OCTOBER 1974

NASA/JSC Contract NAS 9-13779 (DRD No. MA-129T)

DESIGN ANALYSIS AND COMPUTER-AIDED PERFORMANCE EVALUATION OF SHUTTLE ORBITER ELECTRICAL POWER SYSTEM

HUGHES

HUGHES AIRCRAFT COMPANY
SPACE AND COMMUNICATIONS GROUP

FINAL REPORT

(NASA-CR-140383) DESIGN ANALYSIS AND COMPUTER-AIDED PERFORMANCE EVALUATION OF SHUTTLE ORBITER ELECTRICAL POWER SYSTEM. VOLUME 2: SYSTID USER'S GUIDE (Hughes Aircraft Co.) 91 p HC $4.75 CSCL 22B G3/18 N75-13019 Unclas 05004

OCTOBER 1974

NASA/JSC Contract NAS 9-13779 (DRD No. MA-129T)

DESIGN ANALYSIS AND COMPUTER-AIDED PERFORMANCE EVALUATION OF SHUTTLE ORBITER ELECTRICAL POWER SYSTEM

HUGHES

HUGHES AIRCRAFT COMPANY
SPACE AND COMMUNICATIONS GROUP

FINAL REPORT

VOLUME II

SYSTID USER'S GUIDE

Hughes Ref. No. D1191 • 74-085

REPORT NO. (S)^{3} 74-085

DRL MA-129T

A DESIGN ANALYSIS AND COMPUTER-AIDED PERFORMANCE EVALUATION OF THE SHUTTLE ORBITER ELECTRICAL POWER SYSTEM

FINAL CONTRACT REPORT, VOLUME II

SYSTID USER GUIDE

NAS 9-13779

OCTOBER 1974

R. J. RECHTER
STUDY PROGRAM MANAGER

M. FASHANO
ASSOC. STUDY MANAGER

HUGHES

HUGHES AIRCRAFT COMPANY

1544 CS

SYSTID

LEVEL III

USER'S GUIDE

EXEC 8, LEVEL 31

SYSTID USER;S MANUAL

PREFACE

THIS DOCUMENT DESCRIBES THE USE OF THE COMPUTER PROGRAM SYSTID UNDER THE UNIVAC EXEC8, LEVEL 31 OPERATING SYSTEM. THE VERSION OF SYSTID PROGRAM DESCRIBED IN THIS MANUAL WAS ENHANCED UNDER CONTRACT NAS9-13779, 'A DESIGN ANALYSIS AND COMPUTER-AIDED PERFORMANCE EVALUATION OF THE SHUTTLE ORBITER ELECTRICAL POWER SYSTEM'.

SYSTID WAS ORIGINALLY DEVELOPED FOR THE SIMULATION OF COMMUNICATION SYSTEMS UNDER CONTRACTS NAS9-10832, 'SYSTID - SYSTEM TIME DOMAIN SIMULATION PROGRAM' AND NAS9-11743, 'ADVANCED COMMUNICATION SYSTEM TIME DOMAIN MODELING TECHNIQUES'. THE ORIGINAL VERSIONS OPERATED UNDER THE UNIVAC EXEC 2 OPERATING SYSTEM.

THE MODELS DESCRIBED IN THIS DOCUMENT ARE THOSE WHICH WERE AVAILABLE IN THE ORIGINAL VERSIONS OF SYSTID. THE POWER SYSTEM MODEL LIBRARY DEVELOPED UNDER THE CONTRACT ARE THE SUBJECT OF ANOTHER DOCUMENT.

TABLE OF CONTENTS

1.0 PROGRAM DESCRIPTION

1. PROGRAM DESCRIPTION

SYSTID IS A SYSTEM OF COMPUTER ROUTINES WHICH PROVIDES THE ANALYST WITH A POWERFUL TOOL FOR THE TRANSIENT SIMULATION AND ANALYSIS OF COMPLEX SYSTEMS. SYSTID WAS INITIALLY DEVELOPED FOR THE SIMULATION OF COMMUNICATIONS SYSTEMS, ALTHOUGH OTHER CONTINUOUS SYSTEMS HAVE BEEN SIMULATED. A LIBRARY OF MODELS FOR POWER SYSTEMS HAS RECENTLY BEEN DEVELOPED FOR APPLICATION TO THE SPACE SHUTTLE POWER SYSTEM.

SYSTID ACCEPTS AS INPUT A TOPOLOGICAL 'BLACK-BOX' DESCRIPTION OF A SYSTEM, AUTOMATICALLY GENERATES THE APPROPRIATE ALGORITHMS, AND THEN PROCEEDS TO EXECUTE THE SIMULATION PROGRAM. THUS THE USER IS NOT NECESSARILY REQUIRED TO WRITE THE ALGORITHMS IN A COMPUTER LANGUAGE NOR POSSESS A GREAT FACILITY IN COMPUTER PROGRAMMING. THE SYSTEM DESCRIPTION, INCLUDING BOTH TOPOLOGY AND ELEMENT INFORMATION, IS SUPPLIED TO THE PROGRAM IN A FREE-FORM, USER CONTROLLED ENGINEERING LANGUAGE WHICH IS EASILY LEARNED.

SYSTID OFFERS THE USER ENORMOUS FLEXIBILITY IN THE REPRESENTATION OF SYSTEM ELEMENTS, I.E., 'BLACK BOXES'. AN ELEMENT MAY BE DEFINED AS:

(1) A SYSTID LIBRARY MODEL

(2) A USER WRITTEN, TEMPORARY SYSTID MODEL

(3) A FORTRAN ARITHMETIC EXPRESSION INVOLVING ANY INTRINSIC SYSTID PARAMETER, CONSTANT, VARIABLE, FORTRAN LIBRARY FUNCTIONS, SYSTID LIBRARY FUNCTION, MODEL NODES, AND ANY USER SUPPLIED FORTRAN FUNCTION.

IN ADDITION, FORTRAN DECLARATION AND EXECUTABLE STATEMENTS MAY BE INTERMIXED WITH THE TOPOLOGICAL PROBLEM DESCRIPTION.

THE SYSTID MODEL LIBRARY CONSISTS OF A SET OF COMPUTER ROUTINES, EITHER WRITTEN IN FORTRAN OR SYSTID, WHICH HAVE BEEN STORED ON A LIBRARY FILE AND CATALOGED IN THE SYSTID DIRECTORY. THE USER, AT ANY TIME, CAN MODIFY OR REPLACE THE LIBRARY AND DIRECTORY AS HE MAY CHOOSE--THUS EVERY USER CAN EASILY CREATE HIS OWN LIBRARY. ONE UNIQUE CHARACTERISTIC OF SYSTID IS THE CAPABI-LITY OF NESTING MODELS. THAT IS, ANY MODEL (OR SYSTEM) CAN REFERENCE OTHER MODELS OTHER THAN ITSELF. THE NESTING FEATURE PROVIDES THE USER WITH THE TOOLS NECESSARY TO BUILD A MODEL LIBRARY TO SUIT HIS NEEDS BASED UPON A CANONIC SET OF MODELS. AN EXAMPLE MIGHT BE A RECEIVER THAT IS USED IN SEVERAL SYSTEMS -- THE RECEIVER WOULD BE A MODEL CONSISTING OF A CONNECTION OF OTHER MODELS.

THE BASIC, OR CANONIC, SYSTID LIBRARY CONSISTS MAINLY OF A GROUP OF ROUTINES WHICH AID IN THE SIMULATION OF CONTINUOUS FUNCTIONS. THE TECHNIQUE APPLIED IS THAT OF THE BI-LINEAR Z-TRANSFORM REPRESENTATION OF TRANSFER FUNCTIONS. THE TRANSFER FUNCTION MAY BE DEFINED IN SEVERAL WAYS -- IN TERMS OF ITS POLES AND ZEROS, OR AS ONE OF THE CLASSICAL FUNCTIONS SUCH AS BESSEL,

ELLIPTIC, ETC. THE SAMPLE DATA ROUTINES ACCOMPLISH ALL THE NECESSARY TRANSFORMATIONS IN ADDITION TO THE NUMERICAL PROCESSING SUCH AS INTEGRATION AND DIFFERENTIATION. IN ADDITION, ALL OF THE FORTRAN ARITHMETIC FEATURES ARE AN INTRINSIC PART OF THE SYSTID LIBRARY, ALTHOUGH THEY DO NOT APPEAR IN THE LIBRARY DIRECTORY.

THE BI-LINEAR Z-TRANSFORM RATHER THAN THE STANDARD Z-TRANSFORM IS USED IN THE REPRESENTATION OF CONTINUOUS FUNCTIONS BECAUSE IT ELIMINATES ALIASING ERRORS OF THE FUNCTION, MAKING POSSIBLE THE REALIZATION OF COMMONLY ENCOUNTERED FUNCTIONS WHOSE RESPONSE DOES NOT APPROACH ZERO AT HIGH FREQUENCIES. NOTE THAT ALIASING OF THE SIGNALS, HOWEVER, IS POSSIBLE.

ANOTHER ASPECT OF THE SYSTID MODEL LIBRARY IS THAT IT CONTAINS FORTRAN SUBROUTINES -- THAT IS, WHEN A MODEL (OR SYSTEM) IS PROCESSED BY SYSTID, THE RESULT IS A FORTRAN SUBROUTINE (OR MAIN PROGRAM) WHICH IS AVAILABLE TO THE USER FOR ANY PURPOSE, WHETHER FOR A SYSTID SIMULATION OR NOT. THUS, SYSTID CAN BE VIEWED AS A FORTRAN PROGRAM GENERATOR WHICH CONVERTS A TOPOLOGI-CAL, NON-PROCEDURAL INPUT INTO A PROCEDURAL LANGUAGE, NAMELY FORTRAN. ALTHOUGH NOT UNIQUE TO SYSTID, THIS ASPECT ALLOWS ONE TO EVALUATE MATHEMATICAL PROBLEMS VIA SYSTID WITH NO CONCERN FOR THE INPUT-OUTPUT CODING NECESSARY IN MOST FORTRAN PROGRAMS. THAT IS, SYSTID MAY THUS BE VIEWED AS A SHORT HAND FORTRAN SYSTEM.

SYSTID'S FLEXIBILITY IS IN PART ATTAINED BY DESIGNING THE

PROGRAM TO EXECUTE AS A MULTIPASS PROCESSOR IN A BATCH OR DEMAND MODE OF OPERATION. THE FIRST PHASE READS THE USER INPUT DESCRIPTION OF ALL MODELS AND/OR A SYSTEM AND PROCEEDS TO FORMULATE THE CORRESPONDING FORTRAN ALGORITHMS. IN THIS PHASE, THE PROGRAM CHECKS FOR INPUT ERRORS SUCH AS ERRONEOUS MODEL REFERENCES, TYPOS,ETC, IN WHICH CASE APPROPRIATE ERROR MESSAGES ARE ISSUED. IF THE FIRST PHASE TERMINATES WITHOUT FATAL ERRORS, THE FORTRAN ROUTINES ARE AUTOMATICALLY COMPILED AND COLLECTED WITH THE SYSTID LIBRARY TO FORM THE SECOND PHASE, THAT OF EXECUTING THE SIMULATION.

OUTPUT FROM THE PROGRAM INCLUDES PLOTS AS WELL AS TABULATED DATA. CONVENTIONAL OUTPUT IS ANY SYSTEM NODE OR VARIABLE WHICH MAY BE INDIVIDUALLY SELECTED. GRAPHIC OUTPUT CONSISTS OF BOTH PRINTER AND CALCOMP PLOTS, FORMATTED UNDER USER CONTROL FOR EITHER 8-1/2 INCH PAGES OR THE FULL 14 INCH PAGE. IN ADDITION, AN INTERACTIVE GRAPHICS PROGRAM FOR ACCESSING SYSTID SAVE FILES HAS BEEN DEVELOPED FOR USE WITH THE JSC MOPS INTERACTIVE TERMINAL.

THE ADDITIONAL FLEXIBILITY OF LINKING TO A USER DEFINED POST PROCESSING ROUTINE IS INTRINSIC TO SYSTID WHEN UTILIZING THE POST SYSTEM IDENTIFIER. THIS FEATURE ALLOWS THE USER TO ACCESS THE TIME HISTORIES OF ANY NODE OR VARIABLE MUCH THE SAME WAY AS THE PLOT ROUTINES. UTILITY ROUTINES ARE ALSO AVAILABLE TO PERFORM ANY NECESSARY INPUT AND OUTPUT FOR THE USER.

THE USER, BECAUSE OF THE TWO PHASE ASPECT OF SYSTID, HAS AVAILABLE TO HIM SEVERAL TECHNIQUES FOR CONTROLLING HIS COMPUTER RUNS AND ENSURING THAT THE MOST EFFECTIVE USE IS MADE OF COMPUTER TIME. ONE METHOD IS THAT OF SAVING THE RESULTS OF THE FIRST PHASE, THAT IS, THE COLLECTED SIMULATION PACKAGE FOR SUBSEQUENT RERUNS WITH ALTERNATE INPUT DATA. RERUN WOULD THEN SIMPLY ENTAIL A LOAD-GO OPERATION. THE ALTERNATE INPUT DATA CAN BE PROVIDED AT EXECUTION TIME BY THE USE OF THE DATA IDENTIFIED IN THE FIRST PHASE. ANOTHER EFFECTIVE USE OF MACHINE TIME IS USING THE CHECKPOINT FEATURE OF UNIVAC EXEC8 . LINKAGE TO THE CHECKPOINT PROCESSOR IS AVAILABLE WITHIN SYSTID.

2.0 DATA PREPARATION

2. DATA PREPARATION

GIVEN A DESCRIPTION OF A SYSTEM OR MODEL IN AN ENGINEERING ORIENTED LANGUAGE, SYSTID WILL AUTOMATICALLY GENERATE THE FORTRAN CODE REQUIRED TO SIMULATE THE SYSTEM OR MODEL. THE USER, HOWEVER, MUST REDUCE THE SYSTEM OR MODEL TO AN EQUIVALENT BLOCK DIAGRAM FORM CONSISTING OF MODEL REFERENCES, MATH EXPRESSIONS, ETC WHICH CAN BE INTERPRETED BY SYSTID. THE TRANSCRIPTION OF THE EQUIVALENT BLOCK DIAGRAM INTO THE INPUT LANGUAGE IS STRAIGHTFORWARD AND EASILY MASTERED.

THE INITAL STEP IN USING SYSTID IS TO PREPARE THE EQUIVALENT BLOCK DIAGRAM OF THE PROBLEM TOPOLOGY UTILIZING THE AVAILABLE MODELS AND FORTRAN EXPRESSIONS. THE STANDARD MODEL LIBRARY DIRECTORY IS GIVEN IN SECTION 2.5. THE USER IS NOT IN ANY WAY RESTRICTED OR LIMITED TO THIS LIBRARY -- ANY MODEL WITHIN THE LIBRARY CAN BE REPLACED WITH ONE'S OWN MODEL. THUS, THE USER'S MODEL REPETOIRE CONSISTS OF:

1) THE INVOKED SYSTID LIBRARY DIRECTORY DEFINING THE AVAILABLE MODELS.

2) FORTRAN MATH EXPRESSIONS, INCLUDING VARIABLES, CONSTANTS, NODES, ETC.

3) ANY STSTID MODEL DEFINED IN THE SAME RUN STREAM.

4) FORTRAN EXPRESSIONS INTERMIXED WITH THE TOPO- LOGICAL DESCRIPTION.

ONCE THE USER HAS SPECIFIED THE EQUIVALENT BLOCK DIAGRAM OF HIS SYSTEM OR MODEL, NODE NAMES AND VARIABLES MUST BE ASSIGNED WHERE APPROPRIATE. IN MANY INSTANCES, IT IS DESIRABLE TO MODULARIZE A LARGE SYSTEM INTO MODELS TO SIMPLIFY THE DATA PREPARATION AND DEBUGGING.

SECTION 3 PRESENTS THE DETAILS OF THE INPUT LANGUAGE AND METHODS OF PREPARING THE DESCRIPTION OF THE SYSTEM OR MODEL.

2.1. RULES AND ASSUMPTIONS

ANY COMPUTER LANGUAGE IS SUBJECT TO SOME SET OF RULES AND ASSUMPTIONS DEFINED AT THE TIME OF CONCEPTION AND IMPLEMENTATION. THE BASIC ASSUMPTION MADE IN THE DEVELOPMENT OF SYSTID IS THAT THE USER POSSESSES A NOMINAL AMOUNT OF COMMON SENSE.

THE LIMITED NUMBER OF RULES AND ASSUMPTIONS THAT MUST BE ADHERED TO IN THE PREPARATION OF A PROBLEM DESCRIPTION ARE SUMMARIZED BELOW:

1) NODE NAMES MUST BE NO MORE THAN SIX ALPHANUMERIC CHARACTERS.

2) ALL VARIABLE NAMES AND EXPRESSIONS MUST BE FORTRAN COMPATABLE, INCLUDING INTEGER, FLOATING POINT, COMPLEX, ETC CONVENTIONS.

3) INTERNAL MODEL NODES ARE NOT AVAILABLE EXTERNAL TO THE MODEL.

4) THE VALUE OF THE SIGNAL AT A NODE IS THE SUM OF THE OUTPUT OF ALL MODELS CONNECTED TO THE NODE.

5) A MODEL MAY NOT REFERENCE ITSELF, BUT MAY REFERENCE OTHER MODELS.

6) SYSTID WILL VERIFY THE NUMBER OF ARGUMENTS TO A MODEL REFERENCE, HOWEVER IT WILL NOT VERIFY THE TYPE OF ARGUMENT.

2.2. INTRINSIC SYSTID VARIABLES

THERE ARE SEVERAL INTERNAL VARIABLES USED WITHIN SYSTID INORDER TO EXECUTE THE SIMULATION. THESE VARIABLES ARE AVAILABLE FOR USE IN ANY EXPRESSION. CAUTION MUST BE USED TO PREVENT AN INADVERTANT DESTRUCTION OF THESE VARIABLES. THE VARIABLES AND THEIR USAGE ARE:

TIME OR T	THE TIME AS KEPT BY THE SIMULATION CLOCK (UNRELATED TO ACTUAL COMPUTER RUN TIME)
TSTART	THE SIMULATION START TIME. THAT IS, THE TIME BIAS FOR OUTPUT LABELING.
TSTOP	SIMULATION STOP TIME
SETTLE	SETTLING TIME BEFORE OUTPUTTING
DT	SIMULATION SAMPLING TIME INCREMENT
S	DENOTES THE SIGNAL VALUE OF THE FIRST INPUT NODE OF A TOPOLOGY STATEMENT
Z+1	ABSOLUTE ADDRESS OF THE FIRST REAL DATA CELL AVAILABLE TO THE MODEL (V(Z+1))
ZW+1	ABSOLUTE ADDRESS OF THE FIRST

	COMPLEX DATA CELL AVAILABLE TO THE MODEL (W(ZW+1))
ZZ	ABSOLUTE ADDRESS OF THE LAST DATA CELL USED BY THE MODEL (SUM OF REAL AND 2*COMPLEX VALUES)
V(.,)	DYNAMIC STORAGE ARRAY FOR REAL NODES AND VARIABLES
W(.,)	DYNAMIC STORAGE ARRAY FOR COMPLEX NODES AND VARIABLES
PI	3.14159
TWOPI	6.28318
NPRINT	PRINT OUTPUT EDIT INTERVAL,THAT IS, PRINT EVERY NPRINT SOLUTIONS

2.3. COMPLEX ENVELOPE SIMULATIONS

WHEN DEALING WITH MODULATED COMMUNICATION SYSTEMS, IT IS HIGHLY DESIRABLE TO SIMULATE THE COMPLEX ENVELOPE OF THE SYSTEM, ASSUMING THE SIGNAL IS ANALYTIC. THE ANALYTIC ASSUMPTION WILL BE TRUE IF THE BASEBAND (OR MODULATION) SIGNAL SPECTRUM IS ESSENTIOALLY ZERO AT THE CARRIER FREQUENCY. IF NOT, A RIPPLE IN THE SIMULATION OUTPUT AT ROUGHLY TWICE THE CARRIER FREQUENCY IS INTRODUCED -- THE CASE IF THE SIGNAL IS A SQUARE WAVE, FOR INSTANCE. IN ANY EVENT, THE RIPPLE IS NORMALLY NEGLIGIBLE DUE TO THE LARGE RATIO BETWEEN BASEBAND AND CARRIER FREQUENCIES.

THE IMPLEMENTATION OF COMPLEX ENVELOPE SIMULATION WAS ACCOMPLISHED IN THE PREVIOUS VERSIONS OF THE STANDARD SYSTID LIBRARY.

DOCUMENTATION OF THE METHODOLOGY CAN BE FOUND IN THE REFERENCES. THE NEW VERSION OF SYSTID, LEVEL III, THRU IMPLEMENTATION OF COMPLEX NODES,ETC GREATLY SIMPLIFIES THE USER'S INTERFACE WHEN DEALING IN THE COMPLEX TIME DOMAIN.

2.4. MISCELLANEOUS COMMENTS

.... OPEN FOR CONTRIBUTIONS OR COMPLAINTS .....

2.5. SYSTID MODEL DIRECTORY

THE USER MAY MODIFY OR REPLACE THE SYSTID MODEL DIRECTORY AND LIBRARY AT WILL, OR CREATE ANY NUMBER OF SUPPLEMENTAL LIBRARIES AS NECESSARY. THIS TASK IS ACCOMPLISHED UTILIZING THE UNIVAC EXEC8 FILE HANDLING AND EDITING CAPABILITIES IN EITHER THE DEMAND OR BATCH ENVIRONMENT. THE STANDARD DIRECTORY GIVEN BELOW CONTAINS THE NECESSARY INSTRUCTIONS FOR MODIFYING THE DIREC- TORY.

THE MODEL DIRECTORY SERVES SEVERAL PURPOSES:

1) CROSS REFERENCES MODEL NAMES AND SUBROUTINE ENTRY POINTS.

2) FLAGS THE MODEL AS A FORTRAN SUBROUTINE OR FUNCTION.

3) DEFINES THE NUMBER OF ARGUMENTS TO THE MODEL.

4) DEFINES THE NUMBER AND TYPE OF INPUT AND OUTPUT NODES.

WHENEVER A MODEL IS PROCESSED BY SYSTID, AN ENTRY POINT NAME IS ASSIGNED TO IT AND ANY SUBSEQUENT MODELS IN THE RUN STREAM. THE ASSIGNMENT IS MADE BY ALPHABETICALLY INCREMENTING THE LEAST SIGNIFICANT CHARACTER OF THE DIRECTORY ENTRY IN LINE 24. THAT IS, IF LINE 24 IS ** MODEL , THEN THE FIRST TEMPORARY MODEL WOULD HAVE ENTRY MODELA, THE SECOND MODELB, ETC. IF LINE 24 WAS MODELX, THEN THE FIRST MODEL WOULD BE MODELY, THE SECOND MODELZ, THE THIRD MODEMA, ETC. IN ADDITION, THE APPROPRIATE ENTRIES ARE TEMPORILY MADE IN THE DIRECTORY FOR DEFINING THE MODEL FOR THE PARTICULAR RUN ONLY. AN EXAMPLE IS GIVEN IN SECTION 6, WHERE THE DIRECTORY ENTRY IS THE FIRST LINE OF OUTPUT. IF IT WERE DESIRABLE TO ADD THE MODEL TO THE PERMANENT LIBRARY, ONE OF TWO PATHS IS AVAILABLE:

1) RENAME THE ENTRY POINT BY RECOMPILING AND UPDATING THE FORTRAN SUBROUTINE AND ENTER THE APPROPRIATE DATA INTO THE DIRECTORY.

2) ADD THE SYSTID GENERATED DIRECTORY CARD INTO THE DIRECTORY AND CHANGE LINE 24 TO REFLECT THE HIGHEST MODEL NAME IN THE LIBRARY.

CERTAINLY THE LATTER IS MUCH EASIER, BUT MAY LEAD TO FUTURE CONFUSION.

```
1
2 THIS ELEMENT IS THE MODEL LIBARY DIRECTORY FOR THE SYSTID PROCESSOR.
3 ADDITIONS AND DELETIONS CAN BE MADE SIMPLY BY REMOVING OR ENTERING THE
4 DESCRIPTOR CARD.  ALL INTEGERS MUST BE RIGHT JUSTIFIED.
*
5          FORMAT
*
6     A    CC  1-36 MODEL NAME, ALPHANUMERIC, LEFT ADJUST AND NO EMBEDDED
7                          BLANKS
8     B    CC 37-42 THE ENTRY POINT NAME CORRESPONDING TO (A)
9     C    CC    45 'F' FOR FUNCTION, ' ' OR 'S' FOR SUBROUTINE
10    D    CC 47-48 THE NUMBER OF ARGUEMENTS REQUIRED FOR (B)
11    E    CC 53-54 THE NUMBER OF INPUTS TO THE MODEL
12    F    CC 56-57 THE NUMBER OF OUTPUTS FROM THE MODEL
13    G    CC 61-72 AN OCTAL CONSTANT REPRESENTING THE TYPE OF EACH
14                  INPUT & OUTPUT NODE.  IF ALL THESE NODES ARE REAL
15                  THIS FIELD CAN BE LEFT BLANK.  OTHERWISE, THE LEFT-
16                  MOST BIT OF THIS WORD SHOULD BE SET TO 1 IF THE FIRST
17                  INPUT NODE IS COMPLEX, THE 2ND BIT FOR THE NEXT
17                  INPUT NODE ETC.  THE FIRST OUTPUT NODE FOLLOWS THE
18                  LAST INPUT NODE.  A MODEL WITH 1 INPUT NODE & 1 OUTPUT
19                  NODE WOULD NEED A 6 IN COLUMN 61 IF BOTH NODES WERE
20                  COMPLEX.  A MODEL WITH 4 REAL INPUT NODES & 4 COMPLEX
21                  OUTPUT NODES WOULD REQUIRE 036 IN CC 61,62,63
*
*
**    MODEL             THIS CARD STARTS THE LIBRARY AND ENTRY POINTS
FILTER                                    FILTER    9     1  1
GENRAL                                    GENRAL    9     1  1
GENERALFILTER                             GENRAL    9     1  1
BUTWTH                                    BUTWTH    6     1  1
BUTTERWORTH                               BUTWTH    6     1  1
BUFUNCTION                                BUTWTH    6     1  1
BESSEL                                    BESSEL    6     1  1
BEFUNCTION                                BESSEL    6     1  1
BUTOM                                     BUTOM     7     1  1
BUTTERWORTHTHOMPSON                       BUTOM     7     1  1
BTFUNCTION                                BUTOM     7     1  1
ELIPTC                                    ELIPTC    8     1  1
ELLIPTIC                                  ELIPTC    8     1  1
ELFUNCTION                                ELIPTC    8     1  1
QFACT                                     QFACT     7     1  1
QFACTOR                                   QFACT     7     1  1
QUADRATICFACTOR                           QFACT     7     1  1
LEADLAG                                   LEDLAG    5     1  1
LOOPFILTER                                LEDLAG    5     1  1
LEADFUNCTION                              LEADIT    4     1  1
CHEBY                                     CHEBY     7     1  1
CHEBYCHEV                                 CHEBY     7     1  1
TCHEBYCHEV                                CHEBY     7     1  1
```

AGATE	AGATE		2	1	1	
AMMODULATOR	AMMOD		2	1	1	
AMMOD	AMMOD		2	1	1	
AMDEM	AMDEM		0	1	1	
AMPLITUDEDEMODULATOR	AMDEM		0	1	1	
AMPLITUDEDEMODULATORSQUARELAW	AMDESQ		2	1	1	
AMDESQ	AMDESQ		2	1	1	
CMULT	CMULT		2	1	1	6
COMPLEXMULTIPLIER	CMULT		2	1	1	6
COMPLEXADDER	CADD		2	1	1	6
CADD	CADD		2	1	1	6
COSINE	COSINE	F	1			
DELTAMODULATOR	DELMOD		2	1	1	
DELMOD	DELMOD		2	1	1	
DIFFERENTIATOR	DIFFER		0	1	1	
DIF	DIFFER		0	1	1	
DIFFER	DIFFER		0	1	1	
FMMODULATOR	FMMOD		2	1	1	
FMMOD	FMMOD		2	1	1	
FREQUENCYDEMODULATOR	FMDEMM		2	1	1	
FMDEMM	FMDEMM		2	1	1	
FMDEM	FMDEMM		2	1	1	
FMDEMOD	FMDEMM		2	1	1	
FMDCTD	FMDEMM		2	1	1	
GNOISE	GNOISE	F	3			
GNOIS2	GNOIS2	F	2			
HARDLIMITER	HARD		0	1	1	
HARD	HARD		0	1	1	
INTEGRATOR	INTGRT		0	1	1	
INTGRT	INTGRT		0	1	1	
INTEGRALWITHINITIALCONDITIONS	INTGIC		1	1	1	
INTGIC	INTGIC		1	1	1	
MATCHEDFILTER	MFLTER		1	1	1	
MFLTER	MFLTER		1	1	1	
MONOSTABLE	MONO		1	1	1	
MONO	MONO		1	1	1	
MULTILEVELPCM	MLTPCM		2	1	1	
MLTPCM	MLTPCM		2	1	1	
NRZLBITSTREAM	NRZL		3	1	1	
NRZL	NRZL		3	1	1	
PHASEMODULATOR	PMMODD		2	1	1	
PMMOD	PMMODD		2	1	1	
PMMODD	PMMODD		2	1	1	
PHASEDEMODULATOR	PMDEMM		2	1	1	
PMDEMM	PMDEMM		2	1	1	
PHASESHIFTER	PHSHFT		2	1	1	
PHSHFT	PHSHFT		2	1	1	
PERIODICTABLEFUNCTION	PTABLE	F	10			
PTABLE	PTABLE	F	10			
PULSE	PULSE	F	5			

RANDOMBITGENERATOR	RBGEN		1	1	1	
RBGEN	RBGEN		1	1	1	
RFSOFT	RFSOFT		2	1	1	6
RFSOFTLIMITER	RFSOFT		2	1	1	6
RFLIMITER	RFLIMT		0	1	1	6
RFLIMT	RFLIMT		0	1	1	6
RFSHFT	RFSHFT		2	1	1	6
RFPHASESHIFTER	RFSHFT		2	1	1	6
RFPHAS	RFPHAS		0	1	1	4
RFPHASE	RFPHAS		0	1	1	4
RFPDEM	RFPDEM		1	1	1	4
RFPHASEDEMODULATOR	RFPDEM		1	1	1	4
RPMMOD	RPMMOD		1	1	1	2
RFPHASEMODULATOR	RPMMOD		1	1	1	2
RFFREQ	RFFREQ		0	1	1	4
RFREQUENCY	RFFREQ		0	1	1	4
RFMDEM	RFMDEM		1	1	1	4
RFFREQUENCYDEMODULATOR	RFMDEM		1	1	1	4
RFMMOD	RFMMOD		1	1	1	2
RFFMMODULATOR	RFMMOD		1	1	1	2
RFENVE	RFENVE		1	1	1	4
RFENVELOPE	RFENVE		1	1	1	4
RAMDEM	RAMDEM		1	1	1	4
RFAMDEMODULATOR	RAMDEM		1	1	1	4
RAMMOD	RAMMOD		3	1	1	2
RFAMPLITUDEMODULATOR	RAMMOD		3	1	1	2
SINE	SINE	F	1			
SOFTLIMITER	SOFTY		2	1	1	
SOFTY	SOFTY		2	1	1	
SPLIT	SPLIT		0	1	1	2
SQ	SQ		1	1	1	
SQUAREWAVE	SQ		1	1	1	
SQUAREWAVEFREQUENCYMODULATOR	SQFMOD		2	1	1	
SQFMMOD	SQFMOD		2	1	1	
SQUAREWAVEPHASEMODULATOR	SQPMOD		2	1	1	
SQPMOD	SQPMOD		2	1	1	
TABLE	TABLE	F	11			
TABL2	TABL2	F	11			
ZRODET	ZRODET		0	1	1	
ZEROCROSSINGDETECTOR	ZRODET		0	1	1	
ATOD	ATOD		4	1	1	
DTOA	DTOA		4	1	1	
SHDTOA	SHDTOA		4	1	1	
TWT	TWT		3	1	1	
SIN	SIN	F	1			
COS	COS	F	1			
PNPULS	PNPULS	F	4			
RCPULS	RCPULS	F	5			
V	V	F	1			
COMPLEXSET	COMSET		2	1	1	

```
COMGET                                COMGET      0      1  1
COMSET                                COMSET      2      1  1
RCCHANNEL                             RCHAN       7      1  1
PBEESTIMATOR                          ESTMAT     12      1  1
AVERAGE                               AVRAGE      0      1  1
RPULSE                                RPULSE  F   2
ESTMAT                                ESTMAT     12      1  1
$$$$$$
```

3.0 INPUT LANGUAGE

3. INPUT LANGUAGE

3.1. GENERAL STATEMENT, COMMENTS, & CONTINUATIONS

THE SYSTID INPUT LANGUAGE CONSISTS OF DESCRIPTIVE STATEMENTS CONSTRUCTED LARGELY FROM USER DEFINED NAMES AND SPECIFICATIONS. SEVERAL KEY VARIABLES ARE USED, AS DEFINED EARLIER IN SECTION 2-2, IN ADDITION TO NAMES WHICH REFERENCE DEFINED LIBRARY MODELS. THE STATEMENTS CAN BE PUNCHED IN A COMPLETELY FREEFIELD DATA CARD (COLUMNS 1-80). SINCE STATEMENTS ARE SCANNED FROM BOTH THE LEFT AND THE RIGHT, FIELD DELIMITERS MAY, IN GENERAL, BE ANY NON-ALPHANUMERIC CHARACTER. A STATEMENT MAY HAVE CONTINUATION CARDS, WHICH ARE DEFINED BY A NON-ALPHANUMERIC IN COLUMN 1. (OBVIOUSLY, A 7-8 PUNCH MAY NOT BE USED AS A CONTINUATION CHARACTER). A CARD WHICH HAS A NON-ALPHANUMERIC IN COLUMN 2 OR LATER AS THE FIRST NON-BLANK CHARACTER IS TREATED AS A COMMENT. COMMENTS MAY APPEAR ANYWHERE EXCEPT WITHIN A CONTINUED STATEMENT. COMMENTS MAY BE CONTINUED. WITH THE EXCEPTION OF COMMENTS, THE TITLE ON A SYSTEM CARD, AND FORTRAN STATEMENTS, BLANKS ARE IGNORED BY SYSTID.

3.2. COMMANDS, TOPOLOGY, AND HIERARCHY

THERE ARE TWO TYPES OF SYSTID STATEMENTS; I.E. COMMANDS AND TOPOLOGY STATEMENTS. WITH THE EXCEPTION OF THE END CARD, ALL COMMANDS MUST APPEAR AT THE BEGINNING OF A MODEL OR SYSTEM, AND ALL TOPOLOGY STATEMENTS MUST FOLLOW ALL THE COMMANDS.

THE FOLLOWING LIST SHOWS THE ORDER IN WHICH SYSTID STATEMENTS MUST BE PLACED. STATEMENTS IN THE SAME COLUMN MAY BE INTERMIXED WITH EACH OTHER IN ANY ORDER. FOR EXAMPLE, TOPOLOGY STATEMENTS AND EXECUTABLE FORTRAN MAY BE COMPLETELY INTERMIXED, BUT THEY MUST FOLLOW ALL OTHER CARDS IN THE SYSTEM OR MODEL EXCEPT THE END CARD.

```
SYSTEM
   COMPLEX
   DATA
   DEFAULT
   DEFINE
   FORTRAN (NON-EXECUTABLE)
   IMPLICIT COMPLEX
   IMPLICIT REAL
   PAGE
   PLOT
   POST
   PPLOT
   PRINT
   REAL
   SAVE
      SET
      VARY
         TOPOLOGY STATEMENTS
         FORTRAN (EXECUTABLE)
            END
```

```
MODEL
   DEFINE
   FORTRAN (NON-EXECUTABLE)
      SET
         TOPOLOGY STATEMENTS
         FORTRAN (EXECUTABLE)
            END
```

3.3. COMMANDS

THE SYSTID COMMANDS ARE PRIMARILY CONCERNED WITH INITIALIZING AND CONTROLLING THE EXECUTION OF A SYSTID SIMULATION. THE SYSTID COMMAND KEY WORD IS FOLLOWED BY A DELIMITER, WHICH IS FOLLOWED BY AN EXPRESSION, A NODE LIST, OR OTHER DATA RELEVANT TO THE COMMAND.

3.3.1. SYSTEM & MODEL DECLARATION:

3.3.1.1. SYSTEM: TITLE

THIS IDENTIFIER INSTRUCTS SYSTID TO EXPECT OTHER COMMANDS DEALING WITH INPUT/OUTPUT, ETC., AND TO GENERATE A MAIN PROGRAM. THE TITLE SERVES TO LABEL ALL OUTPUT (36 CHARACTERS), E.G.:

SYSTEM : TRY ME SOMETIME

3.3.1.2. MODEL : IN1=...=INK= NAME(ARG1,...,ARGN)=OUT1=...=OUTJ

THIS IDENTIFIER INSTRUCTS SYSTID TO EXPECT A MODEL DEFINITION (MOST COMMANDS ARE ILLEGAL IN A MODEL) AND TO GENERATE A SUBROUTINE. THE MODEL NAME MUST BE NO MORE THAN 36 CHARACTERS.

THE PARENTHESIS AND COMMAS ARE REQUIRED. THE DELIMITERS, (':' AND '=' USED IN ABOVE EXAMPLE), SHOULD NOT BE EITHER LEFT OR RIGHT PARENS TO AVOID CONFUSION. TO AVOID AMBIGUITY IN THE CASE WHERE THE MODEL NAME IS LESS THAN SEVEN CHARACTERS AND THERE ARE NO ARGUEMENTS TO THE MODEL, THE FOLLOWING FORMS MAY BE USED :

... = NAME() = ...

OR ... = (NAME) = ...

EXAMPLES:

```
MODEL : X, UHF RECEIVER , Y
MODEL * INPUT = FM MODULATOR(BETA,FC) = OUTPUT
MODEL * IN1,IN2,IN3,IN4 = ADD4(ARG) = RESULT
```

3.3.1.3. END : COMMENT

SIGNIFIES THE END OF A MODEL OR SYSTEM DESCRIPTION.

3.3.2. NODE TYPING

ALL NODES IN A SYSTID SIMULATION ARE TYPED EITHER REAL OR COMPLEX (REAL IS DEFAULT). SEE THE SECTION ON INTERMIXING FORTRAN FOR THE TYPE CONVENTIONS OF VARIABLES WHICH ARE NOT NODES.

3.3.2.1. IMPLICIT REAL

ASSUME ALL NODES IN THE SIMULATION ARE REAL UNLESS DECLARED OTHERWISE. THIS IS THE DEFAULT CONDITION.

3.3.2.2. COMPLEX : NAME1, NAME2, . . . , NAMEN

THE N NODES, NAME1 THRU NAMEN, ARE TYPED COMPLEX. CONTRAST THE SYSTID REAL AND COMPLEX STATEMENTS WITH THE CORRESPONDING FORTRAN STATEMENTS, WHICH HAVE NO DELIMITER FOLLOWING THE KEY-WORD, AND ARE USED TO TYPE FORTRAN VARIABLES.

3.3.2.3. IMPLICIT COMPLEX

ASSUME ALL NODES IN THE SIMULATION ARE

COMPLEX UNLESS DECLARED OTHERWISE.

3.3.2.4. REAL : NAME1, NAME2, . . . , NAMEN

THE N NODES, NAME1 THRU NAMEN, ARE TYPED REAL. SEE THE COMPLEX COMMAND. THIS STATEMENT IS REQUIRED TO HAVE A REAL NODE IF THE IMPLICIT COMPLEX STATEMENT IS USED.

3.3.3. I/O AND POST-PROCESSING

SYSTID POST-PROCESSING COMMANDS (PRINT, PLOT, PPLOT, POST, SAVE) ARE AN EXCEPTION TO THE GENERAL SYSTID RULES OF USING COMPLEX NODES, IN THAT A NODE NAME REFERENCES EITHER A REAL NODE OR THE REAL PART OF A COMPLEX NODE. IF THE IMAGINARY PART OF A COMPLEX NODE IS TO BE SAVED FOR POST-PROCESSING, IT MUST BE PRECEEDED BY A DELIMITER. THE $ IS RECOMMENDED TO CONFORM TO THE NOTATION USED ELSEWHERE IN THIS TEXT. FOR EXAMPLE:

```
COMPLEX * X,Y,CT
PRINT * X,$Y,CT,$CT
```

WILL PRINT THE REAL PART OF X, THE IMAGINARY PART OF Y, AND BOTH THE REAL AND IMAGINARY PARTS OF CT.

3.3.3.1. PAGE : COMMENT

THIS COMMAND, MERELY BY BEING PRESENT, CAUSES ALL PRINTED OUTPUT TO BE 8-1/2 BY 11 INCH COMPATIBLE.

3.3.3.2. PLOT : NAME1, NAME2, . . . , NAMEN

THIS COMMAND IS SIMILAR TO PPLOT, THE EXCEPTION BEING THAT HERE THE CALCOMP (OR SC4060) PLOTTER IS UTILIZED, E.G.,

PLOT: X,NODE4,OUTPUT

3.3.3.3. POST : SUBROUTINE NAME, NAME1,NAME2, . . . ,NAMEN

THIS COMMAND CAUSES A POST-PROCESSING ROUTINE NAMED 'SUBROUTINE NAME' TO BE CALLED FOLLOWING THE SIMULATION. THIS ROUTINE IS CALLED FOR EACH OCCURENCE OF NAMEI, WHICH MAY BE NODES OR VARIABLES. THE CALL SEQUENCE GENERATED IS SIMILAR TO THAT OF THE PLOT REQUESTS, THE ONLY DIFFERENCE BEING THE ENTRY POINT. UTILITY ROUTINES ARE AVAILABLE FOR INTERFACING A USER POST-PROCESSOR WITH THE DATA TIME HISTORIES WHICH ARE STORED ON TEMPORARY FILES. SEE ALSO THE SECTION ON WRITTING FORTRAN POST-PROCESSING ROUTINES.

3.3.3.4. PPLOT : NAME1,NAME2, . . . ,NAMEN

THIS COMMAND DEFINES THE DATA TO BE PRINTER PLOTTER FOLLOWING THE SIMULATION. THE QUANTITIES MAY BE NODES OR VARIABLES, E.G.,

PPLOT : NODES1,B, SCOMP

3.3.3.5. PRINT : NAME1,NAME2, . . . ,NAMEN

THIS IDENTIFIER DEFINES THE DATA TO BE PRINTED DURING THE SIMULATION. BOTH NODES AND VARIABLES MAY BE PRINTED. NOTE THAT TIME IS AUTOMATICALLY PRINTED - IT NEED NOT BE REQUESTED. SEE ALSO THE PAGE COMMAND AND THE INTRISIC SYSTID VARIABLE 'NPRINT'.

3.3.3.6. SAVE : QUAL : NAME1,NAME2, . . . ,NAMEN

THE TIME HISTORIES OF THE N NODES, NAME1 THRU NAMEN, ARE SAVED ON SEPARATE FILES FOR LATER EXAMINATION OR PROCESSING. QUAL MAY BE UP TO FIVE ALPHANUMERIC CHARACTERS, THE FIRST OF WHICH MUST BE A LETTER. THE NAMES OF THE

FILES ON WHICH THE TIME HISTORY VALUES ARE SAVED ARE FORMED FROM THIS QUALIFIER AND THE NODE NAMES

3.3.4. DATA INITIALIZATION AND RUN CONTROL

3.3.4.1. DATA : VAR1,VAR2, . . . ,VARN

THIS COMMAND NAMES A LIST OF VARIABLES WHICH CAN BE READ IN AT SIMULATION TIME UNDER NAMELIST SYSTID. WHENEVER THIS STATEMENT APPEARS, A NAMELIST DATA CARD MUST APPEAR AT EXECUTION TIME. THE NUMBER OF VARIABLES IS LIMITED TO 20. THE VARIABLES MUST CONFORM TO FORTRAN REQUIREMENTS, AND MAY BE ANY INTRINSIC SYSTID VARIABLE, E.G.,

DATA = TSTOP,DT,NPRINT,MEDIA,SAM,A

THE PHASE 2 INPUT DATA FOR THIS EXAMPLE WOULD BE, FOR EXAMPLE:

$SYSTID DT=1.5E-6, TSTOP=10., . . . $END

WHICH IS STANDARD FORTRAN NAMELIST INPUT. NOTE: THIS STATEMENT SHOULD NOT BE CONFUSED WITH THE FORTRAN DATA STATEMENT.

3.3.4.2. DEFAULT : VAR1=CONST, . . . ,VARN=CONST

ALTERNATELY, DEFAUL : . . .

THIS COMMAND SERVES TO LOAD DEFAULT VALUES FOR ANY VARIABLE IN THE SIMULATION, INCLUDING ANY INTRINSIC SYSTID VARIABLE. THE CONSTANT VALUES MUST CONFORM TO THE FORTRAN RULES OF INTEGER , FLOATING POINT, AND COMPLEX VARIABLES OR ERRORS MAY OCCUR. THE NUMBER OF ENTRIES IS LIMITED TO 25.

DEFAULT: DT=1.5E-6,TSTOP=2.0, A=10.,IOU=3

3.3.4.3. DEFINE : VARIABLE=FORTRAN EXPRESSION

THIS COMMAND GENERATES A FORTRAN 'DEFINE' STATEMENT. 'VARIABLE' MUST BE A LEGAL

FORTRAN NAME. WHEREVER THIS NAME APPEARS IN THE GENERATED FORTRAN PROGRAM, THE FORTRAN EXPRESSION IS CALCULATED USING THE CURRENT VALUES OF ANY VARIABLE WHICH MAY APPEAR IN THE EXPRESSION, AND THIS RESULT IS USED FOR THE VALUE OF 'VARIABLE'.

NOTE: A FORTRAN DEFINE STATEMENT MAY BE USED DIRECTLY BY OMITTING THE DELIMITER.

3.3.4.4. SET : VARIABLE = FORTRAN EXPRESSION

THIS COMMAND GENERATES A FORTRAN ASSIGNMENT STATEMENT WHICH SETS 'VARIABLE' EQUAL TO THE VALUE OF 'FORTRAN EXPRESSION'.

NOTE: THE SAME AFFECT CAN BE HAD WITH A FORTRAN ASSIGNMENT STATEMENT. HOWEVER, A SET STATEMENT DOES NOT CAUSE SYSTID TO BEGIN SCANNING FOR TOPOLOGY STATEMENTS, AS AN EXECU-TABLE FORTRAN STATEMENT WOULD. ALSO, THE CODE GENERATED BY A SET STATEMENT IS OUTSIDE THE RANGE OF THE SIMULATION LOOP, WHEREAS THE FORTRAN STATEMENT WOULD BE INSIDE THE LOOP.

3.3.4.5. VARY : VARIABLE=MIN,MAX,DELTA

THIS COMMAND GENERATES A FORTRAN DO-LOOP WHICH HAS WITHIN ITS RANGE ANY VARY OR SET STATEMENTS WHICH FOLLOW IT IN THE SYSTID DECK, AS WELL AS THE SIMULATION AND POST-PROCESSING CALLS. IF THE 'VARIABLE' NAME IS TYPE INTEGER (FIRST LETTER IS I THRU N OR Z), THE PARAMETERS ARE EXPECTED TO BE OF INTEGER TYPE. OTHERWISE, THE VARIABLE NAME AND PARAMETERS CAN BE REAL OR INTEGER.

3.4. TOPOLOGY STATEMENTS

THERE ARE TWO KINDS OF TOPOLOGY STATEMENTS IN SYSTID, I.E., EXPRESSIONS AND MODEL REFERENCES.

3.4.1. EXPRESSIONS

EXPRESSIONS HAVE THE FORM:

LNF : EXPRESSION : RNF

WHERE

LNF IS THE LEFT NODE FIELD

RNF IS THE RIGHT NODE FIELD

: REPRESENTS A DELIMITER WHICH MAY BE ANY NON-ALPHANUMERIC

EXPRESSION IS AN EXPRESSION COMPOSED OF NODE NAMES, VARIABLES, FUNCTION REFERENCES, OPERATORS, ETC.

THE EXPRESSION IS EDITED FOR SYSTID FUNCTIONS AND SHORTHAND CONVENTIONS (SEE 'ACCESS TO COMPLEX NODES' FOR AN EXAMPLE) AND THE FOLLOWING FORTRAN LINE IS GENERATED.

RNF = EDITED EXPRESSION

3.4.2. MODEL REFERENCES

A MODEL IS REFERENCED IN A MANNER SIMILAR TO THE METHOD OF DECLARING A MODEL:

IN1,...- INK < MODEL NAME(ARG1,...,ARGN) > - OUT1,...- OUTJ

WHERE

IN1-INK ARE THE K INPUT NODES (AT LEAST 1)

MODEL NAME IS THE NAME OF THE MODEL REFERENCED

ARG1-ARGN ARE THE N MODEL PARAMETERS

OUT1-OUTJ ARE THE J OUTPUT NODES (AT LEAST 1)

THE PARENTHESIS AND COMMAS IN THE MODEL NAME ARGUEMENT LIST ARE REQUIRED. THE OTHER DELIMITERS MAY BE ANY NON-ALPHANUMERIC CHARACTER, HOWEVER, TO AVOID CONFUSION THEY SHOULD NOT BE A '(' OR ')'. TO AVOID AMBIGUITY IN THE CASE WHERE THE MODEL NAME IS LESS THAN SEVEN CHARCTERS AND THERE ARE NO ARGUMENTS TO THE MODEL, THE FOLLOWING FORM MAY BE USED:

```
.... - NAME () - ...
```

EXAMPLE: A MODEL TO ADD FOUR NODES TOGETHER AND MULTIPLY THE RESULT BY A CONSTANT:

```
MODEL * IN1,IN2,IN3,IN4 * ADD4(ARG) * RESULT
IN1 < S + IN2 + IN3 + IN4 > TEMP
TEMP < TEMP * ARG > RESULT
END
```

A TYPICAL REFERENCE FROM ANOTHER MODEL OR SYSTEM MIGHT BE:

```
NODE1-NODE2-NODE3-NODE4 < ADD4(2.718) > NODE10
```

3.4.3. ACCESSING NODES IN EXPRESSIONS

SYSTID TOPOLOGY STATEMENTS USE THE COMPLEX VALUE OF A COMPLEX NODE, AND ACCEPT THE FUNCTIONS REAL AND AIMAG. ALSO, THE TOPOLOGY STATEMENTS ACCEPT THE FOLLOWING SHORTHAND NOTATION:

```
#NODE => REAL(NODE)
$NODE => AIMAG(NODE)
$$NODE => #NODE => REAL(NODE)
$ => VALUE OF THE FIRST INPUT NODE
```

FOR EXAMPLE:

```
COMPLEX * X,Y,R,S
X < $ + #R + $S > Y
```

WILL GENERATE: Y=X + REAL(R) + AIMAG(S)

3.5. INTERMIXING FORTRAN STATEMENTS

ALL SYSTID STATEMENTS ARE EITHER COMMANDS, TOPOLOGY STA- TEMNTS, OR THE END CARD. ALL COMMANDS (POST,DEFAULT,ETC) MUST APPEAR AT THE BEGINNING OF THE MODEL (OR SYSTEM) AND ALL TOPOLOGY STATEMENTS FOLLOW THE COMMANDS. EXECUTABLE FORTRAN STATEMENTS AND LABELLED FORMAT STATEMENTS MAY BE INTERMIXED WITH

SYSTID TOPOLOGY STATEMENTS. NON-EXECUTABLE FORTRAN STATEMENTS WITH THE EXCEPTION OF THE FORMAT STATEMENT MAY BE INTERMIXED WITH THE SYSTID COMMANDS.

3.5.1. EXECUTABLE FORTRAN AND FORMAT STATEMENTS

EXECUTABLE FORTRAN STATEMENTS MAY BE INTERMIXED WITH SYSTID TOPOLOGY STATEMENTS IN ANY OF THREE WAYS:

FORM1 =>

```
                FORTRAN - ...FORTRAN STATEMENT...

EXAMPLE:          FORTRAN * XX = NODE1
                  FORTRAN * WRITE(6,300)TIME,XX
```

FORM2 (SHORTHAND) =>

```
                  4TRAN - ...FORTRAN STATEMENT

EXAMPLE:          4TRAN - IF(TIME.GT.1.) NODE5=0
```

NOTE: THERE CANNOT BE ANY BLANKS BETWEEN THE '4' AND THE 'T' IN 4TRAN.

FORM3 (LABELLED STATEMENT) =>

```
                LABEL ....FORTRAN STATEMENT...

EXAMPLE:          300 FORMAT(' ',2F10.5)
```

NOTE: THERE CANNOT BE ANY IMBEDDED BLANKS IN THE LABEL.
A BLANK MUST FOLLOW THE LABEL.

3.5.2. NON-EXECUTABLE FORTRAN (EXCEPT FORMAT)

NON-EXECUTABLE FORTRAN STATEMENTS MAY BE INTERMIXED WITH THE SYSTID COMMANDS, WITH THE EXCEPTION OF FORMAT AND IMPLICIT STATEMENTS. FOR EXAMPLE:

```
SYSTEM * USE FORTRAN
DEFAULT * TSTOP=1.,DT=.01
DIMENSION TABLE(5)/1.,2.,6.,24.,120./
PRINT * N1,N2
          .
          .
          .
          .
          .
```

3.5.3. VARIABLE NAMES AND TYPE CONVENTIONS

VARIABLES MAY BE OF ANY TYPE DEPENDING ON THE FIRST LETTER OF THEIR NAME AND ANY FORTRAN TYPE STATEMENTS WHICH THE USER MAY WISH TO INCLUDE. THE FORTRAN IMPLICIT STATEMENT IS ILLEGAL IN SYSTID ALTHOUGH SPECIFIC TYPE STATEMENTS MAY BE USED (E.G. INTEGER,ETC). IF FORTRAN TYPE STATEMENTS ARE NOT USED, THE FOLLOWING RULES APPLY:

A-H,O-V,X,Y ARE REAL

I-N,Z ARE INTEGER

W IS COMPLEX

THE USER SHOULD BE CAUTIOUS IN USING VARIABLES WHICH BEGIN WITH THE LETTERS V,W, AND Z AS A CONFLICT WITH INTERNAL VARIABLES MAY RESULT. THE USER IS ALSO CAUTIONED TO CHECK THE TYPES OF VARIABLES USED AS ARGUMENTS PASSED TO MODELS OR IN MODEL DEFINITIONS.

IF A NODE NAME IS TYPED COMPLEX, ANY FORTRAN STATEMENT REFERENCING THE NODE WILL USE THE COMPLEX VALUE. THE FORTRAN INTRINSIC FUNCTIONS REAL AND AIMAG MUST BE USED TO REFERENCE THE REAL OR IMAGINARY PARTS.

3.6. WRITTING FORTRAN POST-PROCESSING SUBROUTINES

3.6.1. INTERFACING WITH SYSTID

LINKAGE TO ANY USER POST-PROCESSING ROUTINE IS AVAILABLE THRU USE OF THE SYSTID COMMAND 'POST'. WHEN A REFERENCE TO POST IS MADE IN ANY SYSTEM DESCRIPTION, A SUBROUTINE CALL IS MADE TO THE USER'S PROGRAM FOR EVERY SPECIFIED VARIABLE IN THE FORM:

```
CALL NAME(LABEL,NPAGE)
```

WHERE

LABEL IS THE HOLLERITH NAME OF THE VARIABLE.

NPAGE CONVEYS THE OUTPUT SIZING, I.E., NPAGE=4 IS 8.5 X 11 COMPATIBLE OUTPUT, NPAGE=7 IS STANDARD COMPUTER SIZE OUTPUT.

FOR EXAMPLE, THE SYSTID STATEMENT:

```
POST - SPECTM - NODE1 - OUTPUT
```

WOULD GENERATE THE FOLLOWING TWO LINES IN THE MAIN SIMULATION PROGRAM:

```
CALL SPECTM('NODE1',7)
CALL SPECTM('OUTPUT',7)
```

THE FIRST LINE OF THE USER WRITTEN SUBROUTINE SPECTM WOULD GENERALLY BE AS FOLLOWS:

```
SUBROUTINE SPECTM(LABEL,NPAGE)
```

TWO INPUT PARAMETERS ARE REQUIRED, EVEN IF THE USER IS NOT CONCERNED WITH OUTPUT SIZING.

WHENEVER A POST STATEMENT APPEARS IN A SYSTID PROGRAM, MAIN/SYSTID CONTAINS THE FOLLOWING COMMON BLOCK (ALL VARIABLES WHICH BEGIN WITH Z ARE INTEGERS):

```
COMMON /DRMHED/ ZDATE(2), ZTOD(2), TSTART,
TSTOP, VEQDT, SETTLE, ZUSED, QUAL, ZNAMES,
ZNAME(<ZNAMES>)
```

WHERE

```
ZDATE  = BCD OF DATE, LEFT JUSTIFIED, BLANK
         FILLED
ZTOD   = BCD OF TIME OF DAY, LEFT JUSTIFIED,
         BLANK FILLED
TSTOP  = STOP TIME OF SIMULATION
TSTART = START TIME OF SIMULATION
VEQDT  = DT (WHICH APPEARS IN /COGENT/)
```

```
SETTLE = SETTLE TIME OF SIMULATION
ZUSED  = NUMBER  OF  VALUES  FOR  EACH VARIABLE
         STORED  ON  TEMPORARY  FILE  (IN  SOME
         ROUTINES NAMED NVAL)
QUAL   = I/O QUALIFIER (SEE SAVE STATEMENT)
ZNAMES = NUMBER  OF  DIFFERENT VARIABLES STORED
         ON FILES.
ZNAME(I), I=1,<ZNAMES> = NAMES  OF  VARIABLES
         STORED ON FILES.
```

INCLUSION OF THIS COMMON BLOCK IN A POST-PROCESSING ROUTINE ALLOWS THE USER ACCESS TO SEVERAL IMPORTANT VARIABLES, PARTICULARLY NVAL (ZUSED) THE NUMBER OF VALUES AVAILABLE FOR ANY PARTICULAR VARIABLE STORED ON FILE.

3.6.2. I/O (RETRIEVAL OF TIME HISTORIES)

ALL RETRIEVAL OF INFORMATION STORED ON I/O FILES SHOULD BE ACCOMPLISHED WITH THE TWO SUBROUTINES DRMGET AND DRMEDT.

DRMGET: CALLING SEQUENCE:

```
CALL DRMGET(NAME,ARRAY,LENGTH,MSKIP)
```

WHERE

NAME CONTAINS NAME OF VARIABLE REFERENCED (BCD)

ARRAY IS ARRAY INTO WHICH VALUES ARE TO BE READ (DIMENSIONED AT LEAST BY LENGTH)

LENGTH IS NUMBER OF VALUES TO BE READ

MSKIP IS NUMBER OF WORDS TO SKIP BEFORE READING

CONTINUING THE EXAMPLE, THE FOLLOWING LINES OF

CODE RETRIEVE ALL VALUES STORED ON FILE FOR A PARTICULAR VARIABLE, 1000 VALUES AT A TIME .

EXAMPLE 1:

```
          SUBROUTINE SPECTM (LABEL,NPAGE)
          COMMON /DRMHED/ DUMMY(8), NVAL
          DIMENSION ARRAY(1000)
              .
              .
              .
          DO 100 MSKIP = 0,NVAL,1000
          CALL DRMGET(LABEL,ARRAY,1000,MSKIP)
              .
              .
              .
                ANALYSIS OF THE 1000 VALUES OF THE
                VARIABLE NAMED IN LABEL.
              .
              .
              .
      100 CONTINUE
```

DRMEDT: CALLING SEQUENCE:

```
          CALL DRMEDT(NAME,NREQ,ARRAY)
```

WHERE

NAME CONTAINS THE NAME OF THE VARIABLE REFERENCED (BCD).

NREQ EQUALS THE NUMBER OF VARIABLES TO BE READ OFF FILE.

ARRAY IS ARRAY INTO WHICH VARIABLES ARE PLACED (DIMENSIONED AT LEAST BY NREQ).

LET NVAL BE THE NINTH WORD OF COMMON BLOCK /DRMHED/, NVAL EQUALS THE NUMBER OF VALUES STORED ON FILE FOR EACH VARIABLE, IF

NREQ>=NVAL, THEN NVAL WORDS (ALL THAT ARE ON THE FILE) WILL BE READ INTO ARRAY. IF NREQ<NVAL, THE DATA ON FILE IS EDITED AND NREQ EQUALLY SPACED VALUES ARE READ INTO ARRAY.

EXAMPLE 2:

```
SUBROUTINE SPECTM(LABEL,NPAGE)
DIMENSION ARRAY (1000)
    .
    .
    .
CALL DRMEDT (LABEL,1000,ARRAY)
     ARRAY   NOW   CONTAINS   1000   EQUALLY
     SPACED  VALUES  OF  THE  VARIABLE NAMED IN
     LABEL.
```

VALUES OF TIME ARE NOT STORED ON FILE, BUT ARE COMPUTED BY DRMGET AND DRMEDT WHEN NEEDED, SAVING I/O TIME.

IN EXAMPLE 1, ASSUME THE ARRAY TIME IS DIMENSIONED BY 1000. THEN THE STATEMENT:

```
CALL DRMGET('TIME',TIME,1000,MSKIP)
```

PLACED INSIDE THE DO-LOOP WILL COMPUTE THE 1000 VALUES OF TIME CORRESPONDING TO THE VALUES IN ARRAY.

SIMILARLY, IN EXAMPLE 2, THE STATEMENT:

```
CALL DRMEDT('TIME',1000,TIME)
```

WOULD PRODUCE 1000 VALUES OF TIME IN THE ARRAY TIME SUCH

THAT ARRAY(I) WAS THE VALUE OF THE VARIABLE NAMED IN LABEL AT TIME TIME(I).

IN THE ABOVE EXAMPLES, VALUES ARE RETRIEVED FROM STORAGE FILES IN BLOCKS OF 1000. I/O TIME CAN BE REDUCED IF THE BLOCKSIZE IS CHOSEN TO BE AN INTEGER NUMBER OF SECTORS (A MULTIPLE OF 28).

3.6.3. CORE MANAGEMENT

NORMALLY, LARGE ARRAYS USED IN POST-PROCESSING ROUTINES WOULD BE RESIDENT IN CORE DURING THE ENTIRE SIMULATION, INCREASING CORE CHARGES. THIS WOULD BE THE CASE IN THE ABOVE EXAMPLES. THIS CAN BE AVOIDED, HOWEVER, IF THE ARRAYS ARE PLACED IN THE /VSPACE/ COMMON BLOCK AND THE DYNAMIC CORE ALLOCATION SUBROUTINE IS USED.

EXAMPLE:

```
      COMMON /VSPACE/ ARRAY0, ARRAY(2)
         .
         .
         .
      CALL CORE (1000)
```

WILL EXPAND CORE TO PROVIDE 1000 WORDS FOR ARRAY.

4.0 PROBLEM SUBMISSION

SYSTID USER'S MANUAL

1. PROGRAM DESCRIPTION

SYSTID IS A SYSTEM OF COMPUTER ROUTINES WHICH PROVIDES THE ANALYST WITH A POWERFUL TOOL FOR THE TRANSIENT SIMULATION AND ANALYSIS OF COMPLEX SYSTEMS. SYSTID WAS INITIALLY DEVELOPED FOR THE SIMULATION OF COMMUNICATIONS SYSTEMS, ALTHOUGH OTHER CONTINUOUS SYSTEMS HAVE BEEN SIMULATED. A LIBRARY OF MODELS FOR POWER SYSTEMS HAS RECENTLY BEEN DEVELOPED FOR APPLICATION TO THE SPACE SHUTTLE POWER SYSTEM.

SYSTID ACCEPTS AS INPUT A TOPOLOGICAL 'BLACK-BOX' DESCRIPTION OF A SYSTEM, AUTOMATICALLY GENERATES THE APPROPRIATE ALGORITHMS, AND THEN PROCEEDS TO EXECUTE THE SIMULATION PROGRAM. THUS THE USER IS NOT NECESSARILY REQUIRED TO WRITE THE ALGORITHMS IN A COMPUTER LANGUAGE NOR POSSESS A GREAT FACILITY IN COMPUTER PROGRAMMING. THE SYSTEM DESCRIPTION, INCLUDING BOTH TOPOLOGY AND ELEMENT INFORMATION, IS SUPPLIED TO THE PROGRAM IN A FREE-FORM, USER CONTROLLED ENGINEERING LANGUAGE WHICH IS EASILY LEARNED.

SYSTID OFFERS THE USER ENORMOUS FLEXIBILITY IN THE REPRESENTATION OF SYSTEM ELEMENTS, I.E., 'BLACK BOXES'. AN ELEMENT MAY BE DEFINED AS:

(1) A SYSTID LIBRARY MODEL

(2) A USER WRITTEN, TEMPORARY SYSTID MODEL

4. PROBLEM SUBMISSION

SYSTID, WHICH OPERATES IN THE EXEC8 TIMESHARING/BATCH ENVIRONEMENT MAY BE EXECUTED BY THREE DIFFERENT PROCESSES:

(1) BY SUBMISSION OF A DATA DECK TO THE CLOSED SHOP

(2) STARTING A JOB FROM A REMOTE DEMAND TERMINAL VIA THE @START FILENAME CONTROL CARD

(3) DIRECT EXECUTION FROM A REMOTE DEMAND TERMINAL

IN ANY EVENT, THE EXECUTIVE CONTROL LANGUAGE IS BASICALLY THE SAME. EXTENSIVE USE OF THE EXEC8 @ADD AND @SETC CONTROL CARDS PROVIDE THE USER WITH THE ABILITY TO CONFIGURE ANY SYSTID RUN FOR THE FOLLOWING:

(1) USE OF AN ALTERNATE LIBRARY DIRECTORY (IN TPFS)

(2) PHASE I ONLY

(3) INHIBIT FORTRAN PROGRAM LISTING IF IN ERROR MODE

(4) PHASE II ONLY

IN ADDITION TO THE ABOVE DIRECT CONTROLS, SYSTID WILL AUTOMATICALLY INHIBIT PHASE II EXECUTION IF DATA INPUT ERRORS HAVE BEEN DETECTED, IN WHICH CASE THE FORTRAN ROUTINES AS WRITTEN BY PHASE I WILL BE LISTED (UNLESS INHIBITED BY ONE OF THE @SETC

OPTIONS). THE FIRST EXAMPLE LISTS THE STANDARD SYSTID RUN STREAM WITH ALL THE OPTIONS. PRIOR TO THE @FIN CARD, TPF$ CONTAINS THE COLLECTED SIMULATION PROGRAM NAMED SIMULATE, IN ADDITION TO THE SYSTID GENERATED FORTRAN MODELS IN BOTH SOURCE AND RELOCATABLE FORM.

ALL OF THE FOLLOWING EXAMPLES ARE SET UP FOR BATCH OR @START COMMAND EXECUTION. IF A RUN IS TO BE EXECUTED DIRECTLY FROM DEMAND, THE ADD ELEMENT SYSTID-ABS.RUNIT MUST BE REPLACED BY SYSTID-ABS.DEMAND AND THE @NORMAL: CARD IS NOT REQUIRED. IF ANY PHASE II DATA IS REQUIRED BY THE PROGRAM (SUCH AS A $SYSTID NAMELIST), THE LAST TWO LINES OF ELEMENT DEMAND MUST BE DELETED BEFORE IT IS USED.

EXAMPLE WITH ALL OPTIONS

```
CARD                                                NOTE
----                                                ----

@RUN . . .                                            1
@SETC 1/S3                                            2   +
@SETC 1/S4                                            3   *
@SETC 2/S4                                            4
@SETC 3/S4                                            5   *
@ADD,P SYSTID-ABS.SYSTID                              6
     .                                                7
     .                                                8
     .                                                9
     <PHASE I DATA>                                  10
     .                                               11
     .                                               12
     .                                               13
@ADD,P SYSTID-ABS.RUNIT                              14
 $SYSTID  . . .                                      15
     .                                               16
     .                                               17
     .                                               18
     < PHASE II DATA >                               19
     .                                               20
     .                                               21
     .                                               22
@NORMAL:                                             23
     .                                               24
     .                                               25
     .                                               26
     < OPTIONAL FILE MANIPULATIONS >                 27
     .                                               28
     .                                               29
     .                                               30
@FIN                                                 31
```

+ REQUIRES A DIRECTORY IN TPF$
* THESE TWO CARDS ARE MUTUALLY EXCLUSIVE

NOTES FOR EXAMPLE 1

1 STANDARD EXEC8 RUN CARD

2 INDICATES AN ALTERNATE DIRECTORY IS BEING USED (IN TPF$)

3 PHASE I ONLY IS TO BE EXECUTED (CREATES A SIMULATION PROGRAM BUT DOES NOT EXECUTE IT, SEE LINE 5)

4 ELIMINATES FORTRAN LISTING IF ERRORS ARE ENCOUNTERED

5 PHASE II ONLY IS TO BE EXECUTED (EXECUTE THE PROGRAM GENERATED BY A PREVIOUS RUN)

6 ADDS A PROCEDURE (SYSTID) TO COMPILE SYSTID LANGUAGE STATEMENTS

7-13 PHASE I INPUT, I.E., SYSTID LANGUAGE STATEMENTS DEFINING MODELS AND/OR A SYSTEM

14 ADDS A PROCEDURE (RUNIT) TO COMPILE, MAP, AND/OR EXECUTE THE SIMULATION PROGRAM

15-22 NAMELIST DATA (REQUIRED IF A SYSTID DATA STATEMENT WAS USED), FOLLOWED BY ANY DATA WHICH MAY BE READ IN BY MODELS

23 OPTIONAL, INSURES RUNIT PROCEDURE TERMINATES NORMALLY

24-30 OPTIONAL OPERATIONS TO SAVE SYSTID GENERATED PROGRAMS AND/OR FILES

31 STANDARD EXEC8 END-OF-JOB CARD

TYPICAL EXAMPLE WITH NO OPTIONS:

```
CARD                                              NOTE
----                                              ----

@RUN . . .                                          1
@ADD,P SYSTID=ABS.SYSTID                            2
     .                                              3
     .                                              4
     .                                              5
     < PHASE I DATA >                               6
     .                                              7
     .                                              8
     .                                              9
@ADD,P SYSTID=ABS.RUNIT                            10
     .                                             11
     .                                             12
     .                                             13
     < PHASE II DATA >                             14
     .                                             15
     .                                             16
     .                                             17
@NORMAL:                                           18
@FIN                                               19
```

EXAMPLE WITH ALTERNATE LIBRARY DIRECTORY:

```
CARD                                              NOTE
----                                              ----

@RUN . . .                                          1
@SETC 1/S3                                          2
@COPY,S EXAMPLE.DIRECTORY                           3
@ADD,P SYSTID-ABS.SYSTID                            4
     .                                              5
     .                                              6
     .                                              7
     < PHASE I DATA >                               8
     .                                              9
     .                                             10
     .                                             11
@ADD,P SYSTID-ABS.RUNIT                            12
     .                                             13
     .                                             14
     .                                             15
< PHASE II DATA >                                  16
     .                                             17
     .                                             18
     .                                             19
@NORMAL:                                           20
@FIN                                               21
```

NOTES FOR EXAMPLE 3

2 INDICATES AN ALTERNATE DIRECTORY IS TO BE USED

3 IN THIS CASE, THE ALTERNATE DIRECTORY IS ELEMENT DIRECTORY IN FILE EXAMPLE

EXAMPLE WITH PHASE I ONLY:

```
CARD                                    NOTE
----                                    ----

@RUN . . .                                1
@SETC 1/S4                                2
@ADD,P SYSTID-ABS,SYSTID                  3
     .                                    4
     .                                    5
     .                                    6
     < PHASE I DATA >                     7
     .                                    8
     .                                    9
     .                                   10
@ADD,P SYSTID-ABS,RUNIT                  11
@NORMAL:                                 12
@FIN                                     13
```

NOTES FOR EXAMPLE 4

2 INDICATES PHASE I ONLY

FILE MANIPULATION STATEMENTS COULD BE PLACED BETWEEN LINES 11 AND 12 TO SAVE THE OUTPUT FROM THE FIRST PHASE.

EXAMPLE WITH PHASE II ONLY:

```
CARD                                    NOTE
----                                    ----

@RUN . . .                                1
@SETC 3/S4                                2
@COPY,A EXAMPLE.SIMULATE                  3
@ADD,P SYSTID-ABS.RUNIT                   4
     .                                    5
     .                                    6
     .                                    7
     < PHASE II DATA >                    8
     .                                    9
     .                                   10
     .                                   11
@NORMAL:                                 12
@FIN                                     13
```

NOTES FOR EXAMPLE 5

2 PHASE II ONLY (EXECUTE SAVED SIMULATION)

3 IN THIS CASE, THE ABSOLUTE ELEMENT GENERATED BY A PREVIOUS PHASE I IS NAMED **SIMULATE** AND IT IS STORED IN FILE EXAMPLE

THE DETAILED EXECUTIVE CONTROL CARD SEQUENCE REPRESENTED BY THE ELEMENTS SYSTID-ABS.SYSTID, SYSTID-ABS.RUNIT, AND SYSTID-ABS.DEMAND ARE GIVEN BELOW FOR REFERENCE.

SYSTID-ABS.SYSTID

```
@ASG,T ELTFIL
@USE 9,ELTFIL
@ASG,T 3,,D/3000//5000
@TEST TE/1/S3               . USER DIRECTORY (IF TRUE) IN TPF$
@COPY,S SYSTID-ABS.DIRECTORY      . BRING IN FIRST PHASE
@XQT SYSTID-ABS.SYSTID
@ADD,P DIRECTORY            . LOAD YOUR DATA NOW
```

SYSTID-ABS.DEMAND

```
@FREE 3.
@COPY,S SYSTID-ABS.PROCS/SYSTID
@ADD ELTFIL.
@ELT,I GOSIM
 IN MAIN/SYSTID
 LIB TPF$.,SYSTID-RLIB.
 LIB SYSTID-CLIB$.
 IN VSPACE
@MAP GOSIM,TPF$.COB
@FREE OUTPUT.
@ASG,T OUTPUT.
@BRKPT PRINT$/OUTPUT
@TPF$.COB
@BRKPT PRINT$
@MSG,N OUTPUT IS IN FILE OUTPUT.
```

SYSTID-ABS.RUNIT

```
@FREE 3
@TEST TNE/3/S4
@JUMP GO1
@TEST TNE/2/T3         . SKIP IF ERROR IN PASS1
@JUMP ERRORS
@COPY,S SYSTID-ABS.PROCS/SYSTID
@ADD ELTFIL.           . HERE COME THE FORTRAN....
@TEST TNE/1/S4
@JUMP NORMAL
@MSG,N    CONTINUE INTO PASS 2
@GO1:
@TEST TNE/3/S4
@JUMP GO2
@MAP,I GOSIM,SIMULATE
IN MAIN/SYSTID
LIB TPF$.,SYSTID-RLIB.
LIB SYSTID-CLIB$.
LIB ISD*RPLOT.
@JUMP GO2
@ERRORS:
@MSG,N ERRORS IN PROCESSING YOUR INPUT DATA
@TEST TE/0/S4
@JUMP NORMAL
@DATA,L ELTFIL.
@END
@JUMP NORMAL
@GO2:
@XQT SIMULATE
```

5. BASIC & POWER SYSTEM SYSTID LIBRARIES

5.1. THE BASIC SYSTID LIBRARY

THE FOLLOWING IS A BRIEF SUMMARY OF THE MODELS AVAILABLE IN THE BASIC SYSTID LIBRARY. FOR A MORE COMPLETE LIST, SEE SECTION 2.5. FOR A COMPLETE DESCRIPTION OF EACH MODEL, AND HOW IT IS USED, SEE 'ADVANCED COMMUNICATION SYSTEM TIME DOMAIN MODELING TECHNIQUES, ASYSTD SOFTWARE DESCRIPTION, VOLUME 1, PROGRAM USERS GUIDE' APPENDIX B. (R72-001, CONTRACT NAS9-11743, AUGUST 1972).

SIGNAL GENERATORS

- GAUSSIAN NOISE
- PULSE GENERATOR
- SQUARE WAVE GENERATOR
- TABLE GENERATORS
- PERIODIC TABLE GENERATORS
- TRANSCENDENTAL FUNCTION GENERATORS

MODULATORS

- AMPLITUDE MODULATORS
- FREQUENCY MODULATORS (SINE WAVE)
- FREQUENCY MODULATOR (SQUARE WAVE)
- PHASE MODULATORS (SINE WAVE)
- PHASE MODULATOR (SQUARE WAVE)
- DELTA MODULATOR

DEMODULATORS

- AMPLITUDE DEMODULATORS
- PHASE DEMODULATORS
- FREQUENCY DEMODULATORS
- FREQUENCY DEMODULATOR WITH FEEDBACK

FILTERS

- GENERAL FILTER MODEL
- BUTTERWORTH
- CHEBYCHEV
- BESSEL
- BUTTERWORTH-THOMPSON
- ELLIPTIC
- LEAD LAG FUNCTION
- LEAD FUNCTION
- MATCHED FILTER

LIMITERS

- SOFT LIMITERS
- HARD LIMITERS
- RF SOFT LIMITER
- RF HARD LIMITER

TRANSFORMS

- FOURIER TRANSFORM (FFT AND INVERSE)
- HAAR TRANSFORM (AND INVERSE)
- HADAMARD TRANSFORM (AND INVERSE)

CODERS

- ANALOG-TO-DIGITAL
- DIGITAL-TO-ANALOG
- SAMPLE HOLD DIGITAL-TO-ANALOG
- MULTI-LEVEL PCM
- INTERLEAVER
- DE-INTERLEAVER

MATH

- DIFFERENTIATOR
- INTEGRAL WITH INITIAL CONDITIONS
- INTEGRATOR

MISCELLANEOUS

- TIME DELAY
- PHASE SHIFTER
- SIGNAL SPLIT
- TIME LATCH
- ZERO CROSSING DETECTOR

5.2. THE POWER SYSTEM LIBRARY

THE FOLLOWING IS A BRIEF SUMMARY OF THE POWER SYSTEM MODELS GENERATED UNDER THIS CONTRACT. VOLUME I CONTAINS THE COMPLETE DESCRIPTION OF EACH MODEL AND ITS USE.

POWER CONTROL ELEMENTS

- REMOTE POWER CONTROLLER
- REMOTE CIRCUIT BREAKER
- REMOTE SOFTWARE 'SWITCH'
- SEQUENCE OF EVENTS GENERATOR & LOAD CONTROLLER
- FUSE
- DIODE

POWER SOURCES

- FUEL CELL
- POWER CONDITIONING (INVERTER)

LOADS

- LOAD DATA BASE GENERATION

DISTRIBUTION

- CABLE

6.0 EXAMPLES

6. EXAMPLES

THREE EXAMPLES ARE PRESENTED IN THIS SECTION, NAMELY:

1 SYSTID SIMULATION OF AN APOLLO PCM/PM/PM COMMUNICATIONS LINK WHOSE CHARACTERISTICS ARE GIVEN IN THE FOLLOWING DIAGRAM.

2 SYSTID SQUARING LOOP MODEL ALSO DEFINED IN THE DIAGRAM.

3 SYSTID SIMULATION OF FILTER RESPONSE. THIS EXAMPLE ILLUSTRATES THE DEFINE, VARY, AND SET COMMANDS.

ALL OF THE PLOTTED OUTPUT PRESENTED IN THIS SECTION HAS BEEN EDITED SLIGHTLY TO FIT THE REQUIREMENTS OF THE DOCUMENT PROCESSOR.

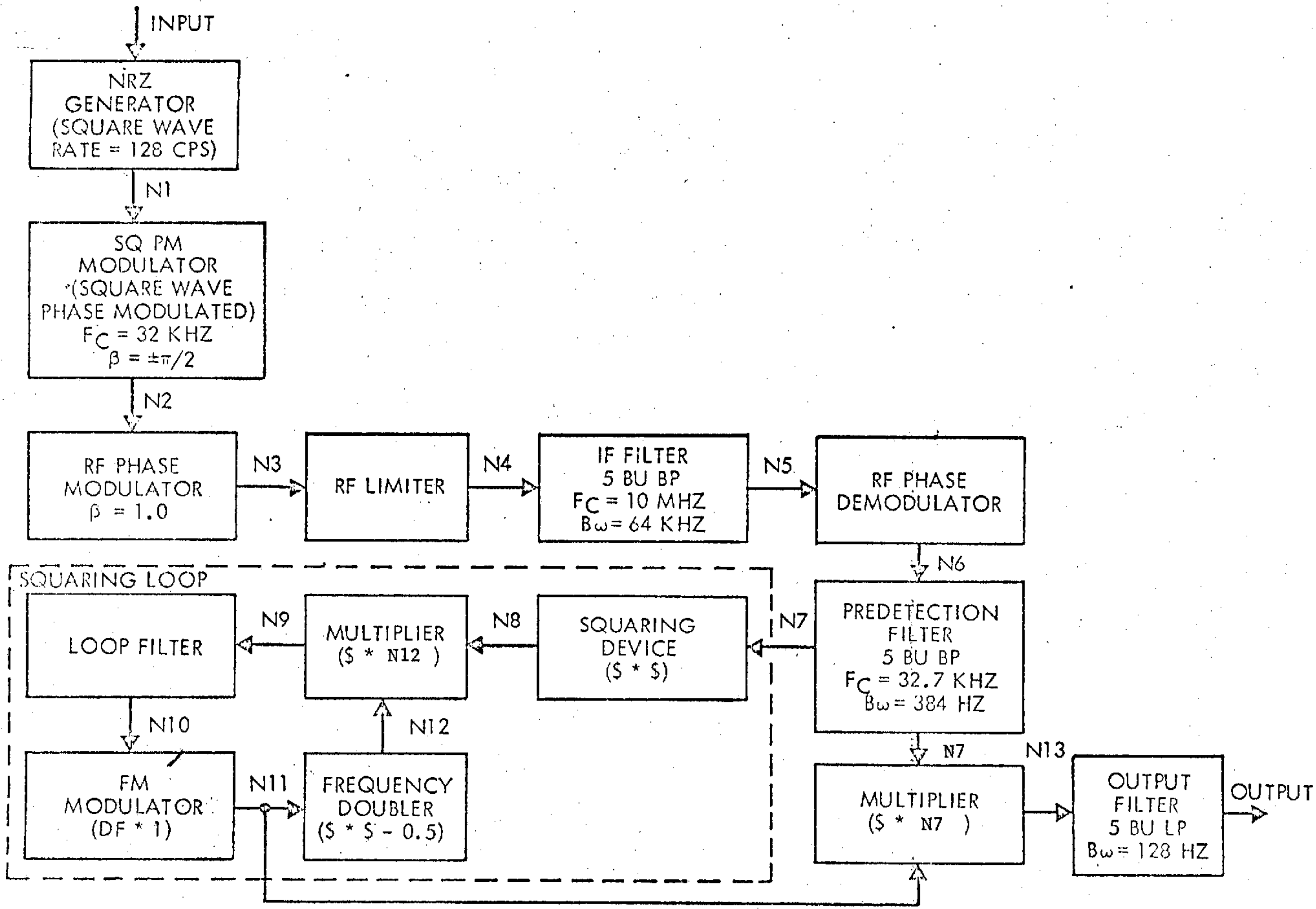


Figure 6-1. Apollo PCM/PM/PM Link Block Diagram

6.1. PCM/PM/PM LINK

THE FIRST EXAMPLE CONSISTS OF A TEMPORARY DEFINITION OF MODEL NRZ AND A SYSTEM WHICH DEMONSTRATES THE USE OF SEVERAL LIBRARY MODELS, ALONG WITH SOME MATH EXPRESSIONS. IT IS FOLLOWED BY THE PHASE I SYSTID OUTPUT, THE GENERATED FORTRAN PROGRAM, AND SOME OF THE GENERATED RESULTS.

A FEW REMARKS CONCERNING THIS SIMULATION ARE IN ORDER. NOTICE THAT FOLLOWING THE SQUARE WAVE PHASE MODULATOR IS AN RF PHASE MODULATOR. AT THIS POINT, WE ARE REPRESENTING THE RF PORTION OF THE LINK AT BASEBAND. THIS PROCESS IS ACTUALLY IMPLE- MENTED BY ASSUMING THAT THE INPUT SIGNALS ARE ANALYTIC. THIS CONDITION IS MET IF THE BASEBAND SIGNAL SPECTRUM IS ESSENTIALLY ZERO AT THE CARRIER FREQUENCY. IF THIS APPROXIMATION IS NOT TRUE, A RIPPLE IN THE SIMULATION OUTPUT AT FREQUENCIES OF APPROXIMATELY TWICE THE CARRIER IS INTRODUCED -- THE CASE IF THE INPUT IS A STEP FUNCTION, FOR INSTANCE. IN ANY EVENT, THE RIPPLE IS NORMALLY NEGLIGIBLE DUE TO THE LARGE RATIO BETWEEN THE BASEBAND AND CARRIER.

```
NRZ                                        MODELA    1     1  1   00000000n000

SYSTID PROCESSOR LEVEL III
VERSION DATED 4 JULY 74 FOR THE MSC U1108/1110 SYSTEM
THIS DECK PROCESSED ON 11:07:74  AT 19:46:36

SYSTID MODELS REFERENCED                                ENTRY POINT

     SQ                                                  SQ

THIS MODEL ASSIGNED THE ENTRY POINT NAME MODELA


000001            MODEL : INPUT-NRZ(BR)-OUTPUT
000002                  INPUT < SQ(BR/2.) > N1
000003                  N1 < $*.5+.5 > OUTPUT
000004            END
```

APOLLO PCM/PM/PM LINK (EXAMPLE 1)

```
SYSTID PROCESSOR LEVEL III
VERSION DATED 4 JULY 74 FOR THE MSC U1108/1110 SYSTEM
THIS DECK PROCESSED ON 11:07:74  AT 19:46:37


SYSTID MODELS REFERENCED                    ENTRY POINT

      NRZ                                      MODELA
      SQPMOD                                   SQPMOD
      RFPHASEMODULATOR                         RPMMOD
      RFLIMITER                                RFLIMT
      BUTTERWORTH                              BUTWTH
      RFPHASEDEMODULATOR                       RFPDEM
      LOOPFILTER                               LEDLAG
      FMMODULATOR                              FMMOD


                 SYSTEM .  APOLLO PCM/PM/PM LINK (EXAMPLE 1)
                 PAGE.  SMALL
                 DEFAUL. TSTART=0.,TSTOP=.03,DT=1.5E-6,NPRINT=200
                 PRINT,N1,N13,OUTPUT
                 PPLOT,N1,N13,OUTPUT
                 COMPLEX. N3,N4,N5
                 INPUT < NRZ(128.) > N1
                 N1 < SQPMOD(PI,32.768E3)> N2
                 N2 <RF PHASE MODULATOR (1,0) > N3
                 N3 < RF LIMITER > N4
 ** WARNING **   AN IN/OUT NODE TO THIS MODEL MAY BE OF THE WRONG TYPE
                 N4 < BUTTERWORTH (5,3,10.E6,64.E3,10.E6,1.) > N5
                 N5 < RF PHASE DEMODULATOR (1,0) > N6
                 N6 < BUTTERWORTH (5,3,32.768E3,384.,0.,1.) > N7
                 N7 < $*$ > N8
                 N8 < $*N12 > N9
                 N9 < LOOP FILTER (200.,1.8775994,0.,.19751719,0.) > N10
                 N10 < FM MODULATOR (2.*PI,32.768E3) > N11
                 N11 < $*$-0.5 > N12
                 N11 < $*N7 > N13
                 N13 < BUTTERWORTH (2,1,0.,128.,0.,1.) > OUTPUT
                 END
```

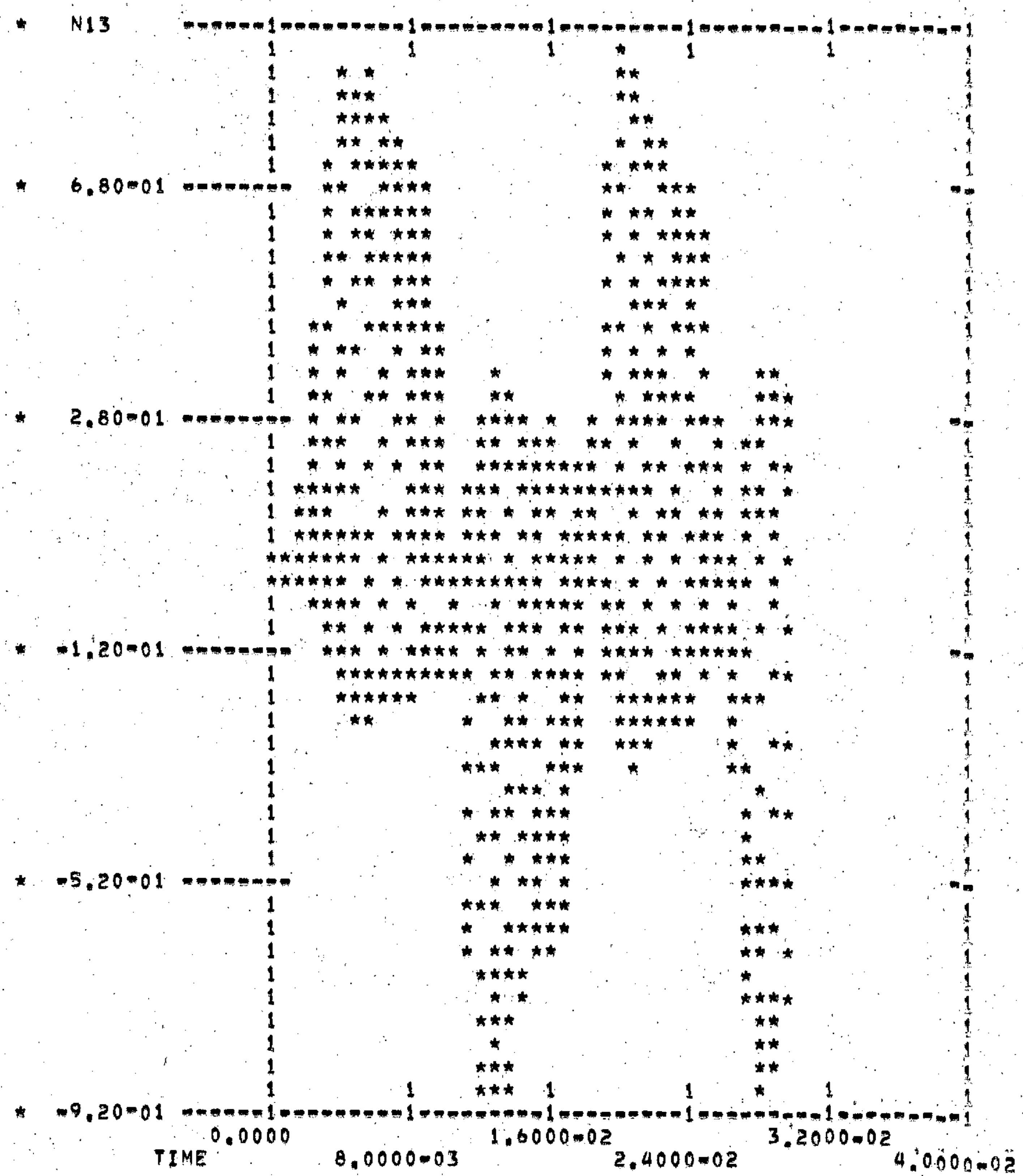
```
@ELT,I MODELA/SYSTID
      SUBROUTINE MODELA(BR)
C     NRZ
      INCLUDE MODEL1,LIST
      DEFINE INPUT=V(ZIN)
      DEFINE OUTPUT=V(ZOUT)
      DEFINE N1=V(Z+1)
      ZIN=ZZIN
      ZOUT=ZZOUT
      Z=ZZ
      ZZ=Z+1
C     SQ
      ZZIN=ZIN                                                        INPUT
      ZZOUT=Z+1                                                       N1
      CALL SQ(BR/2.)
      OUTPUT=N1*.5+.5
      RETURN
      END
@FOR,S MODELA/SYSTID,MODELA/SYSTID
@ELT,I MAIN/SYSTID
      INCLUDE MAIN1,LIST
      DATA TITLE/'APOLLO PCM/PM/PM LINK (EXAMPLE 1)                     '/
      DEFINE N3=W(1)
      DEFINE N4=W(2)
      DEFINE N5=W(3)
      DEFINE INPUT=V(7)
      DEFINE N1=V(8)
      DEFINE N2=V(9)
      DEFINE N6=V(10)
      DEFINE N7=V(11)
      DEFINE N8=V(12)
      DEFINE N9=V(13)
      DEFINE N10=V(14)
      DEFINE N11=V(15)
      DEFINE N12=V(16)
      DEFINE N13=V(17)
      DEFINE OUTPUT=V(18)
      DATA TSTART/0./,TSTOP/.03/,DT/1.5E-6/,NPRINT/200/
      PARAMETER ZPSIZE= 4
      DIMENSION VPRINT(6,ZPSIZE)
      VPRINT(1,1)=' '
      VPRINT(2,1)='TIME'
      VPRINT(1,2)=' '
      VPRINT(2,2)='N1'
      VPRINT(1,3)=' '
      VPRINT(2,3)='N13'
      VPRINT(1,4)=' '
      VPRINT(2,4)='OUTPUT'
      PARAMETER ZDRMSZ= 560
      DIMENSION VDRUM(ZDRMSZ, 3)
```

```
      COMMON /DRMHED/ZDATE(2),ZTOD(2),TSTART,TSTOP,VEQDT,SETTLE,
     .ZUSED,ZQUAL,ZNAMES,ZNAME( 3)
      DATA ZNAMES/ 3/
      ZNAME( 1)= 'N1     '
      ZNAME( 2)= 'N13    '
      ZNAME( 3)= 'OUTPUT'
      INCLUDE MAIN2,LIST
      ZZ=18
C     NRZ
      ZZIN=7                                                    INPUT
      ZZOUT=8                                                   N1
      CALL MODELA(128.)
C     SQPMOD
      ZZIN=8                                                    N1
      ZZOUT=9                                                   N2
      CALL SQPMOD(PI,32.768E3)
C     RFPHASEMODULATOR
      ZZIN=9                                                    N2
      ZZOUT=1
      CALL RPMMOD(1.0)
C     RFLIMITER
      ZZIN=1                                                    N3
      ZZOUT=2                                                   N4
      CALL RFLIMT
C     BUTTERWORTH
      ZZIN=2                                                    N4
      ZZOUT=3                                                   N5
      CALL BUTWTH(5,3,10.E6,64.E3,10.E6,1.)
C     RFPHASEDEMODULATOR
      ZZIN=3                                                    N5
      ZZOUT=10                                                  N6
      CALL RFPDEM(1.0)
C     BUTTERWORTH
      ZZIN=10                                                   N6
      ZZOUT=11                                                  N7
      CALL BUTWTH(5,3,32.768E3,384.,0.,1.)
      N8=N7*N7
      N9=N8*N12
C     LOOPFILTER
      ZZIN=13
      ZZOUT=14                                                  N10
      CALL LEDLAG(200.,1.8775994,0.,.19751719,0.)
C     FMMODULATOR
      ZZIN=14                                                   N10
      ZZOUT=15                                                  N11
      CALL FMMOD(2.*PI,32.768E3)
      N12=N11*N11-0.5
      N13=N11*N7
C     BUTTERWORTH
      ZZIN=17                                                   N13
```

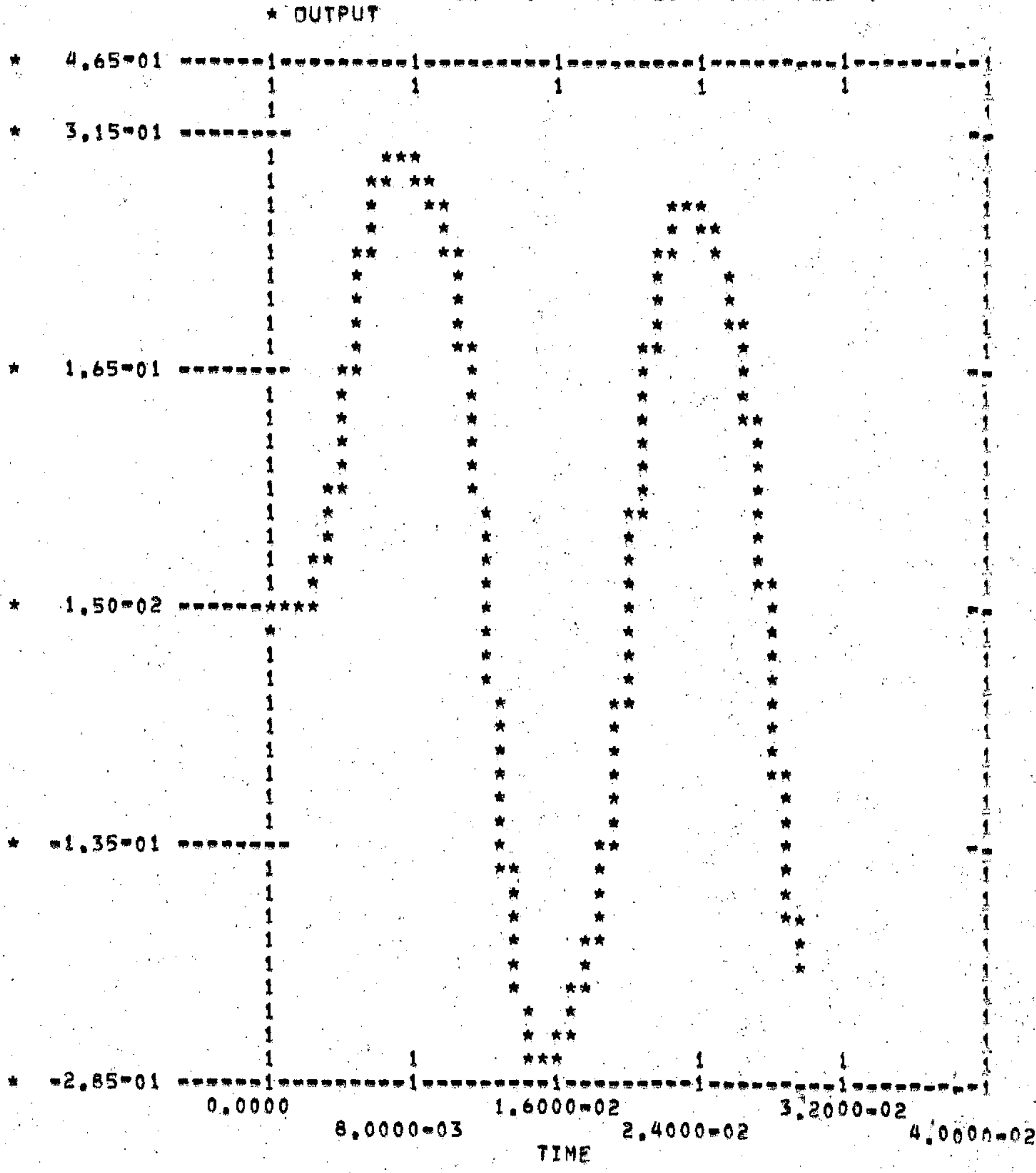
```
      ZZOUT=18                                                        OUTPUT
      CALL BUTWTH(2,1,0.,128.,0.,1.)
      IF(ZTIME.LT.ZSETTL) GO TO 99010
      INCLUDE MAIN3,LIST
      VDRUM(ZDRUM,1)=N1
      VDRUM(ZDRUM,2)=N13
      VDRUM(ZDRUM,3)=OUTPUT
      ZCOUNT=ZCOUNT+1
      IF(ZCOUNT.NE.NPRINT) GO TO 99010
      ZCOUNT=0
      ZPRINT=ZPRINT+1
      VPRINT(ZPRINT,1)=TIME
      VPRINT(ZPRINT,2)=N1
      VPRINT(ZPRINT,3)=N13
      VPRINT(ZPRINT,4)=OUTPUT
      IF(ZPRINT.NE.6) GO TO 99010
      WRITE(6,99050)VPRINT
99050 FORMAT( 4(6X,A2,A6,4(4X,E12.6),/))
      INCLUDE MAIN5,LIST
      CALL DRUMIT(VDRUM,ZDRMSZ)
      CALL PTPLT('N1',4)
      CALL PTPLT('N13',4)
      CALL PTPLT('OUTPUT',4)
99200 CONTINUE
      STOP
      END
@FOR,S MAIN/SYSTID,MAIN/SYSTID
@EOF .
```

APOLLO PCM/PM/PM LINK (EXAMPLE 1)

TIME	.000000	.300000-03	.600000-03	.900000-03
N1	.500000+00	.100000+01	.100000+01	.100000+01
N13	.000000	.226510-04	.173319-03	.244073-03
OUTPUT	.000000	.155124-07	.167936-05	.243135-04
TIME	.120000-02	.150000-02	.180000-02	.210000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
N13	.169178-01	.371338-02	.955325-02	.146721+00
OUTPUT	.151752-03	.596404-03	.174566-02	.416507-02
TIME	.240000-02	.270000-02	.300000-02	.330000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
N13	.206673-02	.563126-01	.418765+00	-.287346-01
OUTPUT	.854985-02	.156239-01	.260217-01	.401682-01
TIME	.360000-02	.390000-02	.420000-02	.450000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
N13	.144494+00	.690174+00	-.870913-01	.228112+00
OUTPUT	.581655-01	.797507-01	.104296+00	.130856+00
TIME	.480000-02	.510000-02	.540000-02	.570000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
N13	.814962+00	-.143851+00	.275379+00	.786201+00
OUTPUT	.158280+00	.185356+00	.210904+00	.233912+00
TIME	.600000-02	.630000-02	.660000-02	.690000-02
N1	.100000+01	.100000+01	.100000+01	.100000+01
N13	-.176638+00	.295585+00	.693703+00	-.191108+00
OUTPUT	.253616+00	.269496+00	.281350+00	.289249+00


N13
6.80-01
2.80-01
-1.20-01
-5.20-01
-9.20-01
0.0000
TIME
8.0000-03
1.6000-02
2.4000-02
3.2000-02
4.0000-02

APOLLO PCM/PM/PM LINK (EXAMPLE 1)



6.2. SQUARING LOOP

THE SECOND EXAMPLE ILLUSTRATES THE DEFINITION OF A MODEL, NAMELY A SQUARING LOOP. SQUARING LOOPS HAVE BEEN UTILIZED IN DERIVING SUBCARRIER PHASE REFERENCES FOR DETECTION OF PHASE MODULATED SIGNALS. THE TOPOLOGY IS SHOWN IN THE ILLUSTRATION SINCE THE ELEMENTS MAKING UP THE PROPOSED MODEL WERE USED IN THE APOLLO LINK SIMULATION.

SQUARINGLOOP MODELA 0 1 1 000000000000

SYSTID PROCESSOR LEVEL III
VERSION DATED 4 JULY 74 FOR THE MSC U1108/1110 SYSTEM
THIS DECK PROCESSED ON 11:07:74 AT 20:12:25

SYSTID MODELS REFERENCED	ENTRY POINT
LOOPFILTER	LEDLAG
FMMODULATOR	FMMOD

THIS MODEL ASSIGNED THE ENTRY POINT NAME MODELA

```
MODEL : N7 = SQUARING LOOP = N11
N7 < $*$ > N8
N8 < $*N12 > N9
N9 < LOOP FILTER (200.,1.8775994,0.,.19751719,0.) > N10
N10 < FM MODULATOR (2.*PI,32.768E3) > N11
N11 < $*$-0.5 > N12
END
```

6.3. FILTER RESPONSE

IN THE FINAL EXAMPLE, A FILTER IS SIMULATED, AND ITS RESPONSE TO AN INPUT SIGNAL IS MEASURED FOR VARIOUS BANDWIDTHS. THE SYSTEM VARIABLES DT AND TSTOP ARE ADJUSTED APPROPRIATELY FOR EACH BANDWIDTH (BW) CONSIDERED.

THE RESULTS SHOWN ARE THE FIRST PAGE OF PRINTED OUTPUT, AND THE PLOTS GENERATED WHEN THE BANDWIDTH WAS 10., AND 90. RESPECTI- VELY.

TEST THE VARY/DEFINE FEATURE

SYSTID PROCESSOR LEVEL III
VERSION DATED 4 JULY 74 FOR THE MSC U1108/1110 SYSTEM
THIS DECK PROCESSED ON 11:07:74 AT 17:57:54

SYSTID MODELS REFERENCED	ENTRY POINT
BUTTERWORTH	BUTWTH

```
SYSTEM.  TEST THE VARY/DEFINE FEATURE
PRINT.  OUTPUT,TESTER
PPLOT.  OUTPUT
PAGE. SMALL
DEFINE.  TESTER=1.+(BW-10.)/20
SET:  NPRINT=10
VARY.  BW:10.:110.:20.
SET:  DT=.05/BW
SET:  TSTOP=5/BW
      INPUT< 1,0 >N1
      N1< BUTTERWORTH(5,1,0.,BW,0.,1.) >OUTPUT
END
```

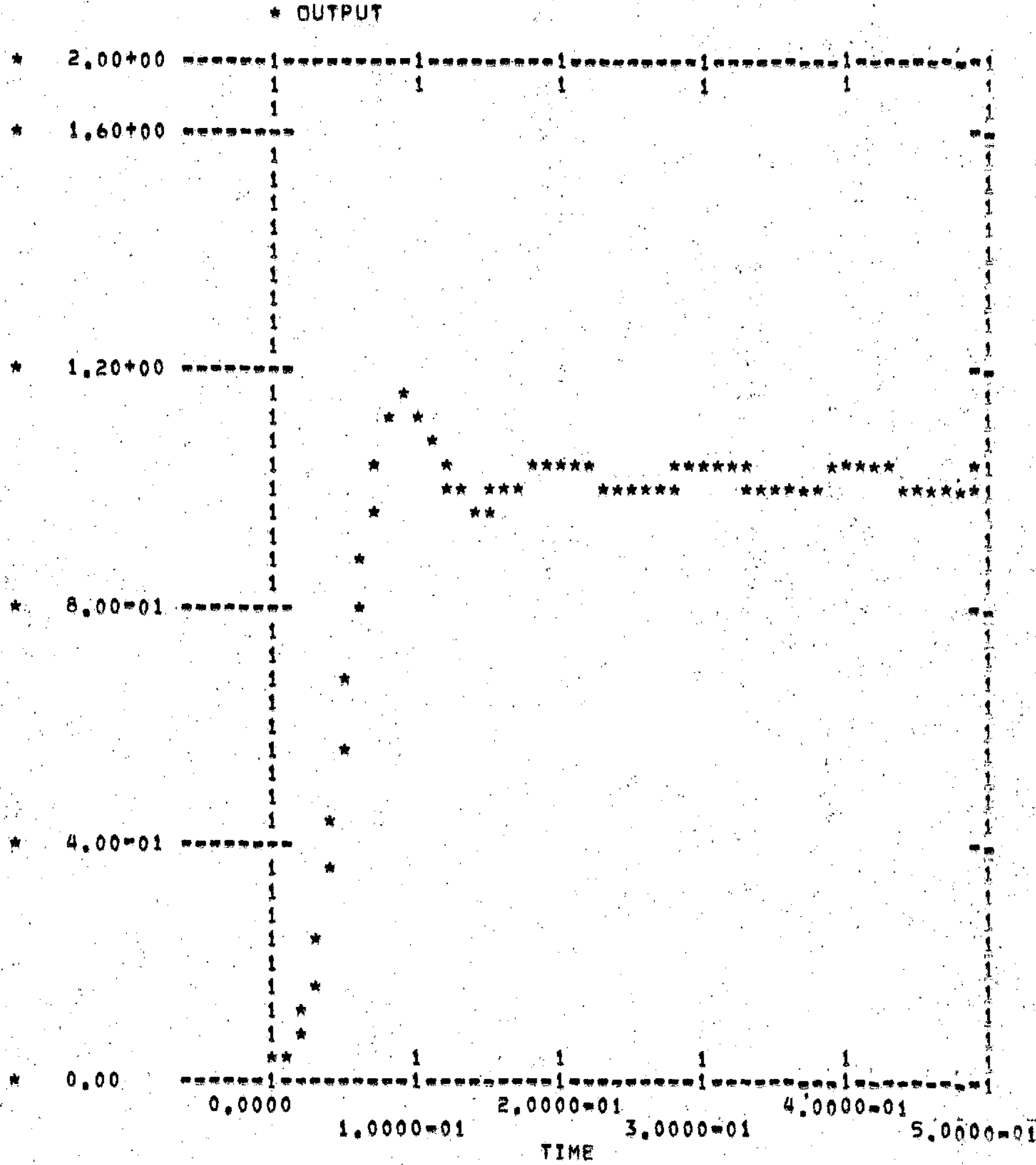
TEST THE VARY/DEFINE FEATURE

TIME	.000000	.500000-01	.100000+00	.150000+00
OUTPUT	.597958-04	.434828+00	.112984+01	.954789+00
TESTER	.100000+01	.100000+01	.100000+01	.100000+01

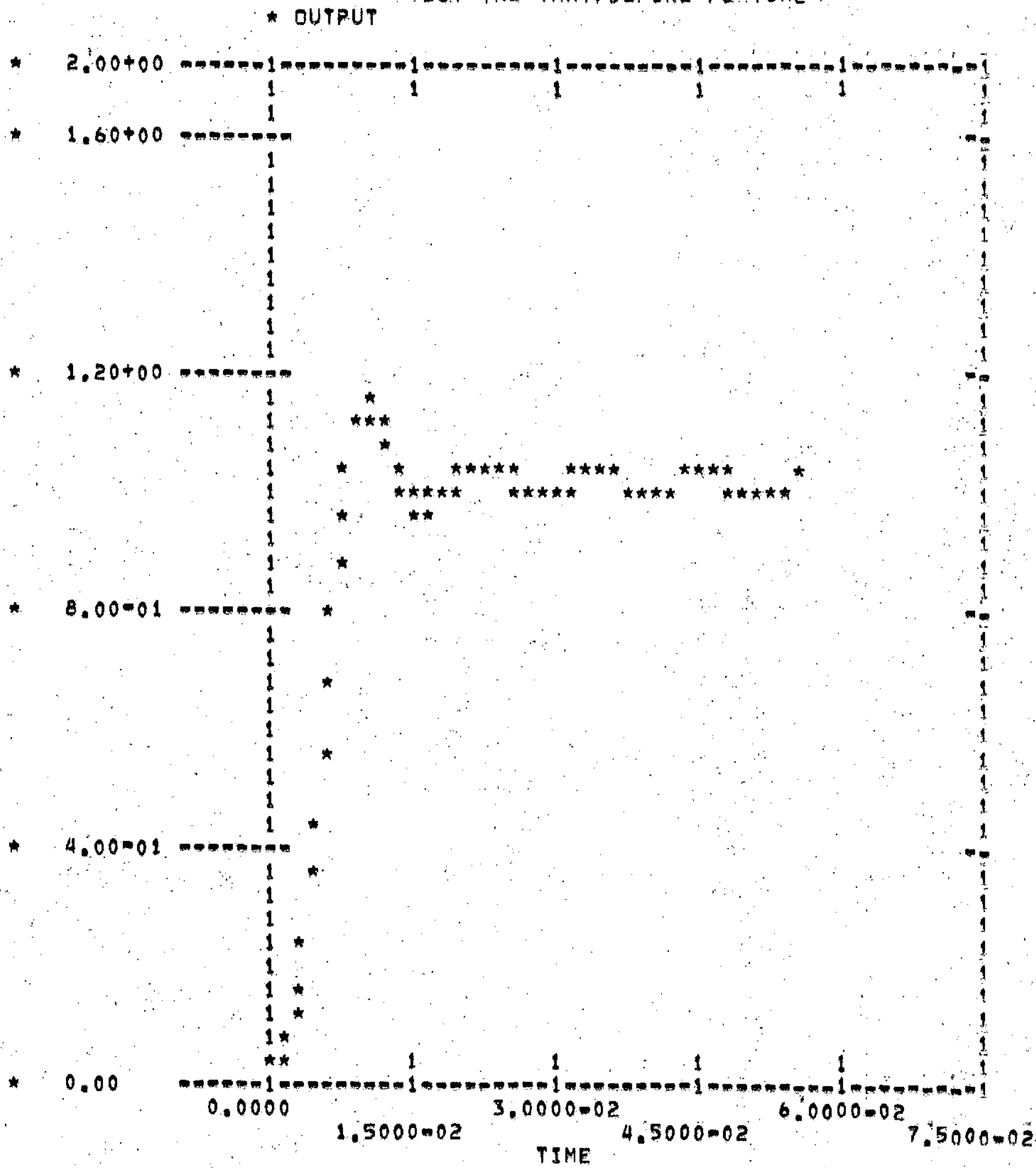
TIME	.200000+00	.250000+00	.300000+00	.350000+00
OUTPUT	.101650+01	.994126+00	.100202+01	.999332+00
TESTER	.100000+01	.100000+01	.100000+01	.100000+01

TIME	.400000+00	.450000+00	.500000+00
OUTPUT	.100021+01	.999938+00	.100002+01
TESTER	.100000+01	.100000+01	.100000+01

TEST THE VARY/DEFINE FEATURE



TEST THE VARY/DEFINE FEATURE



7.0 APPENDICES

7. APPENDIX..........SYSTID LEVEL III FOR THE LEVEL 2 USER

THERE ARE SEVERAL DIFFERENCES BETWEEN THE OLD LEVEL OF SYSTID AND LEVEL III OF SYSTID WHICH THE USER MUST BE AWARE OF. THE DIFFERENCES ARE HOPEFULLY AN IMPROVEMENT, ADDING TO THE INHERENT FLEXIBILITY OF SYSTID. THE CHANGES MADE CAN BE SUMMARIZED AS FOLLOWS:

(1) GENERALIZED TOPOLOGY

A) MODELS DIRECTLY HAVE MULTIPLE INPUTS AND OUTPUTS

B) TAPS HAVE BEEN ELIMINATED

C) TOPOLOGY STATEMENTS ARE NO LONGER SORTED, BUT ARE EXECUTED IN THE ORDER IN WHICH THEY ARE INPUT

D) NODE NAMES MAY BE USED IN EXPRESSIONS

(2) NODE TYPING

A) SYSTID NODES MAY NOW BE TYPED EITHER REAL OR COMPLEX. COMPLEX NUMBERS WILL BE PROCESSED CORRECTLY THRU ALL EXPRESSION EVALUATIONS.

B) THERE ARE CONVENIENT METHODS OF ACCESSING BOTH THE REAL AND IMAGINARY PARTS OF A COMPLEX NODE

(3) FORTRAN STATEMENTS

A) EXECUTABLE FORTRAN STATEMENTS (INCLUDING FORMAT) MAY NOW BE INTERMIXED WITH TOPOLOGY STATEMENTS USING A SPECIAL LEFT NODE FIELD.

B) NON-EXECUTABLE FORTRAN STATEMENTS MAY NOW

BE INTERMIXED WITH SYSTID COMMANDS

7.1. GENERALIZED TOPOLOGY

7.1.1. MULTIPLE INPUTS AND OUTPUTS

THE NODE NAMES INPUT AND OUTPUT NO LONGER HAVE A SPECIAL MEANING IN SYSTID. THE NEW VERSION ALLOWS MULTIPLE INPUT AND OUTPUT NODES WITH A MODEL DECLARED AS FOLLOWS:

MODEL _ IN1,..._ INK _ MODEL NAME(ARG1,...,ARGN) _ OUT1,..._ OUTJ

THE PARENTHESIS AND COMMAS IN THE MODEL NAME ARGUMENT LIST ARE REQUIRED. THE _ CHARACTER MAY BE ANY NON-ALPHANUMERIC CHARACTER, HOWEVER, TO AVOID CONFUSION THEY SHOULD NOT BE A '(' OR ')'. TO AVOID AMBIGUITY IN THE CASE WHERE THE MODEL NAME IS LESS THAN SEVEN CHARCTERS AND THERE ARE NO ARGUMENTS TO THE MODEL, THE FOLLOWING FORMS MAY BE USED:

.... _ NAME () _ ...

OR

.... _ (NAME) _ ....

REFERENCING A MODEL IS ACCOMPLISHED IN A MANNER SIMILAR TO THE THE DECLARATION OF A MODEL:

```
IN1,..- INK < MODEL NAME(ARG1,...ARGN) > - OUT1,...- OUTJ
```

EXAMPLE: A MODEL TO ADD FOUR NODES TOGETHER AND MULTIPLY THE RESULT BY A CONSTANT:

```
MODEL * IN1,IN2,IN3,IN4 * ADD4(ARG) * RESULT
IN1 < $ + IN2 + IN3 + IN4 > TEMP
TEMP < TEMP * ARG > RESULT
END
```

A TYPICAL REFERENCE FROM ANOTHER MODEL OR SYSTEM MIGHT BE:

```
NODE1-NODE2-NODE3-NODE4 < ADD4(2.718) > NODE10
```

7.1.2. ELIMINATION OF TAPS

TAPS ARE NO LONGER RECOGNIZED BY SYSTID. IN THE PRE- VIOUS VERSION, TAPS WERE USED FOR SEVERAL REASONS:

1) TO ALLOW A DANGLING NODE, FOR OUTPUT OR SOME OTHER REASON, WHICH WOULD OTHERWISE CAUSE A DIAGNOSTIC AND AN ABORTED RUN.

   DANGLING NODES ARE ALLOWED IN THE NEW VERSION.

2) TO PROVIDE MORE THAN ONE INPUT OR OUTPUT TO A MODEL.

MULTIPLE INPUTS AND OUTPUTS ARE ALLOWED IN THE NEW VERSION.

3) TO USE THE VALUE OF A NODE IN AN EXPRESSION.

NODE NAMES ARE ALLOWED IN EXPRESSIONS IN THE NEW VERSION.

4) TO ISOLATE THE OUTPUT OF A MODEL FROM THE VALUE AT THE NODE TO WHICH THE MODEL IS CONNECTED (THE VALUE OF THE NODE IS THE SUM OF ALL THE OUTPUTS OF THE MODELS CONNECTED TO THE NODE).

IN THE NEW VERSION, THE OUTPUT OF MODELS MAY BE SEPARATED BY THE INSERTION OF A DUMMY EXPRESSION BETWEEN THE MODEL OUTPUT AND THE OUTPUT NODE.

FOR EXAMPLE, IN THE OLD VERSION:

```
N1 < MODEL1 > N3 * TAP1
N2 < MODEL2 > N3
```

IN THE NEW VERSION, THE COUNTERPART WOULD BE:

```
N1 < MODEL1 > TAP1
TAP1 < $ > N3
N2 < MODEL2 > N3
```

IN BOTH CASES TAP1 IS THE OUTPUT OF MODEL1 AND N3 IS THE SUM OF THE OUTPUTS OF MODEL1 AND MODEL2.

7.1.3. SORTING

IN ALL PREVIOUS VERSIONS OF SYSTID, THE TOPOLOGY STATEMENTS WERE SORTED AS THE FORTRAN CODE WAS GENERATED IN AN ATTEMPT TO COMPUTE ALL OUTPUT NODES BEFORE THEY WERE NEEDED FOR INPUT TO SOME OTHER MODEL. IN THE NEW VERSION, SORTING IS NOT PERFORMED. THE ORDER OF THE MODEL REFERENCES IS LEFT TO THE USER. THIS GIVES THE USER GREATER CONTROL OVER WHAT FINALLY HAPPENS IN THE GENERATED

PROGRAM AND MAKES POSSIBLE THE MIXING OF FORTRAN STATEMENTS WITH THE TOPOLOGY STATEMENTS.

7.1.4. NODE NAMES IN EXPRESSIONS

A NODE NAME MAY BE USED IN ANY TOPOLOGY OR FORTRAN STATEMENT AND THE CURRENT SIMULATION VALUE OF THE NODE WILL BE USED. SINCE THE TOPOLOGY STATEMENTS ARE NO LONGER SORTED, THE PROPER ARRANGEMENT OF STATEMENTS TO INSURE THE CORRECT VALUE BEING USED IS NOW FACILITATED.

7.2. NODE TYPES AND ACCESS TO NODES

7.2.1. NODE TYPES

IN THE PREVIOUS VERSION OF SYSTID ALL NODES WERE ASSUMED TO BE COMPLEX FOR THE PURPOSE OF STORAGE ALLOCATION. COMPLEX VALUES WERE TRANSMITTED FROM ONE MODEL TO ANOTHER ONLY IF THE INTERVENING NODE HAD ONLY ONE MODEL OUTPUTTING TO IT. IF MORE THAN ONE MODEL HAD ITS OUTPUT AT A NODE, THE IMAGINARY PARTS OF THE COMPLEX NODES WERE NOT SUMMED CORRECTLY.

IN THE NEW VERSION, ALL NODES ARE ASSUMED TO BE REAL UNLESS THEY ARE DECLARED COMPLEX BY USE OF THE FOLLOWING STATEMENT:

COMPLEX - ...NODE LIST....

ALSO, ALL NODES MAY BE FORCED TO BE COMPLEX BY USING:

IMPLICIT COMPLEX

WITH ALL NODES BEING COMPLEX, REAL NODES MAY BE DECLARED BY:

REAL - ...NODE LIST...

RESTRICTION. ALL NODES IN A SYSTID MODEL OR SYSTEM ARE THUS TYPED EITHER REAL OR COMPLEX. OTHER VARIABLES MAY BE OF ANY TYPE DEPENDING ON THE FIRST LETTER OF THEIR NAME AND ANY FORTRAN TYPE STATEMENTS WHICH THE USER MAY WISH TO INCLUDE. THE FORTRAN **IMPLICIT** STATEMENT IS **ILLEGAL** IN SYSTID ALTHOUGH SPECIFIC TYPE STATEMENTS MAY BE USED (E.G. INTEGER,ETC). IF FORTRAN **TYPE** STATEMENTS ARE NOT USED, THE FOLLOWING RULES APPLY:

A-H,O-V,X,Y ARE REAL
I-N,Z ARE INTEGER
W IS COMPLEX

THE USER SHOULD BE CAUTIOUS IN USING VARIABLES WHICH BEGIN

WITH THE LETTERS V,W, AND Z AS A CONFLICT WITH INTERNAL VARIABLES MAY RESULT. THE USER IS ALSO CAUTIONED TO CHECK THE TYPES OF VARIABLES USED AS ARGUMENTS PASSED TO MODELS OR IN MODEL DEFINITIONS.

7.2.2. ACCESS TO COMPLEX NODES

IF A NODE NAME IS TYPED COMPLEX, ANY FORTRAN STATEMENT REFERENCING THE NODE WILL USE THE COMPLEX VALUE. THE FORTRAN INTRINSIC FUNCTIONS REAL AND AIMAG MUST BE USED TO REFERENCE THE REAL OR IMAGINARY PARTS.

SYSTID TOPOLOGY STATEMENTS ALSO USE THE COMPLEX VALUE OF A COMPLEX NODE, AND ACCEPT THE FUNCTIONS REAL AND AIMAG. ALSO, THE TOPOLOGY STATEMENTS ACCEPT THE FOLLOWING SHORTHAND NOTATION:

```
#NODE => REAL(NODE)

$NODE => AIMAG(NODE)

$$NODE => #NODE => REAL(NODE)

$ => VALUE OF THE FIRST INPUT NODE
```

FOR EXAMPLE:

```
COMPLEX * X,Y,R,S
X < $ + #R + $S > Y
```

WILL GENERATE: Y=X + REAL(R) + AIMAG(S)

SYSTID POST-PROCESSING COMMANDS (PRINT,PLOT,PPLOT,POST,SAVE) ARE AN EXCEPTION IN THAT A NODE NAME REFERENCES EITHER A REAL NODE OR THE REAL PART OF A COMPLEX NODE. IF THE IMAGINARY PART OF A NODE IS TO BE SAVED FOR POST-PROCESSING, IT MUST BE PRECEEDED BY A DELIMITER. THE $ IS RECOMMENDED FOR THE DELIMITER TO CONFORM TO THE NOTATION USED ABOVE. FOR EXAMPLE:

```
COMPLEX * X,Y,CT
PRINT * X,$Y,CT,$CT
```

WILL PRINT THE REAL PART OF X, THE IMAGINARY PART OF Y, AND BOTH THE REAL AND IMAGINARY PARTS OF CT.

7.2.3. FORTRAN ACCESS TO SYSTID NODES

IN GENERAL, SYSTID NODES MAY BE USED IN FORTRAN STATEMENTS JUST AS IF THEY WERE VARIABLES, WITH ONE IMPORTANT EXCEPTION. BECAUSE OF RESTRICTIONS IN THE FORTRAN COMPILER AND THE FACT THAT NODES ARE NOT VARIABLES, BUT ARE DEFINED AS LOCATIONS IN THE V-ARRAY, SYSTID NODES CANNOT APPEAR IN FORTRAN WRITE STATEMENTS. INSTEAD, A FORTRAN VARIABLE MUST BE SET EQUAL TO THE NODE , AND THE VARIABLE PRINTED. SEE 'EXECUTABLE FORTRAN & FORMAT STATEMENTS' FOR AN EXAMPLE.

7.3. INTERMIXING FORTRAN STATEMENTS

ALL SYSTID STATEMENTS ARE EITHER COMMANDS, TOPOLOGY STATEMNTS, OR THE END CARD. ALL COMMANDS (POST,DEFAULT,ETC) MUST APPEAR AT THE BEGINNING OF THE MODEL (OR SYSTEM) AND ALL TOPOLOGY STATEMENTS FOLLOW THE COMMANDS. EXECUTABLE FORTRAN STATEMENTS AND LABELLED FORMAT STATEMENTS MAY BE INTERMIXED WITH SYSTID TOPOLOGY STATEMENTS. NON-EXECUTABLE FORTRAN STATEMENTS WITH THE EXCEPTION OF THE FORMAT STATEMENT MAY BE INTERMIXED WITH THE SYSTID COMMANDS.

7.3.1. EXECUTABLE FORTRAN AND FORMAT STATEMENTS

EXECUTABLE FORTRAN STATEMENTS MAY BE INTERMIXED WITH SYSTID TOPOLOGY STATEMENTS IN ANY OF THREE WAYS:

FORM1 =>

```
            FORTRAN _ ...FORTRAN STATEMENT...

EXAMPLE:       FORTRAN * XX = NODE1
               FORTRAN * WRITE(6,300)TIME,XX
```

FORM2 (SHORTHAND) =>

```
               4TRAN _ ...FORTRAN STATEMENT

EXAMPLE:       4TRAN _ IF(TIME.GT.1.) NODE5=0
```

NOTE: THERE CANNOT BE ANY BLANKS BETWEEN THE '4' & THE 'T' IN 4TRAN

FORM3 (LABELLED STATEMENT) =>

LABEL ....FORTRAN STATEMENT...

EXAMPLE: 300 FORMAT(' ',2F10.5)

NOTE: THERE CANNOT BE ANY IMBEDDED BLANKS IN THE LABEL.
A BLANK MUST FOLLOW THE LABEL.

7.3.2. NON-EXECUTABLE FORTRAN (EXCEPT FORMAT)

NON-EXECUTABLE FORTRAN STATEMENTS MAY BE INTERMIXED WITH THE SYSTID COMMANDS, WITH THE EXCEPTION OF FORMAT AND IMPLICIT STATEMENTS. FOR EXAMPLE:

```
SYSTEM * USE FORTRAN
DEFAULT * TSTOP=1.,DT=.01
DIMENSION TABLE(5)/1.,2.,6.,24.,120./
PRINT * N1,N2
      .
      .
      .
      .
      .
```

8. CORRELATION OF REQUIREMENTS WITH DEVELOPMENTS

THIS SECTION CORRELATES THE REQUIREMENTS AS PRESENTED IN THE STATEMENT OF WORK FOR CONTRACT NAS9 13779, SECTION 3.3.1, SIMULATION LANGUAGE ENHANCEMENTS WITH THE DEVELOPMENTS DOCUMENTED ABOVE.

MULTINODE MODELS (SOW 3.3.1.1)

REFER TO 3.3.1.2 MODEL DECLARATION AND 3.4.2 MODEL REFERENCES.

SIMPLIFY EXPRESSION PROCESSING (SOW 3.3.1.2)

REFER TO 3.4.3 ACCESSING NODES IN EXPRESSIONS AND 3.5.3 VARIABLE NAMES AND TYPE CONVENTIONS.

AUTOMATIC CHECKPOINT (SOW 3.3.1.3)

A SUBROUTINE CHKPNT IS AVAILABLE IN THE LIBRARY WHICH WILL CHECKPOINT THE SIMULATION PROGRAM WHENEVER IT IS CALLED. THE CHECKPOINTED RUN CAN BE RESTARTED WITH THE EXEC8 @RSTRT COMMAND. THE CALL CAN BE INTERMIXED WITH THE SIMULATION TOPOLOGY (SEC. 3.5). FOR EXAMPLE:

```
FORTRAN * IF(MOD(TIME,10.).LT.DT) CALL CHKPNT
```

CONDITIONAL TERMINATION (SOW 3.3.1.4)

REFER TO 3.5 INTERMIXING FORTRAN. A STATEMENT TO TERMINATE THE PROGRAM CAN BE INTERMIXED WITH THE SIMULATION TOPOLOGY, FOR EXAMPLE:

```
FORTRAN : IF(NODES.GT.CUTOFF) GO TO 99200
```

SYSTID TABLE DEFINITION (SOW 3.3.1.5)

REFER TO 3.5.2 NON-EXECUTABLE FORTRAN. THE REFERENCED SECTION CONTAINS AN EXAMPLE OF TABLE DEFINITION.

SAVE (SOW 3.3.1.6)

REFER TO 3.3.3.6 SAVE. RESULTS MAY BE SAVED ON FILE BY USE OF THE SAVE COMMAND.

MODIFY SORT (SOW 3.3.1.7)

REFER TO 7.1.3 SORTING

OUTPUT FORMAT CAPABILITY (SOW 3.3.1.8)

REFER TO 3.5 INTERMIXING FORTRAN. BY NOT INCLUDING THE PRINT COMMAND, WHICH GENERATES THE STANDARD PRINTED OUTPUT, AND USING INSTEAD THE FORTRAN INCLUDE STATEMENT, THE USER MAY INSERT HIS OWN PROCEDURES FOR I/O INTO THE SIMULATION PROGRAM.

CROSS-REFERENCE OUTPUT (SOW 3.3.1.9)

REFER TO 6. EXAMPLES. THE NODES ARE AUTOMATICALLY

CROSS-REFERENCED WITH THEIR LOCATION IN THE V-ARRAY IN THE GENERATED FORTRAN LISTING.

AUTOMATIC CORE SIZING (SOW 3.3.1.10)

THE FIRST PHASE OF SYSTID NOW REQUIRES LESS THAN 20K WORDS OF CORE. THE SECOND PHASE DYNAMICALLY EXPANDS AND CONTRACTS CORE AS REQUIRED. REFER TO 3.6.3 CORE MANAGEMENT.

AUTOMATIC DIRECTORY UPDATING (SOW 3.3.1.11)

WHEN A MODEL IS COMPILED BY SYSTID, A DIRECTORY CARD IS PRINTED FOR THE MODEL, WHICH MAY BE ADDED TO THE DIRECTORY WITH THE EXEC8 EDITING FACILITIES. REFER TO 2.5 SYSTID MODEL DIRECTORY AND 6. EXAMPLES.

REAL & COMPLEX NODES (SOW 3.3.1.12)

REFER TO 3.3.2 NODE TYPING, AND 3.3.3 I/O AND POST-PROCESSING, AND 3.4.3 ACCESSING NODES IN EXPRESSIONS.

MODEL DEBUGGING CAPABILITY (SOW 3.3.1.13)

REFER TO 3.5 INTERMIXING FORTRAN. A SUBROUTINE EXISTS IN THE LIBRARY WHICH PRINTS THE PORTION OF THE V-ARRAY ALLOCATED TO A MODEL, IF A CALL TO THE SUBROUTINE IS INCLUDED IN THE MODEL. FOR EXAMPLE:

```
FORTRAN * CALL DEBUG('IDENT',ZZ,Z)
```

OTHER I/O STATEMENTS OF THE USERS CHOICE CAN ALSO BE INCLUDED.

SYSTID MODELING AIDS (SOW 3.3.1.14)

REFER TO 3.5 INTERMIXING FORTRAN. A USER CAN WRITE A MODEL ENTIRELY IN FORTRAN, AND BY PRECEEDING IT WITH A CORRECT MODEL STATEMENT, ALL INTERFACING WILL BE GENERATED AUTOMATICALLY.

CONVERT SYSTID TO EXEC8 (SOW 3.3.1.15)

REFER TO THE PREFACE.

INTERACTIVE GRAPHICS INTERFACE (SOW 3.3.1.16)

REFER TO 9. MOPS GRAPHICS PACKAGE.

MISC. SYSTID ENHANCEMENTS (SOW 3.3.1.17)

COVERED IN SOW 3.3.1.1 THROUGH SOW 3.3.1.16.

9. MOPITS -- INTERACTIVE GRAPHICS OUTPUT PROCESSOR

GRAPIC OUTPUT FROM SYSTID NORMALLY CONSISTS OF PRINTER PLOTS OR SC4020 PLOTS ON THE JSC UNIVAC SYSTEM. HOWEVER, USE OF THE JSC MOPS TERMINAL WAS IMPLEMENTED THROUGH USE OF THE SAVE COMMAND AND THE DEVELOPMENT OF A PROGRAM TO DISPLAY SAVED SYSTID DATA. THIS PROGRAM IS RESIDENT IN THE SYSTID-ABS FILE AND IS NAMED MOPITS.

THE PROGRAM QUERIES THE USER FOR THE SYSTID FILE QUALIFIER AND NODE NAME (AS DEFINED WITH A SAVE COMMAND), AND THE TIME LIMITS (X-AXIS) FOR THE PLOT. THE DATA IS FRAMED, IF NECESSARY, INTO SEVERAL PICTURES WITH CONSISTENT AXIS SCALING FOR THE ENTIRE SET.

THE PROGRAM WAS WRITTEN IN MODULAR FORM AND MAY BE EASILY MODIFIED TO SATISFY THE WHIMS OF ANY USER.

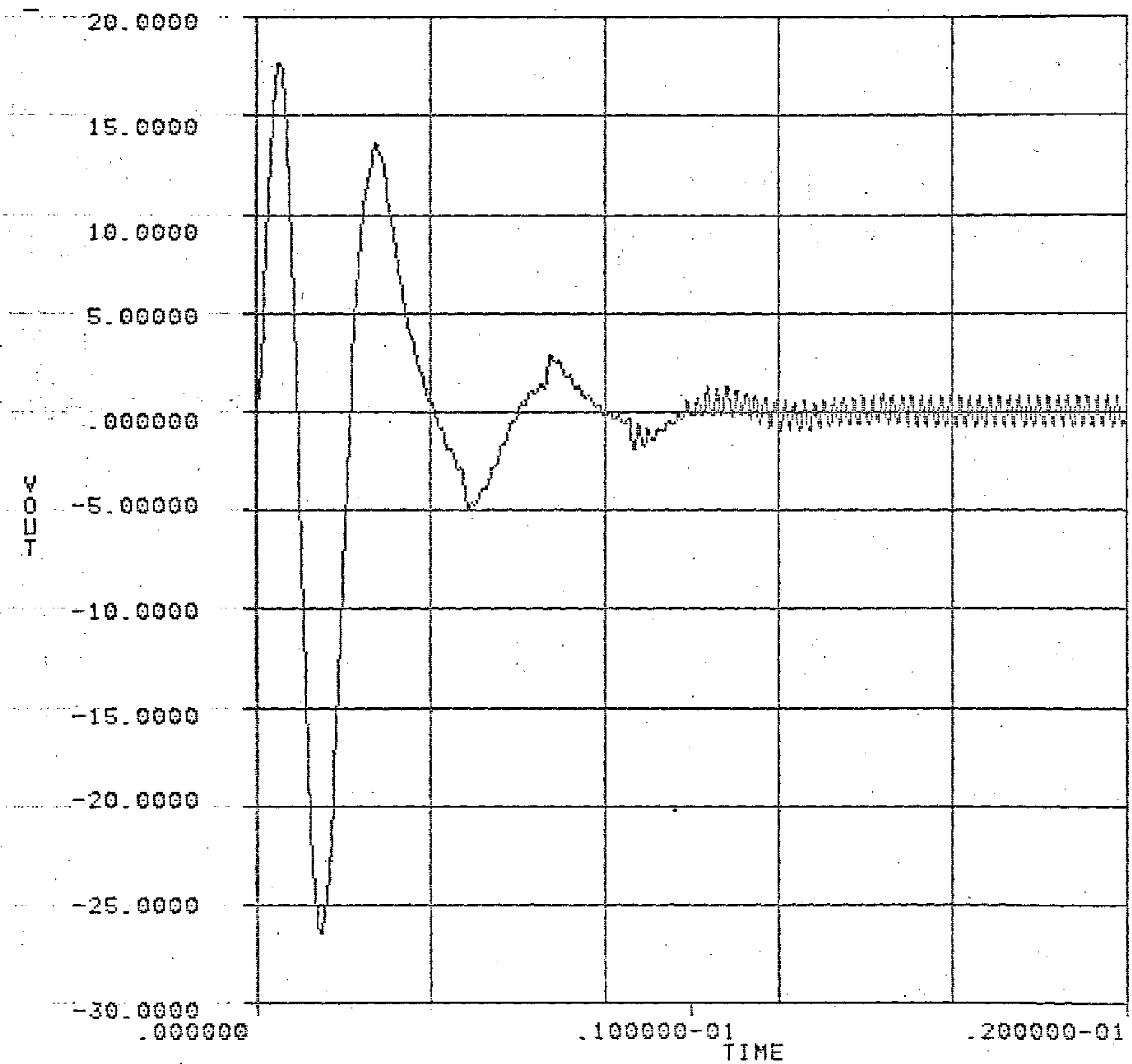
THE EXECUTION PROCEDURE IS AS FOLLOWS:

@XQT SYSTID-ABS.MOPITS

THEN:

- KEY IN FC-0
- RESPOND TO THE QUESTIONS PRINTED ON THE SCREEN
- TRANSMIT A BLANK TO CONTINUE INTO THE NEXT PLOT OUTPUT FRAME
- TYPE IN STOP FOR A GRACEFUL EXIT, OR '@' TO ABORT

The following is an example of the MOPITS routine output:



DISTRIBUTION LIST - SHUTTLE EPDC SIMULATION STUDY PROGRAM

HUGHES AIRCRAFT COMPANY
BOX 92919, LOS ANGELES, CA 90009

P. Ackerman		373/8505	
Z. Bleviss		373/8110	
J. Drebinger		373/8110	
P. Dupont		366/524	
M. Fashano	Assoc. Program Manager	373/8180	(213)648-8021
I. Highberg		373/8505	
J. MacCalla		373/8120	
L. McGlothlen		366/522	
D. Newlands		373/8180	
N. Palmquist		373/8505	
D. Paynter		373/8180	
R. Rechter	Program Manager	373/8110	(213)648-1731
J. Silianoff		373/1161	
J. Stivers		373/8595	
L. Stoolman		373/8110	
J. Sullivan		373/8180	
Data Bank (2)		373/8110	

NASA/JOHNSON SPACE CENTER
HOUSTON, TEXAS 77058

C. Dawson	Technical Officer	Bldg. 16/EJ5	(713)483-5832
R. Moorehead		Bldg. 16/EJ5	
R. Murdock (LEC)		Bldg. 16/EJ5	
J. Pawlowski	Technical Monitor	Bldg. 16/EJ5	(713)483-2981
F. Tabor	Contract Negotiator	Bldg. 16/BC7	(713)483-2746