTR	88 = PPPUT
29 = PCKSTR	59 = IFSTR	89 = PPPRNT
30 = SETSTR	60 = NOT USED	90 = EEPRNT

TABLE 7. ROUTINE CROSS-REFERENCES FROM NUMBER TO NAME (Concluded)

91 = TRCSTR	514 = VVARBL	537 = WWRT
92 = TTRAC	515 = PPIDF	538 = NNODST
93 = UUPRNT	516 = ZZSET	539 = AARCST
94 = DDUMP	517 = IISF	540 = INTLC
95 = WTHSTR	518 = VVMOV	541 = DATIN
96 = ALPSRH	519 = IIMOV	542 = DDOSIM
97 = BLOCK DATA	520 = SSHIFT	543 = WWTMHN
98 = BXDATN	521 = BBLIST	544 = XXINTL
99 = NOT USED	522 = SCOND	545 = XXDATN
100 - 500 = NOT USED	523 = STATE	546 = OTPUT
501 = GGTab	524 = EEVLTF	547 = RREPRT
502 = PPTAB	525 = DDOEVT	548 = PPRRPT
503 = GGPAR	526 = RREWRK	549 = PPRTAB
504 = PPPAR	527 = RREPLC	550 = NOT USED
505 = SSWARC	528 = MNXNOD	551 = SSAVE
506 = IIACST	529 = GGOTO	552 = TTAPES
507 = NOT USED	530 = DDOCMD	553 = DOSTR
508 = NOT USED	531 = DDOLST	554 = NOT USED
509 = GGVAR	532 = EVNTS	555 = SSTWRK
510 = PPVAR	533 = CCOLC	556 = PPFCMD
511 = EEXP	534 = VVWTMN	557 = GGSTEP
512 = FF	535 = PPRCOL	558 = GGSTAT
513 = PP	536 = TTABU	

TABLE 8. ROUTINE CROSS-REFERENCES FROM NAME TO NUMBER

AAIROP = 55	DVARFD = 50	GGSTEP = 557
AARCST = 539	DVFND = 2	GGTAB = 501
ADATA = 39	DVSTR = 46	GGVAR = 509
ALPFND = 12	EEOF = 75	GTOSTR = 48
ALPSRH = 96	EEPRNT = 90	IFSTR = 59
ARANGE = 82	EEROR = 41	IIACST = 506
ARRAYCK = 14	EEVAL = 54	IIMOV = 519
ARCSTR = 28	EEVLTF = 524	IISSET = 517
BBLIST = 521	EEXP = 511	INTLC = 540
BLDALP = 17	ENDNET = 56	JOESTR = 70
BLDDLM = 15	EVNTS = 532	KWFND = 42
BLDNUM = 16	EVTSTR = 47	LBLCHK = 61
BLDOPR = 23	FF = 512	LBLSTR = 58
BLOCK DATA = 97	FNFND = 4	LIBFND = 25
BXDATN = 98	FORSTR = 52	LITFND = 18
CCOLC = 533	FUNC1 = 5	LOLFND = 9
CHOICE = 67	FUNC2 = 10	LOPSTR = 57
CHOIC2 = 68	FUNC3 = 11	NAMFND = 26
COLSTR = 53	FUNC4 = 6	NETSUM = 76
CONVRT = 40	FUNC4A = 81	NNODST = 538
CONVR2 = 13	FUNC5 = 7	NNXNOD = 528
CXDATN = 69	FUNC6 = 8	NODSTR = 34
DATIN = 541	FUNFND = 24	NUMFND = 3
DDOCMD = 530	FUNSTR = 22	NUMSTR = 27
DDOEVT = 525	GBLSTR = 65	OPRSTR = 21
DDOSIM = 542	GETSTR = 86	OTPUT = 546
DDOLST = 531	GGGET = 87	PARSE = 1
DDPRNT = 72	GGOTO = 529	PCKCK = 44
DDUMP = 94	GGPAR = 503	PCKSTR = 29
DOSTR = 553	GGSTAT = 558	PLTSTR = 35

TABLE 8. ROUTINE CROSS-REFERENCES FROM NAME TO NUMBER (Concluded)

PP = 513	RREPET = 547	TTABU = 536
PPFCMD = 556	RREWRK = 526	TTAPES = 552
PPIDF = 515	RSVIN = 79	TTRAC = 92
PPPAR = 504	SAVE = 74	TYPSTR = 64
PPPRNT = 89	SCAN = 19	UNCO1 = 77
PPPUT = 88	SCOND = 522	UNCODE = 63
PPRCOL = 535	SETSTR = 30	USRDAT = 83
PPRPLT = 78	SRHARC = 31	UUPRNT = 93
PPRRPT = 548	SSAVE = 551	VARFND = 49
PPRTAB = 549	SSHIFT = 520	VVARBL = 514
PPTAB = 502	SSTWFK = 555	VVMOV = 518
PPVAR = 510	SSWARC = 505	VVWTMN = 534
PROCES = 40	STATE = 523	WSVIN = 80
PRTSTR = 71	STASTR = 35	WTHSTR = 95
PUTSTR = 84	TBLFND = 38	WWRIT = 537
REPSTR = 45	TBTSTR = 37	WWTHMN = 543
RESCC = 73	TOFSTR = 56	XXDATN = 545
RESEQ = 51	TRCSTR = 91	XXINTL = 544
RPTSTR = 36	TRMSTR = 32	ZERO = 33
RREPLC = 527	TRUTH = 20	ZZSET = 516

IV. DATA DESCRIPTION

Most of the information used in various routines in both the input processing section and the execution section is stored in COMMON blocks. Very little information is passed as formal parameters. This section is an alphabetic list, by variable name, of all the COMMON blocks variables, the COMMON blocks they are located in, their length, routine defined by, routine used in, and their definition. GASP COMMON blocks are not discussed in this section; for GASP COMMON blocks see the GASP-IV text.

1. NAME: CCONS
2. LENGTH: 300
3. COMMON BLOCK LOCATION: I02
4. ROUTINES IN WHICH DEFINED:

EEVAL	PLTSTR	TBLFND
LOLFND	PROCES	TBTSTR
LOPSTR	PUTSTR	TRMSTR
NUMFND	RESCC	VARFND
PCKSTR	RESEQ	ZERO

5. ROUTINES IN WHICH USED:

DDUMP	PLTSTR	TBTSTR
EEVAL	PROCES	TRMSTR
GETSTR	PUTSTR	VARFND
LOLFND	RESCC	VVARBL
PARSE	STASTR	WTHSTR
PCKSTR	TBLFND	

6. DEFINITION: This array contains all of the constants in a NETWORK. Also, the Input Processor uses it for temporary storage while processing the IOSYM commands. It is used to store intermediate results during an expression evaluation. If the dimension of this array is changed, then the variable MMCNS, in routine USRDAT, should be set to the value of the CCONS dimension. The value of the variable NNCNS is the current number of constants in the CCONS array.

1. NAME: DDIN
 2. LENGTH: 100
 3. COMMON BLOCK LOCATION: IO2
 4. ROUTINE IN WHICH DEFINED: ZERO
 5. ROUTINES IN WHICH USED: GGVAR INTLC
 6. DEFINITION: This array contains the initial values of the derivative variables. This array is used to initialize the DD array before each run. If the dimension of this array is changed, then the variable MMV1, in routine DFAUT, should be set to the value of the DDIN dimension. The value of the variable NNV1 is the number of primary variables and, therefore, the number of derivative variables in the DDIN array.
NOTE: The dimension of the SS, DD, SSIN, NNPT1, and DDIN arrays should all be the same.
-

1. NAME: DUMP
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: ADATA PROCES
5. ROUTINE IN WHICH USED: CHOICE
6. DEFINITION: This is a dump indicator. The dump consists of a print out of all the IOSYM arrays. If DUMP is TRUE, the dump is printed prior to performing any simulation, but after initialization. A dump will also be produced immediately after reading a new network if a fatal input error occurred, and DUMP is TRUE.

1. NAME: ECHO
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: ME1
 4. ROUTINES IN WHICH DEFINED: ADATA PROCES
 5. ROUTINES IN WHICH USED: EEROR SCAN
 6. DEFINITION: This is a print indicator. If ECHO is TRUE, then all the IOSYM input records read are printed on the output file. If ECHO is FALSE, then the IOSYM input records read during a NETWORK definition are not printed unless there is an error in the input record and INFO is TRUE. Printing of error and warning messages is determined by the INFO and NOINFO commands. The value of ECHO is set by the IOSYM commands ECHO and NOECHO.
-

1. NAME: FIRST
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: CHOICE SCAN
5. ROUTINES IN WHICH USED: PARSE SCAN
6. DEFINITION: This is a card indicator. If FIRST is TRUE, then the first input record of IOSYM input has not been read. If FIRST is FALSE, then the first input record of IOSYM input has been read. This indicator is used by routines PARSE and SCAN to do initialization before processing the first input record.

1. NAME: ICHAR
2. LENGTH: 50
3. COMMON BLOCK LOCATION: ME1
4. ROUTINE IN WHICH DEFINED: BLOCK DATA
5. ROUTINES IN WHICH USED:

DDOSIM	PPRPLT	UNCODE
NETSUM	PPRRPT	UNC01
PPPRNT	PPRTAB	UPLOT
PPRCOL	SCAN	WTHMN

6. DEFINITION: This array contains all of the characters recognized by IOSYM. For the processing of characters during input, the location in the ICHAR array of the character is the character number. ICHAR contains alphanumeric information as follows.

ICHAR (1) = A	ICHAR (18) = R	ICHAR (35) = 8
(2) = B	(19) = S	(36) = 9
(3) = C	(20) = T	(37) = .
(4) = D	(21) = U	(38) = **
(5) = E	(22) = V	(39) = *
(6) = F	(23) = W	(40) = /
(7) = G	(24) = X	(41) = +
(8) = H	(25) = Y	(42) = -
(9) = I	(26) = Z	(43) = =
(10) = J	(27) = 0	(44) =)
(11) = K	(28) = 1	(45) = (
(12) = L	(29) = 2	(46) = ,
(13) = M	(30) = 3	(47) = blank
(14) = N	(31) = 4	(48) = \$
(15) = O	(32) = 5	(49) = not used
(16) = P	(33) = 6	(50) = not used
(17) = Q	(34) = 7	

1. NAME: ICLEVL
2. LENGTH: 4
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: PARSE PROCES
5. ROUTINES IN WHICH USED:

CHOIC2	PROCES	TBLFND
DVFND	PRTSTR	TYPSTR
EEROR	SCAN	VARFND
PARSE	SRHARC	
6. DEFINITION: This array contains the current command level hierarchy. When command I is processed and KEYWORD(I,3) is greater than zero, then one is subtracted from KEYWORD(I,3), and two is taken to the power of the resultant number. ICLEVL(LEVEL) is the last command level stored. When command J is processed, the value in ICLEVL(LEVEL) is compared with the value of KEYWORD(J,4) to see if the bits set in ICLEVL(LEVEL) are set in the number KEYWORD(J,4). If the bits set in ICLEVL(LEVEL) are set in KEYWORD(J,4), the command J is valid in the current level. If not, command J is not valid, and an error is processed.

-
1. NAME: IFFJ
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: FFJ
 4. ROUTINES IN WHICH DEFINED: FFPUDJ PACWJ
 5. ROUTINE IN WHICH USED: PACWJ
 6. DEFINITION: This is the location in the LWJ array of the first non-blank character.

1. NAME: IIARC
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINE IN WHICH DEFINED: AARCST
 5. ROUTINES IN WHICH USED:

AARCST	EEPRNT	TTRACE
BBLIST	EVNTS	UUPRNT
DDPRNT	TTABU	
 6. DEFINITION: This is the ARC number of the current ARC being processed in the IOSYM simulation. If IIARC is less than or equal to zero, then an ARC is not being processed; a NODE is.
-

1. NAME: IIDOBJ
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINE IN WHICH DEFINED: ZERO
5. ROUTINE IN WHICH USED: Not used at this time.
6. DEFINITION: This word is the pointer to the location in the NSET array where the start of the objective function for the MINIMIZE command is located. The MINIMIZE command is not operational at this time.

1. NAME: IIGASP
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: DOSTR PROCES
 5. ROUTINE IN WHICH USED: DDOSIM
 6. DEFINITION: This variable indicates whether to read GASP input or not before each simulation. If IIGASP is equal to zero, then GASP input is not read. If IIGASP is equal to one, then GASP input is processed before each simulation. IIGASP is set to one if there is a GASP command between the DO and GO commands.
-

1. NAME: IIGO
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINE IN WHICH DEFINED: PROCES
5. ROUTINE IN WHICH USED: INTLC
6. DEFINITION: This is the NODE number of the first NODE to process in an IOSYM simulation.

1. NAME: IINXAC
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: AARCST GGOTO
 INTLC NNODST
 5. ROUTINES IN WHICH USED: AARCST EVNTS
 NNODST
 6. DEFINITION: This is the ARC number of the next ARC to process. If IINXAC is less than or equal to zero, then an ARC isn't the next command to process, a NODE is, and the next NODE number is stored in IINXND.
-

1. NAME: IINXND
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: AARCST GGOTO
 INTLC NNXNOD
5. ROUTINES IN WHICH USED: DDPRNT EEPRNT
 NNODST NNXNOD
 TTRACE UUPRNT
6. DEFINITION: This is the current NODE number being processed or the next NODE number to process. If IINXAC is greater than zero, then an ARC is next, not a NODE.

1. NAME: IIOP
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: AARCST PROCES
ZERO
5. ROUTINES IN WHICH USED:
AARCST EEPRT TTRACE
BBLIST PROCES TYPSTR
CCOLC TTABU UUPRNT
DDPRNT
6. DEFINITION: This is the OPERATION number of the current OPERATION. During IOSYM input, routine PROCES uses the variable IIOP to keep track of which OPERATION is being inputted. During an IOSYM simulation, IIOP determines which OPERATION is currently being processed.

1. NAME: IIOPR
2. LENGTH: 10 x 2
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: AARCST GBLSTR
5. ROUTINES IN WHICH USED: AARCST PPIDF
UNCODE VVARBL
6. DEFINITION: This array contains the substitutable parameter information. If the first dimension of this array is changed, then the variable MMOPR, in routine USRDAT, should be set to the value of the first IIOPR dimension.

IIOPR (I,1). IIDVs of current substitutable parameters.

IIOPR (I,2). IIDVs of default substitutable parameters.

1. NAME: IIRPT
2. LENGTH: 50
3. COMMON BLOCK LOCATION: IO1
4. ROUTINES IN WHICH DEFINED: RPTSTR ZERO
5. ROUTINES IN WHICH USED: DDUMP NETSUM
PPRRPT RPTSTR
RREPRT
6. DEFINITION: This array contains the IOSYM IIDVs for all the variables to be collected and reported in the REPORT statistics. If the dimension of this array is changed, then the variable MMRPT, in routine USRDAT, should be set to the value of the IIRPT dimension. The value of the variable NNRPT is the number of variables in the IIRPT array.

-
1. NAME: IISIM
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINE IN WHICH DEFINED: DDOSIM
 5. ROUTINES IN WHICH USED: DDOSIM PPRCOL
PPRRPT PPRTAB
UPLOT (user written) VVWTMN
WWTMNM
 6. DEFINITION: This is the current number of the simulation in progress.

1. NAME: I72
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ALLJ
4. ROUTINE IN WHICH DEFINED: SETJ
5. ROUTINES IN WHICH USED: PIVECJ PKMATJ
PMATJ PSTRJ
PVECJ
6. DEFINITION: This is an indicator. If I72 is equal to zero, then the output will be printed out for use at a terminal. If I72 is equal to one, then the output will be printed out for use on a line printer. This applies only to the routines mentioned above.

1. NAME: JRD
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ALLJ
4. ROUTINE IN WHICH DEFINED: SETJ
5. ROUTINE IN WHICH USED: PACWJ
6. DEFINITION: This is the device code number used to read input. It is set internally for the free form input utility programs.

1. NAME: KEYWORD
2. LENGTH: 62 x 5
3. COMMON BLOCK LOCATION: ME1
4. ROUTINE IN WHICH DEFINED: BLOCK DATA
5. ROUTINES IN WHICH USED: KWFND

PARSE

ZERO

6. DEFINITION: This array contains command keyword information. If the dimension of the KEYWORD array is changed, then the variable MAXKW, in routine PARSE, should be set to the value of the first KEYWORD dimension.

KEYWORD(I,1). The first four characters of command I's name in internal IOSYM representation.

KEYWORD(I,2). The next three characters of command I's name, plus a blank, in internal IOSYM representation.

KEYWORD(I,3). The command level power. If KEYWORD(I,3) is greater than zero, then two to the power of KEYWORD(I,3)-1 is the command level number for command I. Some of the commands change the command level hierarchy of the IOSYM input processing. Certain commands can only occur after other commands. To keep track of the command level hierarchy when a command level is changed, the command level number calculated from the KEYWORD(I,3) is stored in the ICLEVL array. If KEYWORD(I,3) is equal to zero, then the command level does not change with command I.

KEYWORD(I,4). This is the sum of the command levels that command I can occur on. Since all the command levels are a power of two, when added together, there is no carrying in binary addition; so KEYWORD(I,4) can be decoded into the powers of two that composed it initially. If the value in the ICLEVL(LEVEL) location is one of the numbers in KEYWORD(I,4), then the Ith command is valid in the current command level.

KEYWORD(I,5). The pointer to where the syntax information for the Ith command starts in the NTSFMT array. If KEYWORD(I,5) is equal to zero, then there is no syntax for command I other than the command itself.

1. NAME: LEVEL
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: PARSE PROCES
5. ROUTINES IN WHICH USED: CHOIC2 DVFND
 PARSE PPFCMD
 PROCES PRTSTR
 SRHARC TBLFND
 TYPSTR VARFND
6. DEFINITION: This is the last location in the ICLEVL array where the current command level number is stored.

1. NAME: LIBFUN
2. LENGTH: 16 x 2
3. COMMON BLOCK LOCATION: ME1
4. ROUTINE IN WHICH DEFINED: USF DAT
5. ROUTINES IN WHICH USED: LIBFND PARSE
 UNCODE
6. DEFINITION: This array contains the FORTRAN library function names.

LIBFUN(I,1). The first four characters of the FORTRAN library function I in internal IOSYM representation.

LIBFUN(I,2). The next three characters of the FORTRAN library function I, plus a blank in internal IOSYM representation. The FORTRAN library functions stored in this array are:

1	ATAN	9	COS
2	CMPLX	10	EXP
3	AMAX1	11	ALOG
4	AMIN1	12	SIGN
5	AMOD	13	SQRT
6	SIGN	14	TAN
7	ABS	15	CONGJ
8	ATAN2	16	IFIX

1. NAME: LLARY
2. LENGTH: 2 x 20
3. COMMON BLOCK LOCATION: I02
4. ROUTINES IN WHICH DEFINED: PARSE ZERO
5. ROUTINES IN WHICH USED: JOESTR LBLSTR
6. DEFINITION: This array contains LABEL information. If the dimension of this array is changed, then the variable MMARY, in routine USRDAT, should be set to the value of the second LLARY dimension. The value of the variable NNARY is the number of LABELs stored in the LLARY array.

LLARY(1,I). Is the first four characters of LABEL I's name in internal IOSYM representation.

LLARY(2,I). Is the next three characters of LABEL I's name, plus a blank, in internal IOSYM representation.

NOTE: The second dimension of the LLARY array and the second dimension of the IIFPT array should be the same.

-
1. NAME: LLCOL
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: I03
 4. ROUTINE IN WHICH DEFINED: SSTWRK
 5. ROUTINES IN WHICH USED: CCOLC PPRCOL
 6. DEFINITION: This is the location in the NSET array to store the first COLLECT statistics. If LLCOL is less than or equal to zero, no COLLECT statistics are in effect.

1. NAME: LLCROS
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: AARCS1 RREWRK
 SSTWRK
 5. ROUTINES IN WHICH USED: EVNTS RREWRK
 SCND STATE
 6. DEFINITION: This is the location in the NNWRK array that points to the uniform command sequence for the first CROSS command, on the current ARC, in the NSET array. If LLCROS is less than or equal to zero, then there are no CROSS commands on the current ARC being processed in the IOSYM simulation.
-

1. NAME: LLEVNT
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: DDOEVT RREWRK
 SSTWRK
5. ROUTINES IN WHICH USED: DDOEVT RREWRK
6. DEFINITION: Before subroutine UEVENT it called, the values of the parameters are copied into consecutive locations in the WWRK array. The location of the first parameter value is LLEVNT.

1. NAME: LLIST
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: BBLIST RREWRK
SSTWRK
 5. ROUTINES IN WHICH USED: BBLIST DDOLST
EVNTS RREWRK
 6. DEFINITION: A command list is constructed for type S, I, and E commands by subroutine BBLIST. The list is stored in the array NNWRK and starts at location LLIST.
-

1. NAME: LLRPT
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: RREWRK SSTWRK
5. ROUTINES IN WHICH USED: PPRRPT RREPRT
RREWRK
6. DEFINITION: This is the location in the NNWRK array where storage for REPORT statistics begins. If LLRPT is less than or equal to zero, then no storage for REPORT statistics has been allocated.

1. NAME: LLTABU
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: SSTWRK TBTSTR
 5. ROUTINES IN WHICH USED: TBTSTR TTABU
 6. DEFINITION: When processing input, LLTABU indicates the number of the current TABULATION being processed. During a simulation, LLTABU indicates the location in the NNWRK array where the TABULATE statistics information is stored.
-

1. NAME: LLTRAC
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: TRCSTR ZERO
5. ROUTINES IN WHICH USED: AARCST NNODST
 TRCSTR TTRACE
6. DEFINITION: If LLTRAC is greater than zero, then LLTRAC points to the NSET array where the TRACE information is stored. If LLTRAC is equal to zero, then no TRACE information has been processed on this computer run. If LLTRAC is less than zero, then absolute value of LLTRAC points to the location in the NSET array where inactive TRACE information is stored.

1. NAME: LLWITH
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: DOSTR PROCES
RREWRK WTHSTR
 5. ROUTINES IN WHICH USED: DDOSJM PROCES
RREWRK VVWTMN
WTHSTR WWTMNM
 6. DEFINITION: This is the location in the NNWRK array where the first word of WITH or MINIMIZE information is located. If LLWITH is zero, then there is no WITH or MINIMIZE information.
-

1. NAME: LLWRK
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: AARCST BBLIST
DDOEV T DOSTR
PROCES RREWRK
SSTWRK WTHSTR
5. ROUTINES IN WHICH USED: AARCST BBLIST
DDOEV T EEVAL
PROCES RREWRK
SSTWRK
6. DEFINITION: This is the next available location in the work storage array (NNWRK/WRK) that can be used in any manner. LLWRK minus one locations in the work storage array have already been used.

1. NAME: LLXSET
2. LENGTH: 1
3. COMMON BLOCK LOCATION: I03
4. ROUTINES IN WHICH DEFINED:

ARCSTR	CHOIC2	DVSTR
ENDNET	EVTSTR	FUNSTR
GETSTR	GTOSTR	IFSTR
JOESTR	LBLSTR	LOPSTR
NODSTR	OPRSTR	PCKCK
PCKSTR	PROCES	PRTSTR
PUTSTR	REPSTR	SETSTR
SRHARC	STASTR	TOFSTR
TRCSTR	TRMSTR	TYPSTR
ZERO		

5. ROUTINES IN WHICH USED:

ARCSTR	DDUMP	DVSTR
ENDNET	EVTSTR	FUNSTR
GETSTR	GTOSTR	IFSTR
JOESTR	LBLSTR	LOPSTR
NODSTR	OPRSTR	PCKCK
PCKSTR	PROCES	PRTSTR
PUTSTR	REPSTR	SETSTR
SRHARC	SSTWRK	STASTR
TOFSTR	TRCSTR	TRMSTR
TYPSTR		

6. DEFINITION: Last position in the NSET array used to store the link listed NETWORK.

1. NAME: LLXWRK
2. LENGTH: 1
3. COMMON BLOCK LOCATION: I03
4. ROUTINES IN WHICH DEFINED:

ALPSRH
 FUNC2
 FUNC4A
 NAMFND
 RESCC
 ZERO

DVFND
 FUNC3
 FUNC5
 NUMFND
 RESEQ

FNFND
 FUNC4
 LOLFND
 PARSE
 TBLFND

5. ROUTINES IN WHICH USED:

ALPSRH
 FNFND
 FUNC3
 FUNC5
 NUMFND
 RESEQ
 TBTSTR

COLSTR
 FORSTR
 FUNC4
 LOLFND
 PARSE
 RPTSTR
 WTHSTR

DVFND
 FUNC2
 FUNC4A
 NAMFND
 RESCC
 TBLFND

6. DEFINITION: This is the last position in the NNWRK that has been used for processing a command during NETWORK input. LLXWRK is reset to zero before the processing of each command.

1. NAME: LWJ
2. LENGTH: 81
3. COMMON BLOCK LOCATION: FFJ
4. ROUTINE IN WHICH DEFINED: PACWJ
5. ROUTINE IN WHICH USED: PACWJ

6. DEFINITION: This array contains the last input record read by the free form utility programs. Each word contains one character.

1. NAME: MAXEXP
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: MCHDEF
 4. ROUTINE IN WHICH DEFINED: USRDAT
 5. ROUTINES IN WHICH USED: ARANGE BLDNUM
 6. DEFINITION: This is the largest value that an exponent can have on the computer being used.
-

1. NAME: MAXNUM
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: MCHDEF
 4. ROUTINE IN WHICH DEFINED: USRDAT
 5. ROUTINE IN WHICH USED: BLDNUM
 6. DEFINITION: This is the maximum number of decimal digits a number can have on the computer being used.
-

1. NAME: MMARC
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED: INTLC PARSE
ZERO
6. DEFINITION: This is the maximum number of ARCs that can be processed and stored in the NNPTAC array (the dimension of the NNPTAC array).

1. NAME: MMARY
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO4
 4. ROUTINE IN WHICH DEFINED: USRDAT
 5. ROUTINES IN WHICH USED: PARSE ZERO
 6. DEFINITION: This is the maximum number of LABEL commands that can be processed between TYPE or command level changes (the dimension of the LLARY array).
-

1. NAME: MMCNS
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO4
 4. ROUTINE IN WHICH DEFINED: USRDAT
 5. ROUTINES IN WHICH USED: LOLFND LOPSTR
NUMFND NUMSTR
ZERO
 6. DEFINITION: This is the maximum number of locations in the CCONS array that can be used to store constants (the dimension of the CCONS array).
-

1. NAME: MMCOLC
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED: COLSTR ZERO
6. DEFINITION: This is the maximum number of COLLECT variable identifiers that can be stored in the NNCOLC array (the dimension of the NNCOLC array).

1. NAME: MMFPT
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO4
 4. ROUTINE IN WHICH DEFINED: USRDAT
 5. ROUTINES IN WHICH USED: IFSTR LOPSTR
ZERO
 6. DEFINITION: This is the maximum number of nested IF and LOOP commands that can be processed between TYPE or command level changes.
-

1. NAME: MMFUNF
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO4
 4. ROUTINE IN WHICH DEFINED: ADATA
 5. ROUTINES IN WHICH USED: EEVAL FF
FNFND UNCODE
 6. DEFINITION: This is the number of defined IOSYM menu functions.
-

1. NAME: MMFUNP
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: ADATA
5. ROUTINE IN WHICH USED: PP
6. DEFINITION: This is the number of defined IOSYM probability functions.

1. NAME: MMOPR
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINE IN WHICH USED: GBLSTR
6. DEFINITION: Maximum number of substitutable parameters that can be saved in the IIOPR array. MMOPR should be set equal to the value of the first dimension of the IIOPR array.

1. NAME: MMPAR
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: DFAUT
5. ROUTINES IN WHICH USED: GGPAR PP
PPPAR
6. DEFINITION: This is the maximum parameter set number that has been set aside in the GASP array PPARM.

1. NAME: MMPL0T
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED: DDUMP NETSUM
PLTSTR PPRPLT
SSAVE ZERO
6. DEFINITION: This is the maximum number of PLOTs that can be stored in the NNPL0T array (the dimension of the NNPL0T array).

1. NAME: MMRPT
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED: RPTSTR
ZERO
6. DEFINITION: This is the maximum number of REPORT variables that can be stored in the IIRPT array (the dimension of the IIRPT array).

1. NAME: MMSET
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED:

ARCSTR	LBLSTR	SETSTR
DVSTR	LOPSTR	SRHARC
EEVAL	NODSTR	SSTWRK
ENDNET	OPRSTR	STASTR
EVSTR	PCKCK	TOFSTR
FUNSTR	PCKSTR	TRCSTR
GETSTR	PRTSTR	TRMSTR
GTOSTR	PUTSTR	TYPSTR
IFSTR	REPSTR	ZERO
JOESTR		

6. DEFINITION: This is the maximum number of locations in the NSET array available for storing information (the dimension of the NSET array).

1. NAME: MMV1
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: DFAULT
5. ROUTINES IN WHICH USED: AARCST VARFND
ZERO
6. DEFINITION: This is the maximum number of primary and derivative variables that can be processed in a NETWORK (the dimension of the SS, SSIN, DD, [OIN, and NNPT] arrays).

1. NAME: MMV2
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED: DVFND FUNC2
 FUNC4A VARFND

 ZERO
6. DEFINITION: This is the maximum number of secondary variables that can be processed in a NETWORK (the dimension of the VVAR2 and the NNPT2 arrays).

1. NAME: MMWRK
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED:

AARCST	FUNC2	NUMFND
ALPSRH	FUNC3	PARSE
BBLIST	FUNC4	RESEQ
DDOEVT	FUNC4A	RREWRK
DVFND	FUNC5	SSTWRK
EEVAL	LOLFND	WTHSTR
FNFND	NAMFND	ZERO

6. DEFINITION: This is the maximum number of locations in the NNWRK/WRK array available for storing information (the dimension of the NNWRK/WRK array).
-

1. NAME: MMXTB
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO4
4. ROUTINE IN WHICH DEFINED: USRDAT
5. ROUTINES IN WHICH USED: TBLFND

ZERO

6. DEFINITION: This is the maximum number of TABLEs that can be processed in a NETWORK including the four IOSYM TABLEs. MMXTB is the dimension of the NNPTB array.

1. NAME: NCARD
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: ME1
 4. ROUTINES IN WHICH DEFINED: SCAN
XXDATN
 5. ROUTINES IN WHICH USED: EEROR SCAN
XXDATN
 6. DEFINITION: This is the number of input cards processed by IOSYM and GASP. This does not include the READ command cards or READ continuation cards or any user input processed cards.
-

1. NAME: NDX1
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: PARSE
SCAN
5. ROUTINES IN WHICH USED: ALPSRH CHOIC2
LOLFND PARSE
PROCES SCAN
TRMSTR
6. DEFINITION: This is the number of the current command being processed. For commands with UCSs, this number is stored as the IICM. See Table 3 for number-command correspondences.

1. NAME: NDXVAL
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: PARSE
SCAN
5. ROUTINES IN WHICH USED: BLDALP BLDLDM
BLDOPR FUNFND
KWFND LIBFND
LITFND PARSE
SCAN
6. DEFINITION: This is the number of distinct fields that were processed on the last input record.

-
1. NAME: NFFJ
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: FFJ
 4. ROUTINES IN WHICH DEFINED: FFPUDJ PACWJ
 5. ROUTINE IN WHICH USED: PACWJ
 6. DEFINITION: This is the location in the LWJ array of the last non-blank character.

1. NAME: NNARC
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: PARSE ZERO
 5. ROUTINES IN WHICH USED: DDUMP ENDNET
 FUNC4A GGOTO
 IIACST NETSUM
 NODSTR SSWARC
 6. DEFINITION: This is the number of ARCs that were processed in a NETWORK. NNARC is also the last position in the NNPTAC array used.
-

1. NAME: NNARY
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: LBLSTR PARSE
 ZERO
5. ROUTINES IN WHICH USED: LBLSTR PARSE
6. DEFINITION: During the input of a NETWORK, NNARY is the number of LABELS that must be encountered to satisfy IFs and LOOPS for the current command level. When an IF or LOOP command is encountered, NNARY is incremented by one, and NNARY is decremented when the corresponding LABELs are found. When processing each END command, a check is done to see if there are any undefined LABELS (NNARY=0). If NNARY is greater than zero, then a required LABEL hasn't been found, and an error is generated.

1. NAME: NNCNS
2. LENGTH: 1
3. COMMON BLOCK LOCATION: I03
4. ROUTINES IN WHICH DEFINED:

DDUMP	NUMSTR	RESEQ
EEVAL	PARSE	TBLFND
LOLFND	PLTSTR	TBTSTR
LOPSTR	PROCES	VARFND
NUMFND	RESCC	ZERO

5. ROUTINES IN WHICH USED:

DDUMP	NUMSTR	RESEQ
EEVAL	PARSE	STASTR
LOLFND	PLTSTR	TBLFND
LOPSTR	PROCES	TBTSTR
NUMFND	RESCC	VARFND

6. DEFINITION: This is the number of constants that are stored in the CCONS array. NNCNS is also the last position in the CCONS array used.

1. NAME: NNCOL
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: COLSTR ZERO
5. ROUTINES IN WHICH USED: CCOLC COLSTR
DDUMP ENDNET
NETSUM PPRCOL
SSTWRK
6. DEFINITION: This is the number of COLLECT variables in the NETWORK. If NNCOL is less than or equal to zero, then there are no COLLECT variables, and COLLECT statistics will not be reported.

-
1. NAME: NNCOLC
 2. LENGTH: 50
 3. COMMON BLOCK LOCATION: IO1
 4. ROUTINES IN WHICH DEFINED: COLSTR ZERO
 5. ROUTINES IN WHICH USED: CCOLC COLSTR
DDUMP NETSUM
PPRCOL
 6. DEFINITION: This array contains the IOSYM IIDVs for all the variables to be collected and reported in the COLLECT statistics. If the dimension of this array is changed, then the variable MMCOLC in routine USRDAT, should be set to the value of the NNCOLC dimension. The value of the variable NNCOL is the number of IIDVs stored in the NNCOLC array.

1. NAME: NNCOP
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: COLSTR FORSTR
ZERO
 5. ROUTINES IN WHICH USED: CCOLC COLSTR
ENDNET FORSTR
NETSUM PPRCOL
SSTWFK
 6. DEFINITION: This is the number of COLLECT OPERATIONS there are in the NETWORK. If no FOR command is processed, but a COLLECT command is encountered, then NNCOP is equal to the number of OPERATIONS in the NETWORK.
-

1. NAME: NNCROS
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: AARCST EVNTS
ZERO
5. ROUTINES IN WHICH USED: AARCST EVNTS
GGSTEP SCOND
STATE
6. DEFINITION: This is the number of CROSS commands on the current ARC.

1. NAME: NNDUM
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO2
 4. ROUTINES IN WHICH DEFINED: AARCST ZERO
 5. ROUTINES IN WHICH USED: AARCST PPIDF
VVAREL
 6. DEFINITION: Number of substitutable parameters that occur on the GLOBAL and OPERATION command for the current ARC.
-

1. NAME: NNFPT
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: LBLCHK LBLSTR
ZERO
 5. ROUTINES IN WHICH USED: IFSTR LBLCHK
LBLSTR LOPSTR
TOFSTR
 6. DEFINITION: This is the number of nested IF and LOOP commands that have been processed since the last command level change.
-

1. NAME: NNGDUM
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: GBLSTR ZERO
5. ROUTINE IN WHICH USED: AARCST
6. DEFINITION: This variable indicates the number of default substitutable parameters on the GLOBAL command.

1. NAME: NNMNET
2. LENGTH: 3
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: ADATA ALPSRH
5. ROUTINES IN WHICH USED: ALPSRH DDOSIM
 NETSUM PPRCOL
 PPRPLT PPRRPT
 PPRTAB PROCES
 SAVE UPLLOT
6. DEFINITION: This array contains NETWORK name and NETWORK summary information.
 NNMNET(1). Is the first four characters of the NETWORK name in internal IOSYM representation. If NNMNET(1) is equal to zero, the NETWORK doesn't have a name.
 NNMNET(2). Is the next three characters of the NETWORK name in internal IOSYM representation.
 NNMNET(3). Indicates whether a NETWORK summary is to be printed or not. If NNMNET(3) is equal to one, then a NETWORK summary is to be printed after reading a new NETWORK or after processing a CHANGE command. If NNMNET(3) is equal to minus one, a NETWORK summary is not printed.

1. NAME: NNPLOT
 2. LENGTH: 7 x 5
 3. COMMON BLOCK LOCATION: IO1
 4. ROUTINES IN WHICH DEFINED: PLTSTR ZERO
 5. ROUTINES IN WHICH USED: DDUMP NETSUM
PLTSTR PPRPLT
SSAVE
 6. DEFINITION: This array contains the PLOT information. If the second dimension of this array is changed, then the variable MMPLT, in routine USRDAT, should be set to the value of the second NNPLT dimension. The value of the variable NNPLTS is the number of active PLOTS in the NNPLT array.

NNPLT(1,I). The number of dependent variables on PLOT I. If it is greater than zero, then only line printer plots are to be processed. If it is less than zero, then line printer PLOTS are done, and user routine UPLT is called to do any type of user plots. If equal to zero, then PLOT I is not active.

NNPLT(2,I). The IOSYM IIDV of the independent variable that defines the x-axis of PLOT I.

NNPLT(3,I), (4,I), (5,I), (6,I), (7,I). The IOSYM IIDVs of the dependent variables for the Y-axis of PLOT I.
-

1. NAME: NNPLTS
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: PLTSTR ZERO
5. ROUTINES IN WHICH USED: PLTSTR PPRPLT
SSAVE
6. DEFINITION: Number of active PLOTS in a NETWORK. If NNPLTS is less than or equal to zero, then no PLOTS will be processed in the simulation.

1. NAME: NNPTB
2. LENGTH: 3 x 50
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: TBLFND ZERO
5. ROUTINES IN WHICH USED:

AARCST	GETSTR	PPTAB
DDUMP	GGGET	PUTSTR
DVFND	GGTAB	RPTSTR
FF	NETSUM	TBLFND
FUNC2	NNODST	UNCODE
FUNC3	PP	VARFND
FUNC4A	PPPUT	

6. DEFINITION: This array contains TABLE information. If the dimension of this array is changed, then the variable MMTAB, in routine USRDAT, should be set to the value of the second NNPTB dimension. The value of the variable NNXTB is the number of TABLEs stored in the NNPTB array.

NNPTB(1,I). Is the first four characters of TABLE I's name in internal IOSYM representation.

NNPTB(2,I). Is the next three characters of TABLE I's name, plus a blank, in internal IOSYM representation.

NNPTB(3,I). Is the pointer to the location in the TTABL array where the value of the first element of TABLE I is stored.

1. NAME: NNPTGB
2. LENGTH: 5
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: ADATA GBLSTR
 TYPSTR ZERO
5. ROUTINES IN WHICH USED: BBLIST DDUMP
 PROCES TYPSTR
6. DEFINITION: This array contains the GLOBAL command information.

NNPTGB(1). The first four characters of the word "GLOBAL" in IOSYM internal representation.

NNPTGB(2). The fifth and sixth characters of the word "GLOBAL" plus two blanks in IOSYM internal representation.

NNPTGB(3). A pointer to the location in the NSET array where the TYPE S commands for the GLOBAL command start. If NNPTGB(3) is equal to zero, then there are no TYPE S commands on the GLOBAL command.

NNPTGB(4). A pointer to the location in the NSET array where the TYPE I commands for the GLOBAL command start. If NNPTGB(4) is equal to zero, then there are no TYPE I commands.

NNPTGB(5). A pointer to the location in the NSET array where the TYPE E commands for the GLOBAL command start. If NNPTGB(5) is equal to zero, then there are no TYPE E commands.

1. NAME: NNPTND
2. LENGTH: 4 x 50
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: ENDNET NODSTR
 PARSE RREPLC
 ZERO
5. ROUTINES IN WHICH USED:

ARCSTR	INTLC	PARSE
DDUMP	NETSUM	PCKCK
ENDNET	NNODST	PCKSTR
FUNC4A	NNXNOD	PPPRNT
GGOTO	NODSTR	RREPLC
6. DEFINITION: This array contains NODE information. If the dimension of this array is changed, then the variable MMNODE, in routine USRDAT, should be set to the value of the second NNPTND dimension. The value of the variable NNODE is the number of defined NODEs in the NETWORK.

NNPTND(1,I). The first four characters of NODE I's name in internal IOSYM representation. If NNPTND(1,I) is negative, then NODE I was not user defined, and it is assumed NODE I is a dummy NODE with no other commands associated with it.

NNPTND(2,I). The next three characters of NODE I's name, plus a blank, in internal IOSYM representation.

NNPTND(3,I). The pointer to the location in the NSET array where the uniform command sequence for NODE I starts. If NNPTND(3,I) is less than or equal to zero, then the NODE I has not been defined yet.

NNPTND(4,I). An indicator that tells whether NODE I has been replaced or not. If NNPTND(4,I) is zero, then NODE I has not been replaced. If NNPTND(4,I) is greater than zero, then that value is the number of the NODE that has replaced NODE I.

NOTE: The second dimension of the NNPTND array and the dimension of FIR2 in routine PARSE should be the same number.

1. NAME: NNPTO
2. LENGTH: 2
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: VARFND ZERO
5. ROUTINES IN WHICH USED: DDUMP DVFND
 FUNC2 FUNC4A
 NETSUM PPPRNT
 PPRTAB UNCODE
 VARFND
6. DEFINITION: This array contains independent variable information.
NNPTO(1). The first four characters of the independent variable's name in internal IOSYM representation. Initially, it is set to the internal representation of "TIME".
NNPTO(2). The next three characters of the independent variable's name, plus a blank, in internal IOSYM representation. Initially, it is set to the internal representation of four blanks.

1. NAME: NNPT1
2. LENGTH: 2 x 100
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: VARFND ZERO
5. ROUTINES IN WHICH USED: DDUMP DVARFD
 DVFND FUNC2
 FUNC4A NETSUM
 UNCODE VARFND
6. DEFINITION: This array contains primary variable information. If the dimension of this array is changed, then the variable MMV1, in routine DFAUT, should be set to the value of the second NNPT1 dimension. The value of the variable NNV1 is the number of primary variables in the NNPT1 array.

NNPT1(1,I). The first four characters of the primary variable I's name in internal IOSYM representation.

NNPT1(2,I). The next three characters of the primary variable I's name, plus a blank, in internal IOSYM representation.

NOTE: The second dimension of the NNPT1 array and the dimension of the SS, DD, SSIN, and the DDIN arrays should all be the same.

1. NAME: NNPT2
2. LENGTH: 2 x 150
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED:

BLOCK DATA	DVFND
FUNC2	FUNC4A
VARFND	ZERO
5. ROUTINES IN WHICH USED:

DDUMP	DVFND
FUNC2	FUNC4A
NETSUM	UNCODE
VARFND	
6. DEFINITION: This array contains secondary variable information. If the dimension of this array is changed, then the variable MMV2, in routine USRDAT, should be set to the value of the second NNPT2 dimension. The value of the variable NNV2 is the number of secondary variables in the NNPT2 array.

NNPT2(1,I). The first four characters of secondary variable I's name in internal IOSYM representation. If NNPT2(1,I) is negative, then the Ith secondary variable is not a user initialized secondary variable, i.e., was not defined on a VAR or VARIABLE card.

NNPT2(2,I). The next three characters of the secondary variable I's name, plus a blank, in internal IOSYM representation.

NOTE: The second dimension of the NNPT2 array and the first dimension of the VVAR2 array should be the same.

1. NAME: NNREAD
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: WTHSTR ZERO
 5. ROUTINES IN WHICH USED: DDOSIM WTHSTR
 6. DEFINITION: This is the number of variables specified on the WITH command that must be input with a READ command before each simulation. If NNREAD is less than or equal to zero, then there are no READ variables.
-

1. NAME: NNRPT
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINES IN WHICH DEFINED: RPTSTR ZERO
5. ROUTINES IN WHICH USED: DDUMF NETSUM
PPRRFT RPTSTR
RREPRT SSTWRK
6. DEFINITION: Number of REPORT variables specified on a REPORT command. If NNRPT is less than or equal to zero, then there are no REPORT variables, and no REPORT statistics will be processed.

1. NAME: NNSIM
2. LENGTH: 1
3. COMMON BLOCK LOCATION: IO3
4. ROUTINE IN WHICH DEFINED: DOSTR
5. ROUTINES IN WHICH USED: DDOSIM DOSTR
 VVWTMN WWTMNM
6. DEFINITION: If greater than zero, NNSIM is the number of IOSYM simulations to do as specified on the DO command. If NNSIM is less than zero, then a MINIMIZATION is being processed.

1. NAME: NNTABU
2. LENGTH: 9 x 10
3. COMMON BLOCK LOCATION: IO1
4. ROUTINES IN WHICH DEFINED: TBTSTR ZERO
5. ROUTINES IN WHICH USED: DDPRNT DDUMP
 NETSUM PPRTAB
 SSTWRK TBTSTR
 TTABU
6. DEFINITION: This array contains the TABULATE information. If the second dimension is changed, then the variable MMTABU, in routine USRDAT, should be set to the value of the second NNTABU dimension. The value of the variable NNTBS is the number of active TABULATEs in the NNTABU array.

NNTABU(1,I). If equal to minus one, then TABULATE I is not active. If equal to one, TABULATE I is active, and the time option is off. If equal to two, TABULATE I is active, and the time option is on.

NNTABU(2,I). The IOSYM IIDV of the first variable that TABULATE statistics are to be collected on.

NNTABU(3,I). The report type of the TABULATE statistics to collect on the variable in NNTABU(2,I). If NNTABU(3,I) is a one, then the statistics are to be collected at the start of the OPERATION; if equal to two, at the end of the OPERATION; if equal to three, then the change during the OPERATION is to be TABULATED.

NNTABU(4,I). Same as (2,I) except for second variable.

NNTABU(5,I). Same as (3,I) except for second variable.

NNTABU(6,I). Same as (2,I) except for third variable.

NNTABU(7,I). Same as (3,I) except for third variable.

NNTABU(8,I). Same as (2,I) except for fourth variable.

NNTABU(9,I). Same as (3,I) except for fourth variable.

1. NAME: NNTBS
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: 103
 4. ROUTINES IN WHICH DEFINED: RSAVIN SSTWRK
 5. ROUTINE IN WHICH USED: WSAVIN
 6. DEFINITION: Number of active tabulations.
-

1. NAME: NNV1
2. LENGTH: 1
3. COMMON BLOCK LOCATION: 103
4. ROUTINES IN WHICH DEFINED: RSAVIN VARFND
ZERO
5. ROUTINES IN WHICH USED: DDUMP DVFND
FUNC2 FUNC4A
GGVAR INTLC
NETSUM PPVAR
VARFND WSAVIN
6. DEFINITION: This is the number of defined primary and derivative variables in a NETWORK. NNV1 also is the last position used in the NNPT1, SSIN, and DDIN arrays.

1. NAME: NNWRK/WWRK
2. LENGTH: 2000
3. COMMON BLOCK LOCATION: I03
4. ROUTINES IN WHICH DEFINED:

AARCST		
ALPSRH	FUNC3	RESEQ
ARCSTR	FUNC4	RPTSTR
BBLIST	FUNC4A	SETSTR
DDOEVT	FUNC5	STASTR
DDOSIM	LOLFND	TBLFND
DVFND	NAMFND	TRMSTR
EEVAL	NUMFND	TTABU
FNFND	PARSE	WTHSTR
FUNC2	RESCC	ZERO

5. ROUTINES IN WHICH USED:

ARCSTR		
COLSTR	LBLSTR	RREWRK
DDOEVT	LOLFND	SAVE
DDOLST	LOPSTR	SCOND
DDOSIM	NODSTR	SETSTR
DVFND	OPRSTR	SRHARC
DVSTR	PARSE	STASTR
EEVAL	PCKSTR	STATE
EVNTS	PLTSTR	TBLFND
EVTSTR	PPRRPT	TBTSTR
FORSTR	PROCES	TRCSTR
FUNC2	PRTSTR	TRMSTR
FUNSTR	PUTSTR	TTABU
GBLSTR	REPSTR	TYPSTR
GETSTR	RESCC	VARFND
GTOSTR	RESEQ	VVWTMN
IFSTR	RPTSTR	WTHSTR
JOESTR	RREPRT	WWTMNM

6. DEFINITION: During the processing of IOSYM input the NNWRK array is used as a work storage array to store the input records processed and parsed into the different fields. If there was WITH variable information, the NNWRK array contains this information. During an IOSYM simulation the NNWRK array is used to store the REPORT statistics. During a simulation it is also used to hold pointers, variable identifiers, and other temporary storage. Array WWRK is equivalenced to the NNWRK array and is used to store and retrieve floating point numbers. If the dimension of this array is changed, then the variable MMWRK, in routine USRDAT, should be set to the value of the NNWRK dimension. The value of the variable LLXWRK is the number of consecutive locations in the NNWRK in current use during IOSYM input. During IOSYM execution, the variable LLWRK is the next available location in the NNWRK array available.

1. NAME: NNXTB
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO3
 4. ROUTINES IN WHICH DEFINED: TBLFND ZERO
 5. ROUTINES IN WHICH USED:

DDUMP	GETSTR	
DVFND	GGTAB	RPTSTR
FUNC2	NETSUM	TBLFND
FUNC3	PPTAB	UNCODE
FUNC4A	PUTSTR	VARFND
 6. DEFINITION: This is the number of TABLEs in the current NETWORK. It is initialized to four in routine ZERO to account for the four default IOSYM TABLEs: GV, NV, AV, and MV.
-

1. NAME: NOINFO
2. LENGTH: 1
3. COMMON BLOCK LOCATION: ME1
4. ROUTINES IN WHICH DEFINED: ADATA PROCES
5. ROUTINE IN WHICH USED: EEROR
6. DEFINITION: This is an informative diagnostic indicator. If NOINFO is FALSE, then informative warning messages processed by IOSYM are not printed. If NOINFO is TRUE, then the informative diagnostics along with the input line in error are printed on the output file. Fatal error diagnostics are always printed.

1. NAME: NSET
2. LENGTH: 6000
3. COMMON BLOCK LOCATION: BLANK COMMON
4. ROUTINES IN WHICH DEFINED:

ARCSTR	JOESTR	PUTSTR
DVSTR	LBLSTR	REPSTR
ENDNET	LOPSTR	RSVIN
EVTSTR	NODSTR	SETSTR
FUNSTR	OPRSTR	SRHARC
GETSTR	PCKCK	SSWARC
GTOSTR	PCKSTR	STASTR
IFSTR	PROCES	TOFSTR
INTLC	PRTSTR	TRCSTR
TRMSTR	TTAPES	TYPSTR

5. ROUTINES IN WHICH DEFINED:

AARCST	EEVAL	NETSUM
ARCSTR	ENDNET	NNODST
BBLIST	EVTSTR	NODSTR
DDOCMD	GGGET	PCKSTR
DDOEVT	GGOTO	PIFCMD
DDOLST	GCSTAT	PPPUT
DDPRNT	IIACST	PROCES
DDUMP	INTLC	SCOND
EEPRNT	LBLSTR	SRHARC
SSWARC	STATE	TRCSTR
TRMSTR	TTAPES	TTRACE
TYPSTR	WSVIN	ZERO

6. DEFINITION: This array is in BLANK COMMON and is used to store both the GASP-IV filing structure and the IOSYM NETWORK information. The first NNSET locations (NNSET is a GASP-IV variable) in the NSET array are reserved for the GASP-IV file structure. Subsequent locations hold IOSYM OPERATIONS, ARCS, NODEs, etc. This array is equivalenced to QSET which is used to store and retrieve real numbers from the NSET array. If the length of the NSET array is changed the variable MMSET in routine USRDAT should be set to the new dimension of NSET.

1. NAME: NTSFMT
2. LENGTH: 133
3. COMMON BLOCK LOCATION: ME1
4. ROUTINE IN WHICH DEFINED: BLOCK DATA
5. ROUTINE IN WHICH USED: PARSE
6. DEFINITION: This array contains the syntax for all the IOSYM commands as coded numbers. The fifth value of the KEYWORD array for each command points to the location in the NTSFMT array where the command syntax information is stored. The first element pointed to is the number of elements in the syntax for that command. This number also tells how many of the words following it belong to the syntax for that command. These words contain coded numbers explaining what type of element is to be processed for the syntax element. The following is a list of coded numbers and their definitions.

-1 = ARC name	-10 = List of variables
-2 = NODE name	-12 = List of Literals
-3 = OPERATION name	-13 = NODE or ARC name
-4 = Process TABLE	-14 = Task name
-5 = List of Numbers	-15 = Mind Set name
-6 = Variable	-16 = Alpha
-7 = Number	-17 = List of OPERATIONS
-8 = Function	-18 = LABEL name

-
1. NAME: PPREC
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO4
 4. ROUTINE IN WHICH DEFINED: USF DAT
 5. ROUTINES IN WHICH USED:

AAIROP	ARANGE
CCOLC	DDPRNT
EEVLTF	EEXP
FF	PP
TBLFND	
 6. DEFINITION: The machine precision that the computer has and should be set by the user in routine USRDAT.

1. NAME: PREC
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: ALLJ
 4. ROUTINE IN WHICH DEFINED: SETJ
 5. ROUTINES IN WHICH USED: DDRJVJ PACWJ
 PVARJ XMAJ
 ZVECJ
 6. DEFINITION: This variable is equal to the value of PPREC.
-

1. NAME: QSET
 2. LENGTH: 6000
 3. COMMON BLOCK LOCATION: BLANK COMMON
 4. ROUTINES IN WHICH DEFINED: CCOLC TRMSTR
 5. ROUTINES IN WHICH USED: CCOLC SCOND
 6. DEFINITION: This array is equivalenced to the NSET array and is used to store and retrieve real numbers from the NSET array.
-

1. NAME: SSIN
2. LENGTH: 100
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: VARFND ZERO
5. ROUTINES IN WHICH USED: GGVAR INTLC
6. DEFINITION: This array contains the initial values of the primary variables. This array is used to initialize the SS array before each run. If the dimension of this array is changed, then the variable MMV1 in routine DFAUT, should be set to the value of the SSIN dimension. The value of the variable NNVT is the number of primary variables in the SSIN array.

NOTE: The dimension of the SS, DD, SSIN, NNPT1, and DDIN arrays should all be the same.

1. NAME: TTABL
 2. LENGTH: 1000 x 2
 3. COMMON BLOCK LOCATION: IO2
 4. ROUTINES IN WHICH DEFINED: GGGET INTLC
PPIDF PPTAB
TELFND VVARBL
ZERO
 5. ROUTINES IN WHICH USED: AARCS T DDUMP
GGTAE INTLC
NETSUM NNODST
PPPUT
 6. DEFINITION: This array contains TABLE information. If the dimension of this array is changed, then the variable MMTAB, in routine USRDAT, should be set to the value of the first TTABL dimension.
- TTABL(I,1). Contains the current value of the corresponding TABLE element.
- TTABL(I,2). Contains the initial value of the corresponding TABLE element.
- NOTE: The NNPTB(J,3) array points to the location in the TTABL array where the value of the first element of TABLE J is stored. The first dimension of the TTABL array should be greater than the sum of the number of elements in all the TABLEs in a NETWORK.

1. NAME: TTIME1
 2. LENGTH: 1
 3. COMMON BLOCK LOCATION: IO2
 4. ROUTINES IN WHICH DEFINED: VARFND ZERO
 5. ROUTINES IN WHICH USED: INTLC NETSUM
 6. DEFINITION: This is the initial value of the independent variable.
-

1. NAME: VVAR2
2. LENGTH: 150 x 2
3. COMMON BLOCK LOCATION: IO2
4. ROUTINES IN WHICH DEFINED: AARCST EVNTS
 PPVAR VARFND
 ZERO
5. ROUTINES IN WHICH USED: EVNTS GGGET
 GCVAR INTLC
 PPPUT
6. DEFINITION: This array contains secondary variable information. If the dimension of this array is changed, then the variable MMV2, in routine USRDAT, should be set to the value of the first VVAR2 dimension. The value of the variable NNV2 is the number of defined secondary variables in the NETWORK.

VVAR2(I,1). The current value of the secondary variable I.

VVAR2(I,2). The initial value of the secondary variable I.

NOTE: The first dimension of the VVAR2 array and the second dimension of the NNPT2 array should be the same.

DISTRIBUTION:

Amoco Production Company
Research Center
P.O. Box 591
Tulsa, OK 74102
Attn: B. Wilder

Department of Energy (10)
Division of Geothermal Energy
Mail Station 3344
12th & Penn, NW
Federal Building
Washington, DC 20461
Attn: C. McFarland
C. Carwile (2)
M. Skalka
A. Jelacic
R. Toms
R. R. Reeber
J. Salisbury
R. Black
M. Scheve

Department of Energy (5)
Office of Oil, Gas, and Shale
Technology
Mail Station D107
Washington, DC 20545
Attn: J. Smith
M. Adams
B. Dibona
D. Duke
A. Sikri

BDM Corporation
1801 Randolph Rd SE
Albuquerque, NM 87106
Attn: A. J. Mansure

Dresser Industries, Inc.
P.O. Box 24647
Dallas, TX 75224
Attn: J. W. Langford

Dyna-Drill
P.O. Box C-19576
Irvine CA 92713
Attn: Larry Diamond

Livesay Consultants
129 Liverpool
Cardiff, CA 92007
Attn: B. J. Livesay

Halliburton
Drawer 1413
Duncan, OK 73533
Attn: Dwight Smith

Loffland Brothers Company
P.O. Box 2847
Tulsa, OK 74101
Attn: H. E. Mallory

Los Alamos National Scientific
Laboratory
Mail Stop 570
Los Alamos, NM 87545
Attn: John C. Rowley

Mobil Research and Development
Corporation
Field Research Laboratory
P.O. Box 900
Dallas, TX 75221
Attn: W. Gravley

NL Baroid Petroleum Services
City Centre Building, Suite 365W
6400 Uptown Boulevard
Albuquerque, NM 87110
Attn: Gene Polk

NL Petroleum Services
P.O. Box 1473
Houston, TX 77001
Attn: J. Fontenot

Otis
P.O. Box 34380
Dallas, TX 75243
Attn: W. D. Rumbaugh

Phillips Petroleum Company
P.O. Box 239
Salt Lake City, UT 84110
Attn: Earl Hoff

Pritsker and Associates, Inc.
P.O. Box 8345
Albuquerque, NM 87198
Attn: J. Polito

Scientific Simulation Inc.
P.O. Box 9331
Albuquerque, NM 87119
Attn: D. M. Smith (5)

Smith Tool Company
P.O. Box C-19511
Irvine, CA 92713
Attn: J. Vincent

Texas A&M University
College Station, TX 77843
Attn: M. Friedman
Dept. of Geology

Union Geothermal Division
Union Oil Company of California
Union Oil Center
Los Angeles, CA 90017
Attn: D. E. Pyle

Shell Oil Co.
Two Shell Plaza
P.O. Box 2099
Houston, TX 77001
Attn: W. E. Bingman

4700 J. H. Scott
4740 R. K. Traeger
4741 J. R. Kelsey (20)
4743 H. C. Hardee
4746 B. Granoff
4748 B. E. Bader
4750 V. L. Dugan
4751 J. R. Tillerson
4752 H. M. Dodd
4756 Y. T. Lin (5)
3141 L. J. Erickson (5)
3151 W. L. Garner (3)
For DOE/TIC (Unlimited Release)
DOE/TIC (25)
(J. Hernandez, 3154-4)