NASA Contractor Report 3364

A Computer Program for the Generation of Logic Networks From Task Chart Data

Henry E. Herbert

CONTRACT NAS1-16078
DECEMBER 1980

NASA

NASA Contractor Report 3364

A Computer Program for the Generation of Logic Networks From Task Chart Data

Henry E. Herbert
Computer Sciences Corporation
Hampton, Virginia

Prepared for
Langley Research Center
under Contract NAS1-16078

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1980

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Section 1
INTRODUCTION

The PPARS NETWORK GENERATION PROGRAM (NETGEN) is a batch program that will provide the user with a logic network based on his input of work tasks in task list or Work Breakdown Structure (WBS) format and the relationships among the work tasks.

Past schedule analysis using the Critical Path Method (CPM) has required the schedule analyst to manually convert work tasks into a CPM-type logic network. This conversion required the re-labeling of the work tasks as activities with predecessors and successors, linking them together with interface constraints (activities), calculating all the work task and interface activity durations, and manually drawing up a network to check for logic errors. At this point, the schedular coded up the data on keypunch forms and subsequently ran the data through the PPARS Batch/EZPERT programs to validate the new network by auditing it against the manually drawn logic diagrams before doing any updates.

The NETGEN program was designed to create for the analyst as much of a logic network that could be programmed. Applications of NETGEN include assistance in building new networks and providing practical and creditable CPM analysis of work task or WBS format schedules.

The functions of the NETGEN program are as follows:

- scan the work task information

- create a predecessor and successor for each work task

- create the interface activities needed to tie the network together

- calculate all activity durations not provided by the user

- determine all start and end events

- create the control cards and data file needed to produce an EZPERT PRENET mode network plot

- prepare a file from the input data that can be input to PPARS Batch. This file is a standard logic/CPM network that can be saved and later updated by cards or by the interactive program, IAPERT.

The input consists of the basic PPARS control cards (*TITLE, *DATE, *REPORT, etc.), NETGEN activity cards, and EZPERT control cards for the PRENET(LOGIC) mode.

The output consists of various messages and lists produced by NETGEN as follows:

- printout of the original input data (NETGEN)
- diagnostic messages (NETGEN)
- list of interface activities as created by NETGEN
- list of start and end events as determined by NETGEN
- list of the final network in standard logic format

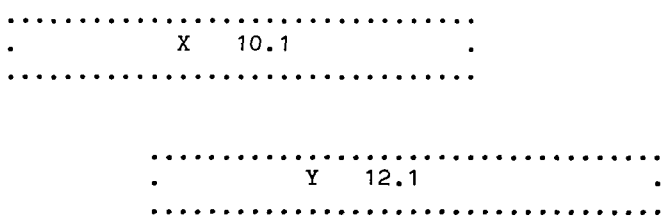
Section 2
NETWORK CONSTRUCTION

On the NASA/LaRC PPARS system, a logic network is defined to be a series of activities preceded and succeeded by event nodes. NETGEN is designed to convert task data from a barchart format into a logic network, thereby relieving the schedular of the manual chores of path construction, node labeling, calculation of time durations, and the subsequent coding of this information on the NASA Form 577. In order to build a network from the task data, NETGEN must know what the task numbers are and how the tasks are related. There are five types of task relationships used in NETGEN, and they are as follows:

1. E/E (End to End)
2. E/S (End to Start)
3. S/S (Start to Start)
4. S/E (Start to End)
5. F/A (Immediate Following Activity)

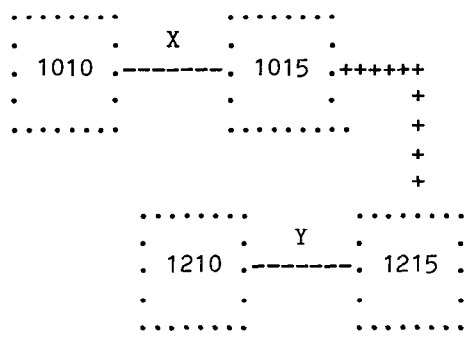
These five types specify how the activities are related, or how the event nodes are to be connected.

As an example, assume we have two tasks, X and Y, with WBS numbers 10.1 and 12.1, respectively. In task chart format, they would be represented as:



Both tasks would have a start/end date, a description and (optionally) resource. For the first example, assume that these are related (to be joined)

End to End; i.e., task X must end before task Y can end. The type of relationship is an E/E, and NETGEN will create the following network:

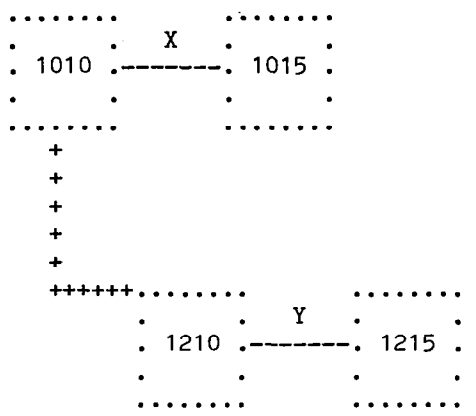


consisting of the three PERT type activities:

PREDECESSOR	SUCCESSOR	DESCRIPTION
1010	1015	X
1015	1215	E/E CONSTRAINT
1210	1215	Y

Note the numbering sequence of the activities here; NETGEN uses the original task number in deriving the predecessor and successor numbers for the new activities. Specifically, the predecessor is equal to the original task number (without the periods) multiplied by ten, i.e. , 101 * 10 = 1010. To form the successor value, NETGEN merely adds a five to the predecesssor, or 1010 + 5 = 1015 (the new successor number).

As a second example, assume that task X and Y are joined Start to Start; i.e, Task X must start before Task Y can start. The interface type is S/S and NETGEN creates the following network:

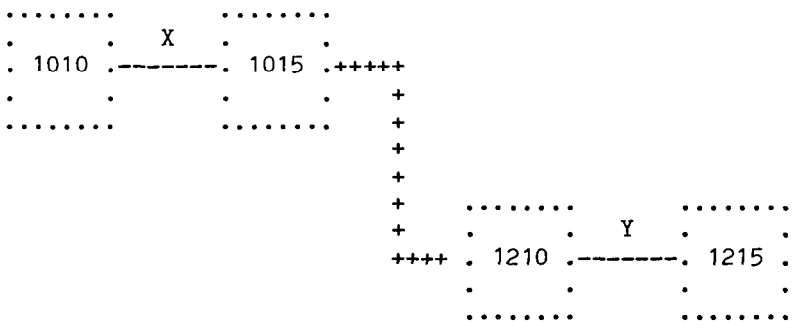


consisting of three activities:

PREDECESSOR	SUCCESSOR	DESCRIPTION
1010	1015	X
1010	1210	S/S CONSTRAINT
1210	1215	Y

Lastly, if two activities are to be joined End to Start (E/S) or Start to End (S/E), NETGEN operates as follows:

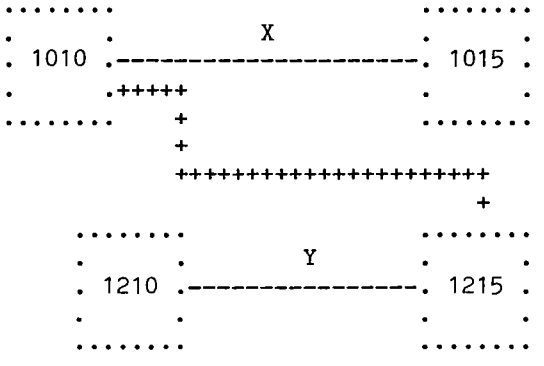
E/S (End to Start)



PREDECESSOR	SUCCESSOR	DESCRIPTION
1010	1015	X
1015	1210	E/S CONSTRAINT
1210	1215	Y

- OR -

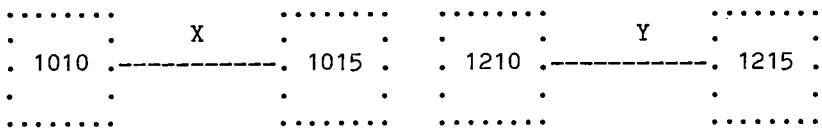
S/E (Start to End)



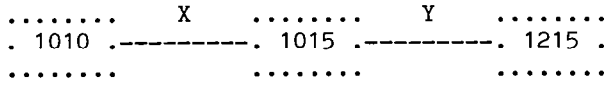
PREDECESSOR	SUCCESSOR	DESCRIPTION
1010	1015	X
1010	1215	S/E CONSTRAINT
1210	1215	Y

The important feature to note here is that these four types of relationships imply the existence of an "interface" activity between the two original tasks. This interface activity, as created by NETGEN, will have a description very similar to those given in the examples, as well as a start and end date. As with all activities, NETGEN will calculate the duration of this interface activity. If this duration is negative, NETGEN will print an error message to the user, and reset the duration to 0.

The fifth type of relationship, Immediate Following Activity, (F/A),works differently in that no interface activity is created. The F/A implies that a given task is immediately followed by the next task with no spare time between them. On F/A tasks, successor and predecessor nodes are merged into a single event. For example, use tasks X and Y again and let the relationship between them be such that Y immediately follows X. NETGEN would first create the activities:



Then, NETGEN would merge event nodes 1015 and 1210 to form the activities:



PREDECESSOR	SUCCESSOR	DESCRIPTION
1010	1015	X
1015	1215	Y

In other words, the successor of the leading activity replaces the predecessor of the following activity. For any subsequent tasks that interfaced to the original predecessor event of Y, i.e., 1210, NETGEN automatically reassigns them to the new predecessor event, 1015.

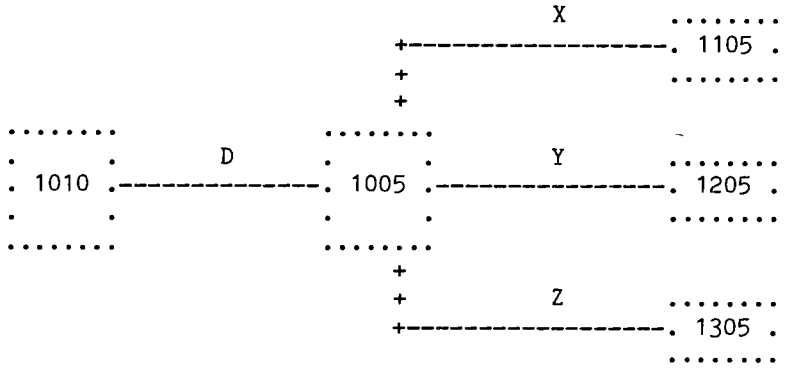
In the case of multiple F/A type activities following an activity, NETGEN replaces all their predecessors with the successor of the activity that they follow; i.e., assume activities X, Y, Z are all F/A Type to activity D. Let the task numbers for this example be as follows:

Task	Description
10.0	D
11.0	X
12.0	Y
13.0	Z

NETGEN would first create the activities:

PREDECESSORS	SUCCESSOR	DESCRIPTION
1000	1005	D
1100	1105	X
1200	1205	Y
1300	1305	Z

Then, NETGEN would merge the predecessors of X, Y, Z with the successor of D to form:



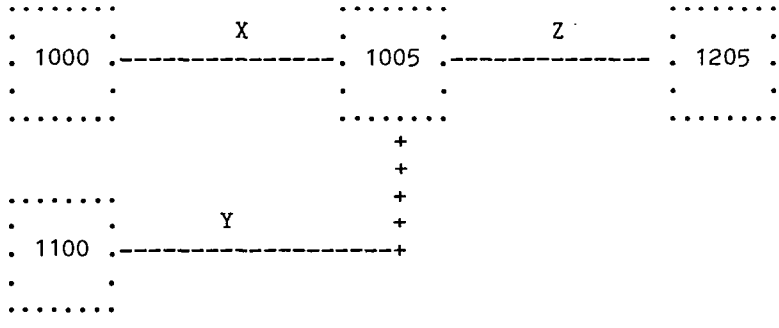
Any other activity that referenced the predecessor node to either X, Y, or Z would be tied to the successor node for D, 1005. Hence, the final network would appear as:

PREDECESSOR	SUCCESSOR	DESCRIPTION
1000	1005	D
1005	1105	X
1005	1205	Y
1005	1305	Z

As a last example on the F/A type, assume that two different tasks are immediately followed by the same single task. Let activity Z be a following activity to both X and Y. Let the task numbers be:

TASK	DESCRIPTION
10.0	X
11.0	Y
12.0	Z

NETGEN would create the following path structure:



In this case, NETGEN replaced the predecessor of Z with the successor of X, and then tied Y to that node. Because of the order of original tasks, the X to Z path was established first, thus preventing the node 1005 from being overwritten by the successor to Y.

The order of tasks in NETGEN is very important. In all the examples given here if task X is related to Y such that X goes first in time, then X is the constraining task, and Y is the constrained task. In other words, X must occur (start or end) before Y can occur (start to end), and X constrains the action of Y; so on the input deck to NETGEN, the task X information is placed ahead of the task Y information. In general the first activity (on input) is considered to be the Project (network) start. The format for coding this data will be detailed in the next section.

Activity duration is considered to be the delta (time) between the start date and the end date. NETGEN computes the duration from the dates supplied on input for the activities. When computing the durations NETGEN uses the

appropriate date (start or end) from the preceding activity for the interface start date , and the appropriate date (start or end) from the succeeding activity for the interface end date . The duration for these is then calculated by subtracting the two values. If NETGEN detects a negative value for any of the computed durations, it prints a warning message and resets the value to zero.

Section 3
RUNNING THE NETGEN PROGRAM

INPUT DESCRIPTION

NETGEN interfaces with the PPARS Batch and EZPERT programs by producing input files for them. NETGEN will accept any valid PPARS Batch control cards, as well as the valid EZPERT PRENET (LOGIC) mode control cards. The input to NETGEN consists of the control cards and the data cards. The control cards are the standard PPARS Batch cards, while the data cards contain such information as the task number, start/end dates, duration, resources, description, etc. The real difference between the data cards for PPARS Batch and NETGEN is that NETGEN uses a two-card format, the "A" and the "B" card. The "A" card contains the information unique to a particular task, while the "B" card contains the other task numbers and their type of relationship to the given task. The layout of the "A" and "B" cards is shown in Figures 1 and 2. The fields for these cards are as follows:

"A" Card

Col	Item	# of Cols
1	A	1
2	Activity Code	1
3-4	Master Schedule	2
5-18	Task Number	14
19-24	Start Date	6
25-30	End Date	6
31-34	Duration	4
35-38	Resources	4
39-72	Description	35
73	EZPERT Milestone Indicator	1
74-80	Organization	7

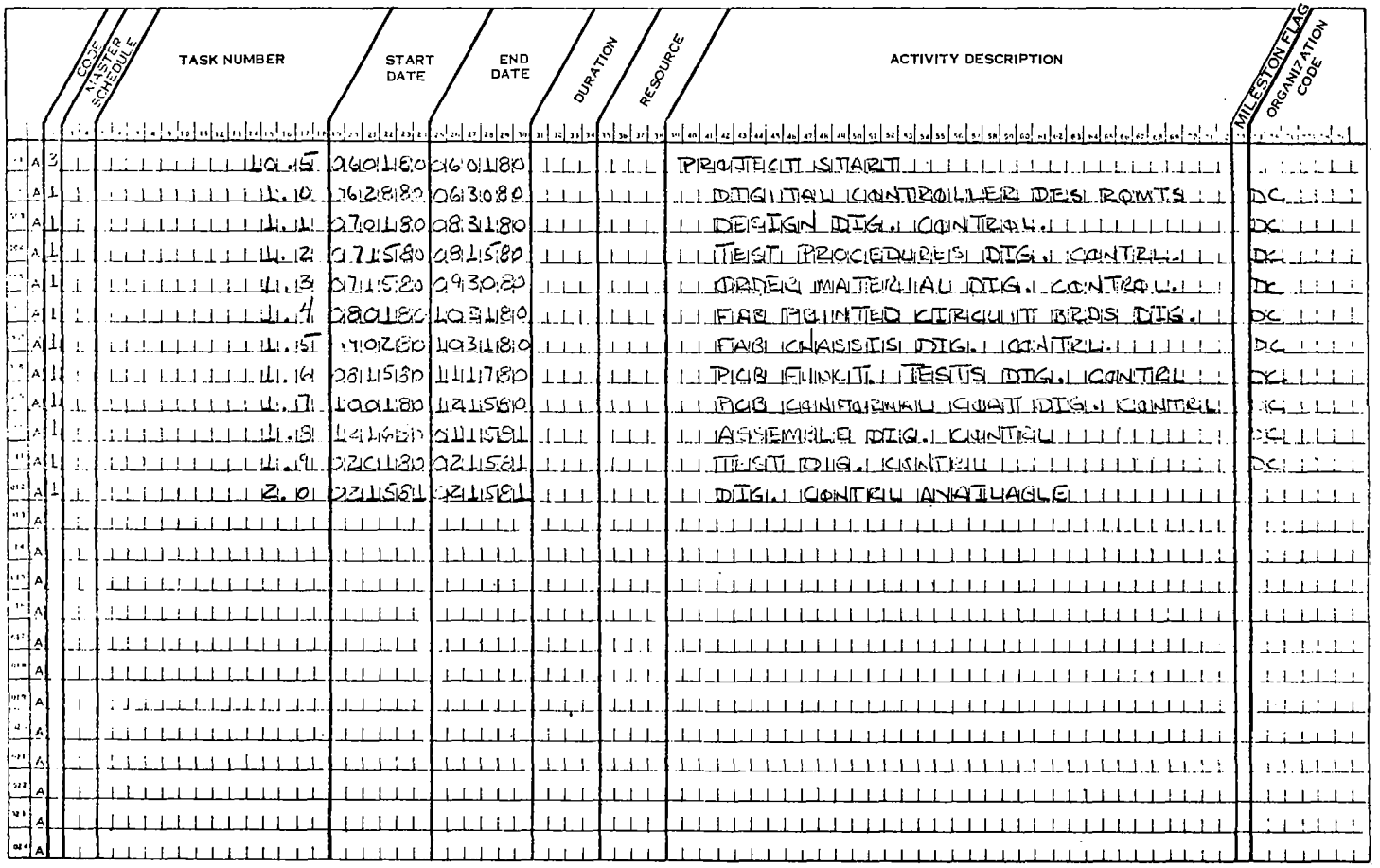
"B" Card

Col	Item	# of Cols
1	B	1
2	blank	1
3-16	Task # from the "A" card	14
17-18	Type of constraint 1	2
19-32	Constrained task #1	14
33-34	Type of constraint #2	2
35-48	Constrained task #2	14
49-50	Type of constraint	2
51-64	Constrained task #3	14
65-66	Type of constraint #4	2
67-80	Constrained task #4	14

LANGLEY RESEARCH CENTER

PPARS/NETGEN

"A" CARD CODING FORM



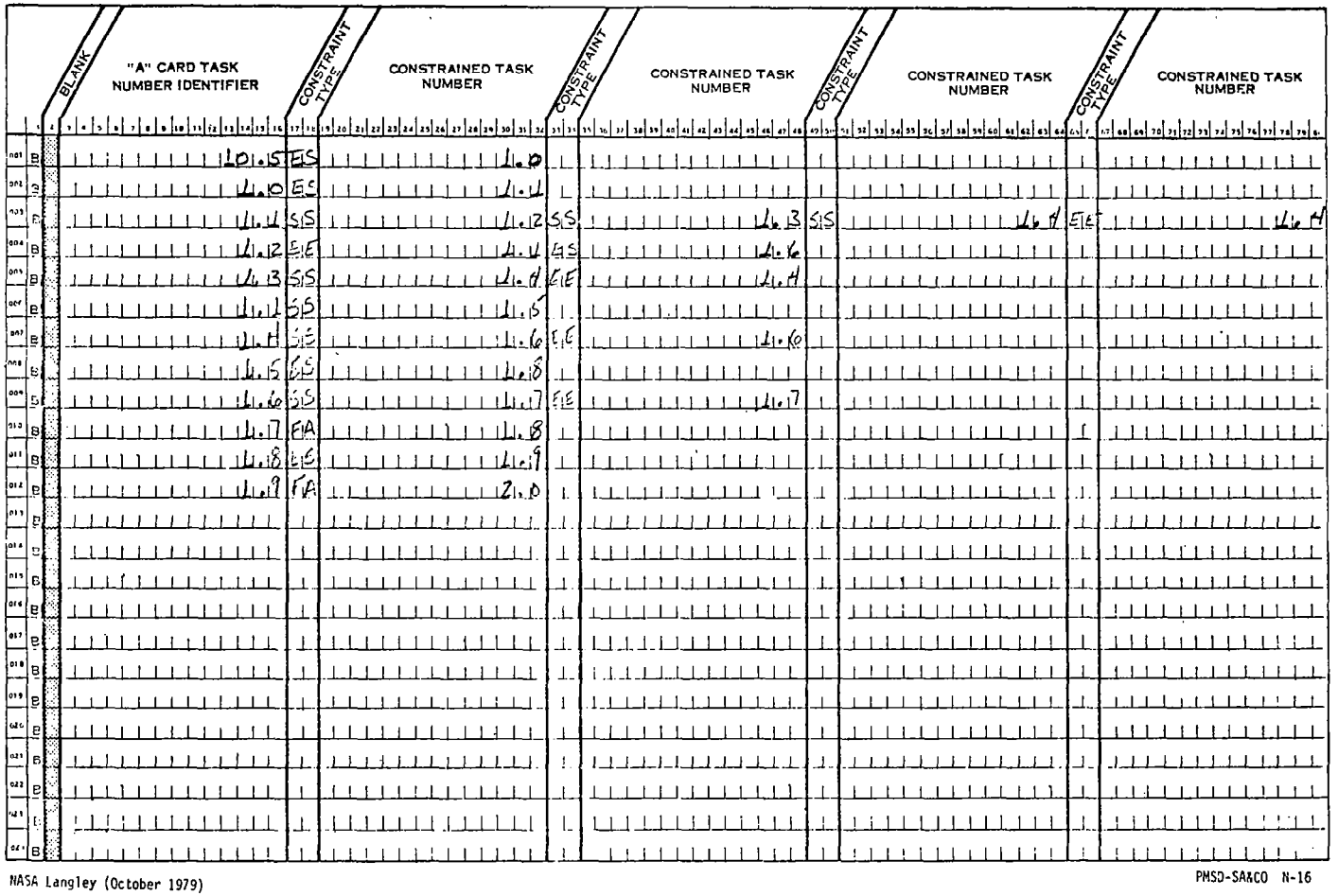
NASA Langley (October 1979) PMSD-SA&CO N-17

FIGURE 1

LANGLEY RESEARCH CENTER

PPARS/NETGEN

"B" CARD CODING FORM



NASA Langley (October 1979)

PMSD-SA&CO N-16

FIGURE 2

The presence of the "A" card number on the "B" card is for sorting purposes only, and allows the user to code up all the "A" cards, and then all the "B" cards and let NETGEN do the subsequent data matching. Any "A" card can have a maximum of five (5) "B" cards associated with it, and since there can be four (4) constrained tasks per "B" cards, this provides the user with up to 20 constrained tasks. If any activity is the ending (last) task of a project, it won't have any constrained tasks, and won't require any subsequent "B" cards.

NETGEN is programmed to compute activity durations based on a five-(default) or seven-day workweek. These are specified by the *5DAY or *7DAY cards in the input. Most task charts show the activity extending over one or more weeks, and even though these may be five-day work weeks, the total time estimate, as drawn on the bar charts usually includes weekends. Hence, it is recommended that the user input the *7DAY card in the input deck when running NETGEN so that the resulting dates match up with those on the original bar chart. It is also recommended that the user input both the start and end dates, since these are generally shown on the bar charts. NETGEN computes the duration from these. However, when the user leaves out the start or end dates, NETGEN will post a "000000" in the date field as a warning. This will cause problems when the resulting files (with these entries) are sent to PPARS Batch, and will cause a fatal error in the EZPERT processing.

OUTPUT DESCRIPTION

There are several portions to the output listing printed for the user after executing the NETGEN program. The first section, fig. 3, on the printout is a listing of the original input cards as set up by the user, and any diagnostics. There is also some information concerning the number and types of constraints detected by the NETGEN program and information concerning those activities with negative duration (fig. 4); this will indicate that the start/end dates were reversed by the user, or the duration for the interface constraint was negative because of the inputted dates. All negative durations are set to zero. There is also information concerning activities that could not be connected with interface activities. Lastly, NETGEN produces a complete listing of all the activities and interface constraints sorted by the predecessor (fig. 5). This report is nearly identical to REPORT 1 from PPARS Batch program; i.e., a complete listing of the network. The work tasks are given in the leftmost columns, except on interface constraints. On these

```
          *** NASA - LRC    PPARS ***
          NETWORK  GENERATION  PROGRAM
                   - NETGEN -
        **  INPUT  CONTROL  CARDS  **


     *TITLE(NETGEN SAMPLE)
     *7DAY
     *DATE(06/01/80)
     *REPORT1,2,3,4,15
     *NETWORK
```

```
                NASA - LRC  NETWORK  GENERATION  PROGRAM
                     **  INPUT  TASK  CARDS  **

A 3              10.5 060180 060180          PROJECT START
A 1               1.0 062680 063080            DIGITAL CONTROLLER DES RQMTS      DC
A 1               1.1 070180 083180            DESIGN DIG. CONTROL.              DC
A 1               1.2 071580 081580            TEST PROCEDURES DIG. CONTRL.      DC
A 1               1.3 071580 093080            ORDER MATERIAL DIG. CONTRL.       DC
A 1               1.4 060180 103180            FAB PRINTED CIRCUIT BRDS DIG.CNT  DC
A 1               1.5 090280 103180            FAB CHASSIS DIG. CONTRL.          DC
A 1               1.6 081580 111780            PCB FUNCT. TESTS DIG. CONTRL      DC
A 1               1.7 100180 121580            PCB CONFORMAL COAT DIG. CONTRL.   DC
A 1               1.8 121680 011581            ASSEMBLE DIG. CONTRL              DC
A 1               1.9 020181 021581            TEST DIG. CONTRL                  DC
A 1               2.0 021581 021581            DIG. CONTRL AVAILABLE             DC
B          10.5ES             1.0
B           1.0ES             1.1
B           1.1SS             1.2SS          1.3SS          1.4EE          1.4
B           1.2EE             1.1ES          1.6
B           1.3SS             1.4EE          1.4
B           1.1SS             1.5
B           1.4SS             1.6EE          1.6
B           1.5ES             1.8
B           1.6SS             1.7EE          1.7
B           1.7FA             1.8
B           1.8ES             1.9
B           1.9FA             2.0
```

FIGURE 3

** INPUT DATA SUMMARY **

THE NUMBER OF ORIGINAL ACTIVITIES :	31
THE NUMBER OF END-TO-END CONSTRAINTS :	5
THE NUMBER OF START-TO-START CONSTRAINTS:	7
THE NUMBER OF END-TO-START CONSTRAINTS :	5
THE NUMBER OF START-TO-END CONSTRAINTS :	0
THE NUMBER OF F/A CONSTRAINTS :	2

NASA - LRC NETWORK GENERATION PROGRAM
** INTERFACE ANALYSIS **

FOLLOWING ACTIVITIES - F/A TYPE

PRED.	SUCC.	START	END
170	175	100180	121580
190	195	020181	021581

NASA - LRC NETWORK GENERATION PROGRAM
** INTERFACE ANALYSIS **

END TO END ACTIVITIES - E/E TYPE

CODE	PRED.	SUCC.	START	END	DUR.	DESCRIPTION
1	115	145	083180	103180	61	E/E - 115 TO 145
1	125	115	081580	083180	16	E/E - 125 TO 115
1	135	145	093080	103180	31	E/E - 135 TO 145
1	145	165	103180	111780	17	E/E - 145 TO 165
1	165	175	111780	121580	28	E/E - 165 TO 175

FIGURE 4

NASA - LRC NETWORK GENERATION PROGRAM

FINAL NETWORK

AC	MS	ORIGINAL WBS#	PRED.	SUCC.	START DATE	END DATE	DUR.	RES.	ACTIVITY DESCRIPTION	ORGAN.
1		1.0	100	105	062880	063080	0.3		DIGITAL CONTROLLER DES RQMTS	DC
1		0	105	110	063080	070180	0.1		E/S - 105 TO 110	
1		1.1	110	115	070180	063180	8.7		DESIGN DIG. CONTROL.	DC
1		0	110	120	070180	071580	2.0		S/S - 110 TO 120	
1		0	110	130	070180	071580	2.0		S/S - 110 TO 130	
1		0	110	140	070180	060180	4.4		S/S - 110 TO 140	
1		0	110	150	070180	090280	9.0		S/S - 110 TO 150	
1		0	115	145	063180	103180	8.7		E/E - 115 TO 145	
1		1.2	120	125	071580	081580	4.4		TEST PROCEDURES DIG. CONTRL.	DC
1		0	125	115	081580	083180	2.3		E/E - 125 TO 115	
1		0	125	160	081580	081580	0.0		E/S - 125 TO 160	
1		1.3	130	135	071580	093080	11.0		ORDER MATERIAL DIG. CONTRL.	DC
1		0	130	140	071580	080180	2.4		S/S - 130 TO 140	
1		0	135	145	093080	103180	4.4		E/E - 135 TO 145	
1		1.4	140	145	080180	103180	13.0		FAB PRINTED CIRCUIT BRDS DIG.CNT	DC
1		0	140	160	080180	081580	2.0		S/S - 140 TO 160	
1		0	145	165	103180	111780	2.4		E/E - 145 TO 165	
1		1.5	150	155	090280	103180	8.4		FAB CHASSIS DIG. CONTRL.	DC
1		0	155	175	103180	121680	6.6		E/S - 155 TO 175	
1		1.6	160	165	081580	111780	13.4		PCB FUNCT. TESTS DIG. CONTRL	DC
1		0	160	170	081580	100180	6.7		S/S - 160 TO 170	
1		0	165	175	111780	121580	4.0		E/E - 165 TO 175	
1		1.7	170	175	100180	121580	10.7		PCB CONFORMAL COAT DIG. CONTRL.	DC
1		1.8	175	185	121680	011581	4.1		ASSEMBLE DIG. CONTRL	DC
1		0	185	190	011581	020181	2.4		E/S - 185 TO 190	
1		1.9	190	195	020181	021581	2.0		TEST DIG. CONTRL	DC
1		2.0	195	205	021581	021581	0.0		DIG. CONTRL AVAILABLE	DC
3		10.5	1050	1055	060180	060180	0.0		PROJECT START	
1		0	1055	100	060180	062880	3.9		E/S - 1055 TO 100	

FIGURE 5

interface constraints, the work task value is set to zero, the description of their type is given in the activities description field and reads "E/S : XXXXXXX to XXXXXXX", etc.

DECK SETUP

The data is keypunched according to the data field layout shown in Figures 1 and 2. A sample of the PPARS control cards that must precede the "A" and "B" data cards are:

```
*TITLE(.............)
*DATE(../../..)
*REPORT
*NETWORK

A...................
A...................
A...................
(A and B data cards)
B...................
B...................
```

The single control card that follows this data is the *ENDBATCH card.

Having keypunched the cards, process them through the NETGEN program using the Cyber Control Language (CCL) procedure, RUNNET. The deck setup would be as follows:

```
NETJOB,T100,CM120000.          BINXX  J. USER
USER,nnnnnnn.
CHARGE,xyxyxyxy
COPYBR,INPUT,TAPE79.
SAVE,TAPE79=TASKDAT.
REWIND,TAPE79.
BEGIN,,RUNNET.
SAVE,TAPE29=NEWNETW.
EXIT.
7/8/9/
*TITLE(.......
*7DAY
*DATE(.......
```

```
  :          (PPARS Batch Control Cards)
  :
  :
*NETWORK
A........
A........
  :        (NETGEN Data Cards)
  :
B.......
B.......
  :
  :
*ENDBATCH
7/8/9
6/7/8/9
```

The idea here is that the RUNNET procedure file is looking for its input on a local file called TAPE79. The contents of procedure RUNNET would be as follows:

```
.PROC,RUNNET.
COMMENT.   *** PROC. FILE RUNNET ***
COMMENT.
REWIND,TAPE79.
GET,LGO=NETBIN/UN=663250N.
MAP,OFF.
GET,NETSGDR.    (Segmentation directives as shown
                  in Appendix)
SEGLOAD(B=NETABS,I=NETSGDR).
LDSET(PRESET=ZERO)
LOAD,LGO.
NOGO.
RETURN,LGO.
NETABS,TAPE79,OUTPUT.
RETURN,NETABS.
RETURN,TAPE13,TAPE15,TAPE9,TAPE19.
REWIND,TAPE8,TAPE17.
REVERT.  *** END RUNNET ***
```

NETGEN produces a PERT-type network on the file TAPE8, that can be saved for later use in the PPARS Batch program. The file TAPE17 can also be saved or

used as input to the EZPERT program to produce the EZPERT PRENET(LOGIC) mode diagrams. Also note that the user can input any of the valid EZPERT control cards if he desires, although NETGEN produces the necessary instructions (along with the data) on the file TAPE17.

Section 4
PROGRAM DESCRIPTION

NETGEN is written in FORTRAN and contains one main program, one block data subprogram and 20 subroutines. NETGEN can be executed in a normal manner or as a segmented program via the CDC SEGRES Loader program. NETGEN tree structure is shown in Figure 6. The segmentation directives for the program are given in the Appendix. All files used in NETGEN are declared in the main program, while all common blocks are declared in or below subroutine DRIVER. This technique of programming allows the I/O buffers to be loaded above the labeled common blocks. There is no unlabeled, or "blank", COMMON in NETGEN.

PROGRAM MODULES

NTWGEN

Main program; used to declare all files used in NETGEN. Program NTWGEN only calls subroutine DRIVER.

DRIVER

Subroutine; contains nearly all of the common blocks used. DRIVER controls the calling of the two main branches (see Figure 1) in the program; i.e., DRIVER calls READCD first for reading/editing and preparing the data base. Then DRIVER calls TIEUP for the subsequent network path construction, data sorting and output generation.

BKDATA

Subprogram; initializes the variables and arrays used in NETGEN.

MOVBIT

Subroutine; is called by subroutines RDACT, PRECON, SUCONV, PATHS, and UNDATE to pack or unpack data on the 60 bit words.

NETGEN TREE DIAGRAM

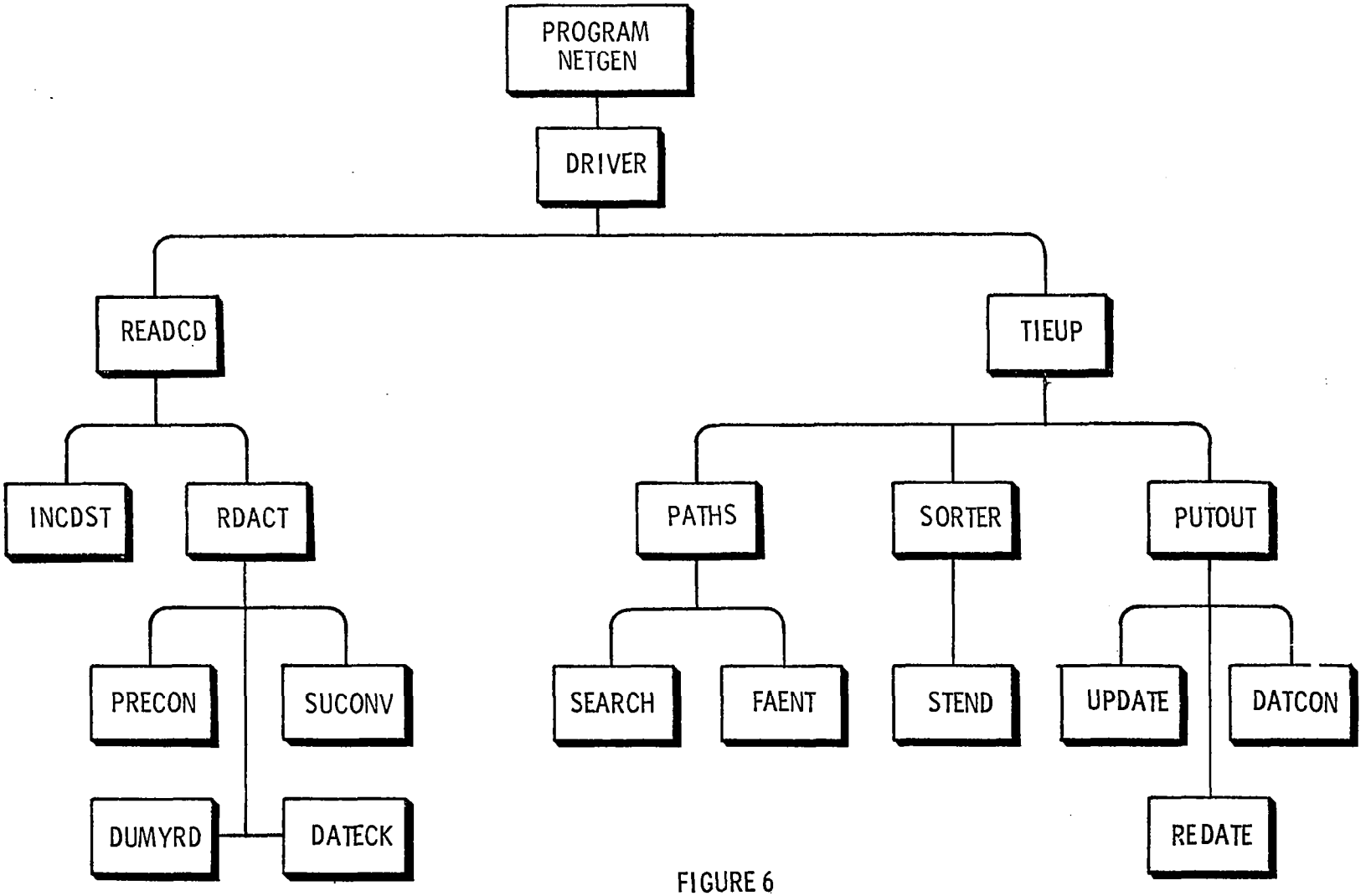


FIGURE 6

READCD

Subroutine; reads the input file down to, and including, the *NETWORK card. READCD controls the processing of the first main branch and calls INCDST and RDACT.

KONVERT

Subroutine; is called by INCDST, PRECON, and SUCONV to convert a task number from character data into a 7-digit integer.

INCDST

Subroutine; reads and rearranges the activity cards so that each "A" card is followed by its associated "B" card(s). The "A" and "B" cards are initially read by INCDST and written to the files TAPE13 and TAPE15, respectively. These are then merged and written to the file TAPE17.

RDACT

Subroutine; reads TAPE17 as created by INCDST and creates the data base on the file TAPE9. RDACT calls routines PRECON,SUCONV, and DUMYRD to handle the "A", "B", and any unrecognized cards, respectively. The information processed from any "A" card and its associated "B" card(s) is temporarily stored in the array, INFOTBL. RDACT dumps this information to TAPE9 when a new "A" card is read. RDACT continues to read from TAPE17 until an *ENDBATCH or an end of file (EOF) is detected.

DUMYRD

Subroutine; called by RDACT to print out any invalid or unrecognized cards. DUMYRD then continues reading and writing the cards on TAPE17 until it encounters an "A" card, *ENDBATCH or end of file.

PRECON

Subroutine; called by RDACT to analyze the "A" cards. PRECON uses routine KONVERT to create an integer value (predecessor) from the task number, and routine DATECK to edit/convert the activity date information. For each valid "A" card, PRECON increments the counter, IPRCNT, and stores the "A" card data

in the first row of the array INFOTBL; columns 1 to 8 of INFOTBL contain the original card data, as well as the new predecessor/successor and duration. PRECON also packs the activity's date information as returned from routine DATECK, into the array DATTBL, at location DATTBL (IPRCNT), via subroutine MOVBIT.

DATECK

Subroutine; called by PRECON to edit/verify/convert the input dates on the "A" card. If only two of the three date items are given, DATECK will compute the third. Dates are given as integer values from the base year, January 1, 1970 and the duration is computed on a 5-day work week, unless the user inputs the *7DAY card.

SUCONV

Subroutine; called by RDACT to edit/convert the "B" card data. Since there can be a max of four (4) tasks on the "B" card, SUCONV moves all the data in the array INFOTBL down one row, stores the new task in column 9 of this next row, and increments the counter IPRCNT. Then SUCONV brings the information in the array DATTBL down by one row, increments the specific constraint type counter, KEXnn and packs the value IPRCNT into the array CONSTBL at location KEXnn. SUCONV skips over the blank fields on the "B", but if a task is given without a constraint type, it is assigned the F/A type.

TIEUP

Subroutine; called by DRIVER to control the processing in the second branch of the program. TIEUP calls subroutines PATHS, SORTER and PUTOUT for the network construction, network sorting and data output, respectively.

PATHS

Subroutine; called by TIEUP to construct the logic network. PATHS reads the file TAPE9 into the array, EVENTBL. The address for each activity according to constraint type is retrieved from the CONSTBL array by PATHS via subroutine MOVBIT. The key values in the EVENTBL array are a task's predecessor, successor, and the predecessor of the constrained task. PATHS calls subroutine SEARCH to go find this constrained task elsewhere in EVENTBL, so that the two tasks can be joined. The F/A tasks (if any) are constructed first, and for each activity affected, PATHS stores a flag value of -100 in

EVENTBL. At the conclusion of the F/A analysis, PATHS calls subroutine FAENT to detect all start/end events in the F/A group, as well as any other "strays", i.e., those activities not joined to any other activities and flags them with a -100. PATHS then constructs the E/E, S/S, E/S, and S/E interface activities. As these are formed, they are written out to file TAPE19. When all the interface analysis is done, PATHS merges the flagged data with the information on file TAPE19 and rewrites the data base to file TAPE9.

SEARCH

Subroutine; called by PATHS to scan the EVENTBL array and find the activity whose predecessor was given (as the constrained activity) elsewhere in EVENTBL by PATHS. SEARCH returns the location of the match to PATHS.

FAENT

Subroutine; called by PATHS to scan only the F/A activities and flag the end events. FAENT then searches for the "stray" activities - those not joined to any others - and flags them.

SORTER

Subroutine; called by TIEUP to sort the data base. The algorithm used in a Bubble Sort and the major sort key is predecessor, with the minor key, successor.

STEND

Subroutine; called by SORTER to find all start/end events in the sorted networks. Start events are given a code 3 (if they have an original code 1) and end events are given a temporary code of 9, so that the end date will be included when the activity is written to TAPE8 for PPARS Batch. This code is reset to 1 by routine PUTOUT before writing the record. This technique forces all end events to be scheduled for PPARS Batch.

PUTOUT

Subroutine; called by TIEUP to write the files going to EZPERT and PPARS Batch. PUTOUT also writes the final output for NETGEN showing the original

task, the new predecessors and successors, etc. PUTOUT calls routines UNDATE, DATCON and REDATE for date conversion during the writing of the output files.

UNDATE

Subroutine; called by PUTOUT to unpack the activity date information from the DATTBL array. UNDATE calls routine MOVBIT to perform the unpacking, but then tests the results for negative duration.

DATCON

Subroutine; called by PUTOUT to convert the activity date information into a format (DDMMYY) acceptable to EZPERT.

REDATE

Subroutine; called by PUTOUT to convert the activity's duration into a two-part integer consisting of weeks and days.

FILES IN NETGEN

There are eight files declared in the PROGRAM statement of NTWGEN. These are:

TAPE5	TAPE13
TAPE6	TAPE15
TAPE8	TAPE17
TAPE9	TAPE19

A brief description of these files is as follows:

TAPE5

TAPE5 is equated to INPUT on the PROGRAM statement.

TAPE6

TAPE6 is equated to OUTPUT on the PROGRAM statement.

TAPE8

TAPE8 is created by routine READCD and contains the control cards input by the user. Subroutine PUTOUT later writes the network activities created by NETGEN out to TAPE8, following the *NETWORK card. PUTOUT also writes the *ENDBATCH card out to TAPE8 as the last record.

TAPE9

TAPE9 is initially written by routine RDACT, and contains the data from the "A" and "B cards. TAPE9 is the main data base for NETGEN and contains 15 fields, for a total length of 101 characters. The layout for TAPE9 is:

1) 2) 3) 4) 5) 6) 7) 8) 9) 10) 11) 12)

1X, A1, A2, A10, A4, I7, I7, A6, A6, I4, 4A10, A6, I7

1) Activity Code
2) Master Schedule
3) First 10 characters of the WBS value
4) Last 4 characters of the WBS value
5,6) Predecessor and Successor
7,8) Start and end date
9) Duration
10,11) Description and Organization
12) Integer "0" for "A" cards or the predecessor value of the constrained task from the "B" cards.

Thus, for every valid "A" card, RDACT writes one record out to TAPE9 with a 0 for element 12. Then, as RDACT reads (and accepts) the "B" cards associated with that particular "A" card, the same information is repeated, but with the predecessor stored in item 12. This process continues until the next "A" card is encountered. TAPE9 is subsequently read by routine PATHS to form the interface activities. Routine SORTER also reads TAPE9, but sorts the data and then writes it back to TAPE19.

TAPE13

TAPE13 is created by routine INCDST and contains all the "A" cards.

TAPE15

TAPE15 is created by routine INCDST and contains all the "B" cards. TAPE15 and TAPE13 are merged in INCDST to form TAPE17

TAPE17

TAPE17 is initially created by INCDST and contains the intermixed "A" and "B" cards. It is read by routine RDACT during the creation of TAPE9. Routine PUTOUT completely overwrites TAPE17 to contain the data for the EZPERT program.

TAPE19

Scratch file, used throughout the program for temporary storage.

COMMON BLOCKS and VARIABLES

All common blocks used in NETGEN are FORTRAN labeled and declared at, or below, subroutine DRIVER. The common blocks and their contents are as follows:

ACTREC

Declared in subroutine DRIVER and contains three (3) integer arrays - CARD(80), CRDCOD(2), and CTYPE(5). The CARD array is used to hold data coming in from or going out to the various files throughout the program. The CRDCOD holds the two (2) card codes, "A" or "B" as variables, and the CTYPE array holds the five (5) valid constraint types - FA, ES, EE, SS, and SE as variables. CRDCOD and CTYPE are defined in subprogram BKDATA.

CONTROL

Declared in subroutine DRIVER and contains the integer array PERTCD along with the logical variables I5DAY and I7DAY. Array PERTCD is defined in subprogram BKDATA to contain the first five (5) characters of the following PPARS Batch control cards: *TITLE, *5DAY, *7DAY, *NETWORK, and *ENDBATCH. The values for I5DAY and I7DAY are also initialized in BKDATA to .TRUE. and .FALSE. respectively. These may be reset in routine READCD when a *7DAY card is detected.

COUNTS

Declared in subroutine DRIVER. COUNTS contains the various counters/ pointers used throughout NETGEN. These variables are as follows:

IPRCNT - Counter, incremented by PRECON for each "A" card accepted, and by SUCONV for each constraint given on a "B" card. IPRCNT equals the number of records written to file TAPE9 by RDACT.

INCRE - Total number of activity cards read.

KEXEE - Total number of E/E constraints.

KEXSS - Total number of S/S constraints.

KEXES - Total number of E/S constraints.

KEXSE - Total number of S/E constraints.

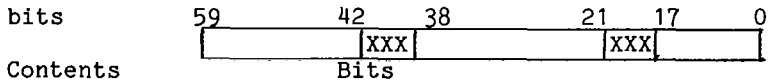
KEXFA - Total number of F/A constraints.

LOCDUM - Scratch variable.

ITOTAL - Total number of activities after the network construction. Incremented by PATHS.

LABEL

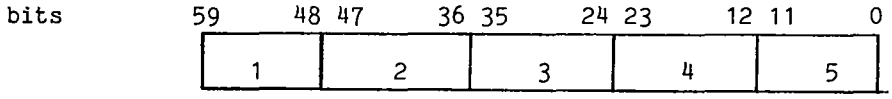
Declared in subroutine DRIVER and contains the integer arrays DATTBL, CONSTBL, and DESTBL (scratch array). The date information for an activity is initially determined by routine DATECK, but is packed (start, end, duration) into one word by routine PRECON into the array DATTBL. The bit pattern of any word in the array DATTBL is as follows:



Contents	Bits
1) start date	59-42
2) end date	38-21
3) duration	17-0

Bits 18-20 and 39-41 are unused. For a given activity these values are stored at location DATTBL (IPRCNT).

The constraint information is kept as a series of pointers to the information written to TAPE9 by RDACT; i.e., as an "A" card is read or as one constraint on a "B" card is read, its location out on TAPE9 will be the value, IPRCNT. When routine SUCONV analyzes the constraint on a "B" card, IPRCNT is incremented and packed into KEXnn word of the CONSTBL array, in the portion set aside for that particular constraint type. The layout of the bit pattern is any word of the array CONSTBL is:



Contents	Bits
1) IPRCNT for F/A	48-59
2) IPRCNT for E/E	36-47
3) IPRCNT for S/S	24-35
4) IPRCNT for E/S	12-23
5) IPRCNT for S/E	0-11

MISC

Declared in DRIVER and used throughout the program. MISC contains the following integer variables:

BLANK - Initialized in BKDATA to 1H.

IERR - Error return, used in routine DRIVER to check if the input is valid.

ZERO - Initialized in BKDATA to 1HO.

MONTHS

Declared in DRIVER and contains date information.

MONTBL - Array,initialized in BKDATA to contain the total number of days in each month.

DAYTOT - Integer array, initialized in BKDATA to hold the cumulative days in the year by the end of each month.

ISTART - Array, used by DATECK in converting the activity start date into an integer.

IEND - See ISTART

BYR - Base year, initialized by BKDATA to 70 (1970).

ACTIME - Integer, used to hold the activity duration.

NEW

Declared in routine PATHS to hold information about the two activities being joined in the network construction.

NEWACT - Array,holds the predecessor, successor, dates, etc., on the new interface activity. The contents of NEWACT are written to TAPE19 by PATH.

NPRED - Predecessor of the constrained task.

NSUCC - Successor of the constrained task.

MATLOC - Location in EVENTBL of the constrained task; MATLOC is computed in routine SEARCH.

NOFIND - Logical, set to .TRUE. if the constrained task is not found by SEARCH.

ENDFLG - Scratch value.

IDESI - IDES4 - Initialized in PATHS to hold the activity descriptions assigned to the constraints.

ITOTFA - Total F/A activities after the network construction.

PETE

Declared in routine RDACT and contains the information being processed on any "A" card and its associated "B" cards. The contents of PETE are:

INFOTBL - Array; the first row (cols. 1-8) is filled with the "A" card data via routine PRECON. Then, as the tasks are analyzed on the "B"

cards, the information is copied down into the next successive row for each task on the "B" card by routine SUCONV. The task number as determined by SUCONV is stored in column 9 of the associated row.

INC - Counter; set and incremented by RDACT to determine how may rows were filled by routine SUCONV.

Declared in DRIVER and used in INCDST. INCD contains two variables, KTOTA and KTOTB that keep a count of the number of "A" and "B" cards, respectively, read by INCDST.

OLDBLK

Declared in PATHS and used to hold information for the network construction. OLDBLK contains the integer array EVENTBL, which is filled by routine PATHS.

OLDTWO

Declared in SORTER and contains the activities to be sorted. Routine SORTER reads the data from file TAPE9 into the integer array IACTBL and sorts the network; the major key is predecessor and minor key, successor. The algorithm used is a Bubble Sort.

Appendix

SEGMENTATION LOADER DIRECTIVES FOR NETGEN

```
*
*
 ROOT      TREE      NTWGEN-(TR1,TR2)
           INCLUDE   NTWGEN,DRIVER,BKDATA,MOVBIT
           GLOBAL    LABEL,ACTREC,MONTHS
           GLOBAL    COUNTS,MISC,CONTRL
*
*
*
 TR1       TREE      READCD-(INCDST,RDACT)
 READCD    INCLUDE   READCD,KONVERT
*
 INCDST    INCLUDE   INCDST
*
*
 RDACT     INCLUDE   RDACT,PRECON,DATECK
 RDACT     INCLUDE   SUCONV,DUMYRD
 RDACT     GLOBAL    PETE
*
*
*
 TR2       TREE      TIEUP-(PATHS,SORTER,PUTOUT)
 TIEUP     INCLUDE   TIEUP
*
*
 PATHS     INCLUDE   PATHS,UNDATE
 PATHS     GLOBAL    NEW,OLDBLK
*
 SORTER    INCLUDE   SORTER,STEND
 SORTER    GLOBAL    OLDTWO
*
 PUTOUT    INCLUDE   PUTOUT,DATCON,REDATE
*
*
                     END
```

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-3364		

4. Title and Subtitle	5. Report Date
A COMPUTER PROGRAM FOR THE GENERATION OF LOGIC NETWORKS FROM TASK CHART DATA	December 1980
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Henry E. Herbert	10. Work Unit No.
9. Performing Organization Name and Address Computer Sciences Corporation 3217 N. Armistead Avenue Hampton, Virginia 23666	11. Contract or Grant No. NAS1-16078
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Contract Monitor: John Hogge, NASA Langley Research Center
Technical Monitor: D. A. Wood, NASA Langley Research Center
Topical Report

16. Abstract

This paper presents the Network Generation Program (NETGEN) that creates logic networks from task chart data. NETGEN is written in CDC FORTRAN IV (Extended) and runs in a batch mode on the CDC 6000 and CYBER 170 Series Computers. Data is input via a two-card format and contains information regarding the specific tasks in a project. From this data, NETGEN constructs a logic network of related activities with each activity having unique predecessor and successor nodes, activity duration, descriptions, etc. NETGEN then prepares this data on two files that can be used in the Project Planning Analysis and Reporting System (PPARS) Batch Network Scheduling program and the EZPERT graphics program.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Critical Path Method PERT Networks Project Scheduling	UNCLASSIFIED - UNLIMITED Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	36	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1980