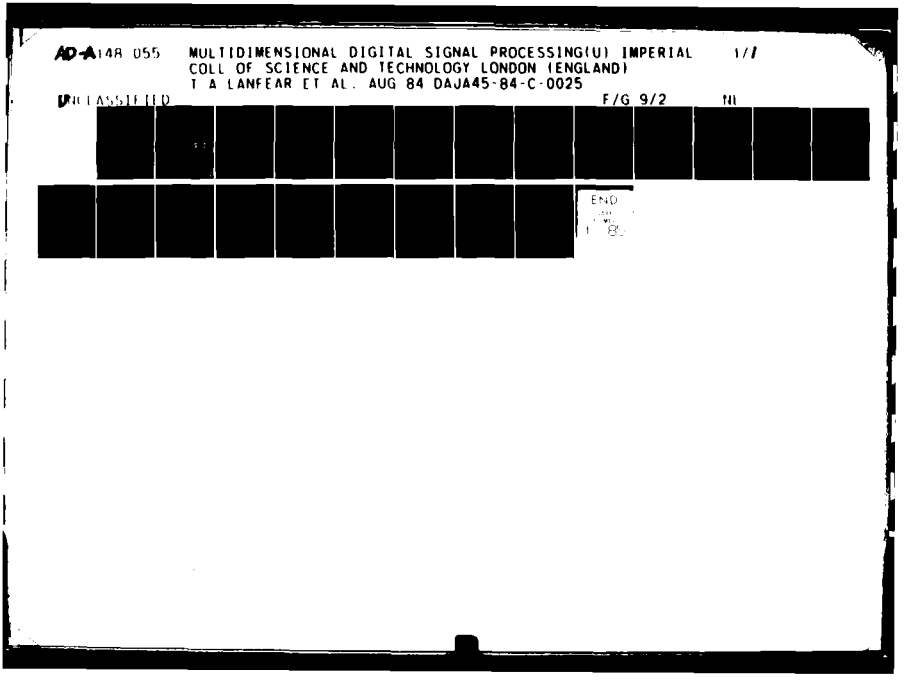
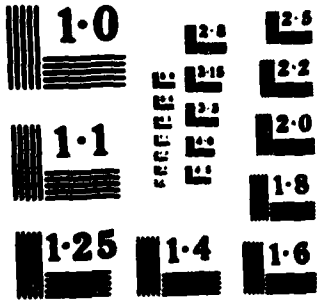

AD-A148 055
MULTIDIMENSIONAL DIGITAL SIGNAL PROCESSING(U) IMPERIAL
COLL OF SCIENCE AND TECHNOLOGY LONDON (ENGLAND)
T A LANFEAR ET AL. AUG 84 DAJA45-84-C-0025
1/1
UNCLASSIFIED
F/G 9/2
NL
END


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5

(13)

AD-A148 055

MULTIDIMENSIONAL DIGITAL SIGNAL PROCESSING

Final Technical Report

by

T. A. Lanfear

A. G. Constantinides

August 1984

DTIC ELECTE NOV 29 1984 S D A

United States Army

EUROPEAN RESEARCH OFFICE OF THE U.S. ARMY

London, England.

CONTRACT NUMBER DAJA45-84-C-0025

Imperial College of Science and Technology- A. G. Constantinides

Approved for Public Release; distribution unlimited

DTIC FILE COPY

84 10 16 151

CONTENTS

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
Distribution/
Availability Codes
Dist Avail and/or
Special
A-1

1 Introduction

This project has been concerned with the development of software tools for the study, at the most general level, of the ALPS system architecture. The primary aim of these tools is to answer such questions as: is a signal processing application feasible with a given hardware configuration?; will the system degrade gracefully if some of the resources fail?; what is the effect on system performance of changes to details such as the number of resources available, the execution time of a resource etc., etc?

2 ALPS System Simulation

A program has been written in FORTRAN 77 which simulates the ALPS Signal Flow Architecture. It takes the following information into consideration:

1. The signal processing application expressed as a signal flow graph with the modifications to this notation which have been suggested at Imperial College.
2. The number and types of resources in the system.
3. The execution time of the resources.
4. The type of resource associated with each graph node.
5. The number of data busses and their data transfer rates, and
6. The rate at which data is presented to the input of the system.

The simulator then steps through time examining the state of the system and modifying it appropriately so as to simulate the allocation of resources to signal flow graph nodes and the transmission of data along busses. When the state of the system changes a summary of the new system state is printed. The

information presented is:

1 The time.

2 The state of the data busses.

3 The number of resources unused.

4 The bus utilisation as a percentage of the total time.

5 The status of the various resources which are currently in use.

A detailed description of the program and a users manual have been given in the First and Second Interim Reports of this project. A listing of the program is appended to this report together with a floppy written in UNIX tar format containing the program and the following example.

3 An Example

Figure 1 shows a typical signal processing flow graph with two inputs (nodes 1 and 4) and a single output (node 14). Node 0 is the added input node. The prioritised inputs are marked with solid lines and the other inputs with dotted lines.

The table below gives the resource descriptions of the graph.

resource type	execution time	graph node	number
A	250	1,6,7,9	2
B	50	3,10	2
C	12	8	1
D	100	5	1
E	100	14	1
F	25	13	1
G	10	12	1
H	50	11	2
I	1	4	1
J	10	2	1
memory	0	N/A	5

The other pieces of information required are:

bus transfer time = 20

number of busses = 1

From this information the input file can be generated. It is:

```
 14  11   1   20
  0  2  0  0  2  0  0  0  0  0  0  0  0  0  0
  0  0  2  1  0  0  0  0  0  0  2  0  0  0  0
  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0
  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0
  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  2  0  0  0  0  2
  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
250 50 12100100 25 10 50  1 10  0
  1 10  2  9  4  1  1  3  1  2  8  7  6  5
```

The output obtained from this example is shown on the following pages.

```
NAME OF FILE TO READ INPUT DATA
GIVE I/P RATE
GIVE TIME FOR FIRST INTERRUPT

TIME IS 0
BUS 1 IS FREE

RESOURCES UNUSED
  1  1  2  1  1  1  1  1  2  0  1  5

BUS UTILISATION
****%

RES NODE   TAG    END    MEM NIP SINK  NODES
  1    1      0      250  0   1   2   3 10
  2    4      0        1  0   1   5


TIME IS 1
BUS 1 IS BUSY

RESOURCES UNUSED
  1  1  2  1  0  1  1  1  2  1  1  5

BUS UTILISATION
 100%

RES NODE   TAG    END    MEM NIP SINK  NODES
  1    1      0      250  0   1   2   3 10
  2    5      0      121  0   1   6

TIME IS 121
BUS 1 IS BUSY

RESOURCES UNUSED
  1  0  2  1  1  1  1  1  2  1  1  5

BUS UTILISATION
  17%

RES NODE   TAG    END    MEM NIP SINK  NODES
  1    1      0      250  0   1   2   3 10
  2    6      0      391  0   1  11


TIME IS 250
BUS 1 IS BUSY

RESOURCES UNUSED
  1  1  1  1  1  1  1  1  2  1  0  4

BUS UTILISATION
  16%

RES NODE   TAG    END    MEM NIP SINK  NODES
  2    6      0      391  0   1  11
  3    2      0      280  0   1   3
  4   10      0   999999  0   1  11
  5    1      0      270  1   1   3
```

TIME IS 26
BUS 1 IS BUSY

RESOURCES UNUSED
1 1 0 1 1 1 1 1 2 1 1 4

BUS UTILISATION
21%

RES	NODE	TAG	END	MEM	NIP	SINK	NODES
2	6	0	391	0	1	11	
4	10	0	999999	0	1	11	
5	1	0	270	1	1	3	
1	3	0	999999	0	1	7	

TIME IS 300
BUS 1 IS BUSY

RESOURCES UNUSED
1 1 0 1 1 1 1 1 2 1 1 5

BUS UTILISATION
27%

RES	NODE	TAG	END	MEM	NIP	SINK	NODES
2	6	0	391	0	1	11	
4	10	0	999999	0	1	11	
1	3	0	370	0	2	7	

TIME IS 370
BUS 1 IS BUSY

RESOURCES UNUSED
1 0 1 1 1 1 1 1 2 1 1 5

BUS UTILISATION
27%

RES	NODE	TAG	END	MEM	NIP	SINK	NODES
2	6	0	391	0	1	11	
4	10	0	999999	0	1	11	
3	7	0	640	0	1	8	

TIME IS 391
BUS 1 IS BUSY

RESOURCES UNUSED
1 1 1 1 1 1 1 1 1 1 1 5

BUS UTILISATION
30%

RES	NODE	TAG	END	MEM	NIP	SINK	NODES
4	10	0	999999	0	1	11	
3	7	0	640	0	1	8	
1	11	0	999999	0	1	12	

```
TIME IS 640
BUS 1 IS BUSY

RESOURCES UNUSED
  1  2  1  0  1  1  1  1  1  1  1  5

BUS UTILISATION
  22%

RES NODE  TAG   END    MEM NIP SINK NODES
  4  10    0  999999  0   1   11
  1  11    0  999999  0   1   12
  2   8    0     672  0   1    9 14


TIME IS 672
BUS 1 IS BUSY

RESOURCES UNUSED
  1  1  1  1  1  0  1  1  1  1  1  5

BUS UTILISATION
  23%

RES NODE  TAG   END    MEM NIP SINK NODES
  4  10    0  999999  0   1   11
  1  11    0  999999  0   1   12
  3   9    0     942  0   1   10
  5  14    0  999999  0   1


TIME IS 942
BUS 1 IS BUSY

RESOURCES UNUSED
  1  2  1  1  1  0  1  1  1  1  1  5

BUS UTILISATION
  19%

RES NODE  TAG   END    MEM NIP SINK NODES
  4  10    0    1012  0   2   11
  1  11    0  999999  0   1   12
  5  14    0  999999  0   1


TIME IS 1012
BUS 1 IS BUSY

RESOURCES UNUSED
  1  2  2  1  1  0  1  1  1  1  1  5

BUS UTILISATION
  19%

RES NODE  TAG   END    MEM NIP SINK NODES
  1  11    0    1082  0   2   12
  5  14    0  999999  0   1
```

```
BUS 1 IS BUSY

RESOURCES UNUSED
  1  2  2  1  1  0  1  0  2  1  1  5

BUS UTILISATION
  20%

RES NODE  TAG   END    MEM NIP SINK NODES
  5  14     0  999999  0   1
  2  12     0    1112  0   1   13

TIME IS 1112
BUS 1 IS BUSY

RESOURCES UNUSED
  1  2  2  1  1  0  0  1  2  1  1  5

BUS UTILISATION
  21%

RES NODE  TAG   END    MEM NIP SINK NODES
  5  14     0  999999  0   1
  1  13     0    1157  0   1   14


TIME IS 1157
BUS 1 IS BUSY

RESOURCES UNUSED
  1  2  2  1  1  0  1  1  2  1  1  5

BUS UTILISATION
  22%

RES NODE  TAG   END    MEM NIP SINK NODES
  5  14     0    1277  0   2


TIME IS 2000
BUS 1 IS FREE

RESOURCES UNUSED
  1  1  2  1  1  1  1  1  2  0  1  5

BUS UTILISATION
  14%

RES NODE  TAG   END    MEM NIP SINK NODES
  1   1     1    2250  0   1   2   3 10
  2   4     1    2001  0   1   5


    TIME IS 2000

    WHAT DO YOU WISH TO DO?
    OPTIONS ARE:
      1 CHANGE INPUT RATE
      2 CHANGE NUMBER OF RESOURCES IN SYSTEM
      3 CHANGE EXECUTION TIME OF A RESOURCE
      4 CHANGE NUMBER OF BUSSES
      8 STOP PROGRAM
      9 CONTINUE SIMULATION
```

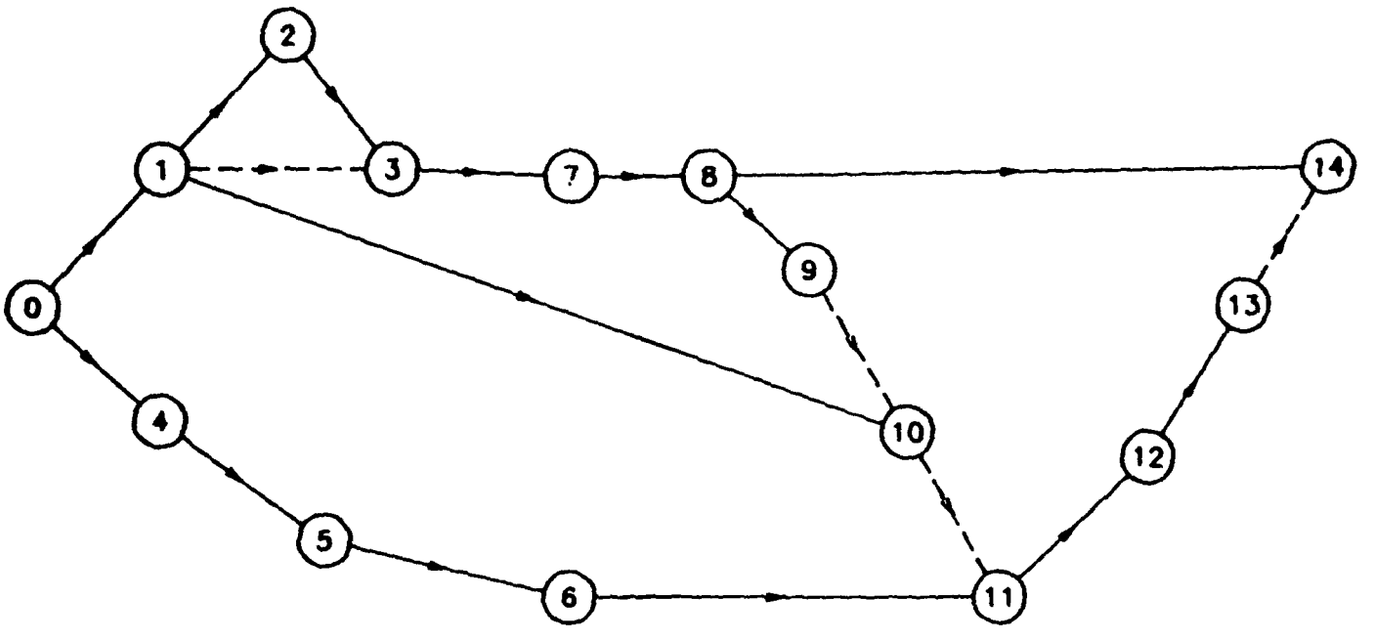


Fig 1 Signal Processing Graph

```
[illegible]
[illegible]

PARAMETER STATEMENT FOR [illegible] DIMENSIONS
PARAMETER (N1=[illegible],N2=[illegible],N3=[illegible],N4=[illegible],N5=[illegible])

COMMON /A/ [illegible](N1),[illegible](N1),POOL(N2),EXTIME(N2),TYPE(N1),BFREE(N3),
*RES(N4,5),CONN(N1,N1),NCONN(N1),NIP(N1),TRANS(N4,N1+1),
*NTRANS(N4),LIST(N4+1),BUTIL(N3,2)

[illegible] I = [illegible] FOR PRIORITY LINE I TO [illegible]
           =1 ORDINARY LINE
           =0 OTHERWISE
POOL(I)=NUMBER OF RESOURCES OF TYPE I
EXTIME(I)=EXECUTION TIME OF RESOURCE TYPE I
TYPE(I)=RESOURCE TYPE OF GRAPH NODE I (INPUT TYPE=0)
RES(I,[illegible])=ALLOCATION OF RESOURCE
   RES(I,1)=GRAPH NODE BEING EXECUTED ON RES(I)
   RES(I,2)=TIME TAG OF DATA BEING EXECUTED (INFINITY IF NO DATA
   RES(I,3)=TIME WHEN PROCESS WILL FINISH (INFINITY IF RES IS INACTIVE)
   RES(I,4)=1 IF RESOURCE I IS A MEMORY, 0 OTHERWISE
   RES(I,5)=NUMBER OF INPUTS WHICH HAVE BEEN SATISFIED
CONN(I,[illegible])=SET OF GRAPH NODES CONNECTED TO I
NCONN(I)=NUMBER OF NODES CONNECTED TO I
BFREE(I)=TIME WHEN BUS I BECOMES FREE
TRANS(I,[illegible])=SET OF GRAPH NODES TO WHICH RES(I) SHOULD TRANSMIT
NTRANS(I)=LENGTH OF SET TRANS(I,.)
LIST(I)=SET OF RESOURCES WHICH ARE DOING SOMETHING
NIP(I)=NUMBER OF I/P S TO GRAPH NODE I
NOW=THE CURRENT TIME


[illegible] STANDARD I/O UNITS
PARAMETER ([illegible],NOUT=[illegible])

[illegible]
PARAMETER ([illegible])

DATA [illegible]

[illegible] [illegible]IZATION ROUTINE
CALL INIT (RES,NODES,RTYPE,[illegible],NOW,[illegible],NLIST,NIP,NOUT,[illegible]RATE,
*          [illegible]TIME,TAG)

SET TIME FOR FIRST BREAKPOINT
WRITE(NOUT,*) 'GIVE TIME FOR FIRST INTERRUPT'
READ(NIN,*) BREAK

THIS IS THE BEGINNING OF THE MAIN PROGRAM LOOP

[illegible] CONTINUE

SET FLAG TO CHECK WHETHER SHOULD PRINT
PRINT=[illegible]

CHECK WHETHER ANY REOURCE HAS COMPLETED PROCESSING AN
OUTPUT GRAPH NODE AND SET RESOURCE FREE IF THIS IS SO
DO [illegible] I=1,NLIST
       IF(NCONN(RES(LIST(I),1)).EQ.0.AND.RES(LIST(I),3).LE.NOW) THEN
         RES(LIST(I),1)=0
         RES(LIST([illegible]
```

```
C
C     ADD I/P RESOURCE TO LIST
      NLIST=NLIST+1
      LIST(NLIST)=I

C     INCREMENT TIME TAG
      THIS=THIS+1

      ENDIF


C     EXAMINE EACH RESOURCE

      DO 20 I=1,NLIST

C           IS IT READY TO TRANSMIT?
            IF(RES(LIST(I)).T.LE.NOW) THEN

C             LOCATE A FREE BUS
              IBUS=0
              DO 70 J=1,NBUS
                    IF(BFREE(J).LE.NOW) THEN
                      IBUS=J
                      GO TO 80
                    ENDIF
   70         CONTINUE

   80         CONTINUE
C             (TRANSMISSIONS FROM RESOURCE 0 ARE ARTIFICIAL
C              AND SO DO NOT NEED BUS)
              IF(IBUS.GT.0.OR.LIST(I).EQ.0) THEN
```

```
      SOURCE=GRAPH NODE CURRENTLY BEING PROCESSED
      RESOURCE LIST(I)
      NRES=RES LIST(I,1)

      EXAMINE EACH GRAPH NODE CONNECTED TO SOURCE
      TO WHICH TRANSMISSION IS NECESSARY
      DO 70 J=1,NTRANS(LIST(I))

         SET NON-OPTIMAL RESOURCE
         IRES=1
         SINK=TRANS(LIST(I),J)

         TEST=1 IF TRANSMISSION TO SINK IS POSSIBLE
              =0 OTHERWISE
         TEST=1

         CHECK FOR PRIORITY I/O
         IF(ADJ(SOURCE,SINK).EQ.1) THEN

           IF PRIORITY I/O THEN NEW RESOURCE REQUIRED SO
           CHECK RECEIVE RESOURCE AVAILABLE
           IF(POOL(IRES).EQ.0) TEST=0

         ELSE

           IF NOT PRIORITY I/O CHECK FOR ACTIVE RESOURCE
           EXECUTING CORRECT GRAPH NODE AND TIME TAG
           CALL ACTIVE(NRES,IRES,TEST,LIST(I),SINK,INRATE)

         ENDIF

         TRANSMIT IF POSSIBLE
         IF(TEST.EQ.1) THEN
           CALL XMIT(NRES,IBUS,NOW,IRES,SINK,LIST(I),INF,TTIME,
                     NLIST
           SET PRINT FLAG
           PRNT=1
         ENDIF

   70 CONTINUE

      ELIMINATE TRANSFERS THAT HAVE OCCURRED FROM LIST TRANS
      INDX=0
      DO 75 I=1,NTRAN,LIST(I)
         IF(TRANS(LIST(I),J).NE.INF) THEN
           INDX=INDX+1
           TRANS(LIST I),INDX)=TRANS(LIST(I),J)
         ENDIF
      CONTINUE
      NTRANS(LIST(I))=INDX

      TRANSMIT TO GLOBAL MEMORY IF RESOURCE IS NOT ALREADY
      A MEMORY AND THERE ARE STILL TRANSFERS TO MAKE

      IF(INDEX.GT.0.AND.RES(LIST I),4).EQ.0) THEN
        CALL XMITM(NRES,IBUS,NOW,IRES,SOURCE,LIST(I),
        MTYPE,NOUT,TTIME,NLIST,TEST)
        SET PRINT FLAG
        PRNT=1
      ENDIF

      SET RESOURCE AS FREE AND INCREMENT RESOURCE POOL
      IF RESOURCE WAS NOT A MEMORY OR IF THERE ARE NO
```

```
[illegible]
              [illegible]
              [illegible]INF
              [illegible]INF
              [illegible]
              [illegible]+1
              [illegible]
              [illegible]
              MARK [illegible] LIST [illegible]
              [illegible]
           [illegible]

           REMOVE [illegible] RESOURCES FROM LIST
           INDEX=0
           DO 75 J=1,NLIST
                 IF(LIST(J).NE.INF) THEN
                   INDEX=INDEX+1
                   LIST(INDEX)=LIST(J)
                 ENDIF
75         CONTINUE
           NLIST=INDEX

         [illegible]
       [illegible]

   [illegible]

   [illegible]
   [illegible]
   [illegible] BUTIL(I,1)=BUTIL(I,1)+1
   [illegible]

   DISPLAY SYSTEM STATUS
   IF(PRNT.EQ.1) CALL STATUS(NOUT,NOW,NRES,RTYPE,INF,NBUS,NLIST,PFLAG)

   INCREMENT TIME AND REPEAT
   [illegible] CALL [illegible](NOUT,NBUS,[illegible],NOW,PFLAG)
   [illegible]
   GO TO 1[illegible]

   END


   SUBROUTINE SETUP(NODES)
   IMPLICIT INTEGER (A-W)

   PARAMETER STATEMENT FOR ARRAY DIMENSIONS
   PARAMETER (N1=[illegible],N2=15,N3=5,N4=50)

   COMMON [illegible] ADJ(0:N1,0:N1),POOL(N2),EXTIME(N2),TYPE(0:N1),BFREE(N1),
  +RES[illegible]:N4,5),CONN(0:N1,N1),NCONN(0:N1),NIF(N1),TRANS(0:N4,N1+1),
  +[illegible],LIST(N4+1),BUTIL(N3,2)

   DO 10 I=0,NODES
         NCONN(I)=0
         DO 10 J=0,NODES
               IF(ADJ(I,J).NE.0) THEN
                 NCONN(I)=NCONN(I)+1
                 CONN(I,NCONN(I))=J
               ENDIF
1[illegible] CONTINUE
```

```
C
C     ELEMENT RESOURCE WAS JUST FREED. IF ANY MEMORY UNITS WERE FREED
      IF BFREE(IBUS,5).GT.0 THEN
        POOL(MTYPE)=POOL(MTYPE)+
C       SET SEARCH INDEX FOR NEXT RESOURCE
        DO 2 I=1,NCONN(ISIN)
          TRANS(IBUS,I)=CONN(ISIN,I)
        CONTINUE
        NTRANS(IBUS)=NCONN(ISIN)
      END IF

C     SPECIAL CONDITION FOR STARTING OFF F B NODES
      IF (ITYP(ISIN).EQ.1.AND.BFREE(IBUS,5).EQ.0).OR.(ISIN.GT.1.AND.BFREE(I-2 ,ED.0)
     -   BFREE(IBUS,5)=BFREE(IBUS,5)+1

      IF (BFREE(IBUS,5).EQ.(IPRIN(ISIN) THEN
C       GET RANDOM TIME OF PRODUCE
        IF(BFREE.NE.0) THEN
          CALL EXPON
          BFREE(IBUS,4)=NOW+EXTIME*PRIN(ISIN)+1.E-1
        ELSE
          CALL TRANSFER(MODE)
          WRITE(NOUT,*)'ERROR IN MODE'
        END IF
      END IF

C     ELIMINATE THIS BRANCH CODE FROM SOURCE RESOURCE TRANSMISSION LIST
      DO 40 I=1,NTRANS(BUS)
        IF(TRANS(BUS,I).EQ.SINK) TRANS(BUS,I)=INF
40    CONTINUE

      RETURN
      END


      SUBROUTINE MIN(NRES,ICON,FREE,SOURCE,BUS,ITYPE,NOUT,TRANST,NLIST,
     -               IER)
      IMPLICIT INTEGER (A-Z)

C     PARAMETER STATEMENT FOR ARRAY DIMENSIONS
      PARAMETER (N1=2 ,NL=1 ,NTYP=4,N4=50)

      COMMON /A/ MST(0:N1),QUE(N1),POOL(N2),ETIME(N2),TYPE(0:N1),BFREE(N1),
     -BFES(0:N4,5),CONN(0:N1,N1),NCONN(0:N1),NIP(N1),TRANS(0:N4,N1+1),
     -NTRANS(0:N4),LIST(N4+1),BUTIL(N1),Z

C     ARE THERE ANY MEMORY UNITS LEFT?
      IF POOL(MTYPE).EQ.0 THEN
        WRITE(NOUT,*) ' OUT OF MEMORY - SYSTEM CRASH
        STOP
      END IF

C     IF SOURCE IS I F TYPE THEN BUS IS NOT USED
C     OTHERWISE SET BUS FREE TIME
      IF(BFREE.NE.0) BFREE(IBUS = NOW-TRANST

C     LOCATE AN UNUSED ELEMENT IN RESOURCE AREA
      DO 1 I=1,NRES
```

```
[illegible]

      SUBROUTINE SYST(NODES,RTYPE,NBUS,NIN,NOUT,INRATE,TTIME)
      IMPLICIT INTEGER(A-W)

C     PARAMETER STATEMENT FOR ARRAY DIMENSIONS
      PARAMETER [illegible]

      [illegible]
     *[illegible],TRANS[illegible]
     *[illegible]
      [illegible]

C     [illegible]
      WRITE[illegible] NAME OF FILE TO READ INPUT DATA
      READ[illegible]

      OPEN[illegible]
      REWIND 1

C     GET THE [illegible] DATA RATE
      WRITE[illegible] RATE
      READ([illegible]) INRATE

      [illegible] NODES,RTYPE,NBUS,TTIME

      DO 1 I=1,NODES
            READ(1,[illegible]) (ADJ(I,J),J=1,NODES)
    1 CONTINUE

      READ(1,[illegible]) (POOL(I),I=1,RTYPE)
      READ(1,900) (EXTIME(I),I=1,RTYPE)
```

```
      SUBROUTINE STATUS(NO,NUMB,NFREE,NT,PF,IHR,IMIN,ISEC,IFLAG)
      IMPLICIT INTEGER (A-W)
      SAVE TEST

      PARAMETER STATEMENT FOR AREA DIMENSIONS
      PARAMETER (N1=15,N2=5,N3=5,N4=50)

      COMMON /A/ ALL(N1,N2,N3), POOL(N2),STIME(N1),TYPE(N1),FREE(N2),
     +FLG(N1,N3),IQUE(N1,N2),NQ(N2),NP(N2),IP(N1),IA(N1,N4),
     + IS(N1),LIST(N1),BUFL(N2)

      DATA TEST/'TEST'/
      IFLAG=3
      RETURN

      WRITE(NOUT,*)
      WRITE(NOUT,*)  TIME IS ',IHR,IMIN,ISEC
      IF(NT.GT.0)GO
      IF(NFREE.EQ.0)WRITE(NOUT,*)  BUF   ,I,  IS BUSY
      IF(NFREE .GE.NT)WRITE(NOUT,*)  BUF   ',I,  IS FREE
      CONTINUE

      WRITE(NOUT,*)
      WRITE(NOUT,*)  RESOURCES AVAIL
      WRITE(NOUT,11) (POOL(I),I=0,RTYPE)

      WRITE(NOUT,*)
      WRITE(NOUT,*)  BUF  AVAILABILITY
       WRITE(NOUT,12) (I,(FLG(I,J),J=1,NFLG),I=1,NBUF)

      WRITE(NOUT,*)
      WRITE(NOUT,*)  FREE MEMORY  AND  MEMORY SIZE REQUEST
      DO 10  I=1,NUMB
            WRITE(NOUT,*)  LIST(I)  NEXT(I)  ,I  IA(I,1),
                           IA(I,2)  ,IA(I,3)  ,IA(I,4)
   10 CONTINUE

      RETURN

   10 FORMAT(1X,10I4,/,10I4,/,I6,2X,I1,1X,3I1)
   11 FORMAT(1X,20I3)
   12 FORMAT(1X,3I4,1H%)

      END


      SUBROUTINE MEMORY
      IMPLICIT INTEGER(A-W)

C     PARAMETER STATEMENT FOR AREA DIMENSIONS
      PARAMETER (N1=15,N2=5,N3=5,N4=50)

      COMMON /A/ ALL(N1,N2,N3),POOL(N2),STIME(N1),TYPE(N1),FREE(N2),
     +FLG(N1,N3),IQUE(N1,N2),NQ(N2),NP(N2),IP(N1),IA(N1,N4),IS(N1),
```

```
      SUBROUTINE INITIAL(MODE,RTYPE,MTYPE,NOW,INF,NEVENT,IN,NOUT,
     1                   IRATE,TTIME,TAS)
      IMPLICIT DOUBLE PRECISION
C
C     ADJUSTABLE DIMENSION FOR ARRAYS DIMENSION
      PARAMETER (MM1=21,MM=15,MM2=5,MM4=50)
C
      DIMENSION A(MM1,MM2),ATRIB(MM),ETIME(MM),TYPE(MM),WEIGHT(MM),
     1          D(MM1,MM1),ND(MM1),NI(MM1),TNOW(MM),II(MM),
     2          NLIST(MM4),LIST(MM4),PQ(MM1,2)
C
      IF(MODE.EQ.1) GO TO 10
      CALL INITIAL(MODE,RTYPE,NEVENT,IN,NOUT,IRATE,TTIME)
C
C     ENTRY POINT FOR MODE=1
   10 CONTINUE
C
      RETURN
C
C     ENTRY FOR ASSIGNMENT OF RESOURCES
      ENTRY ASSIGN(MODE,RTYPE,NEVENT,IN,NOUT,IRATE,TTIME)
      MTYPE=RTYPE
C
      NO RESOURCES IN LIST YET
      NLIST=0
C
      THIS IS SPECIAL TYPE
      IF(RTYPE.EQ.1) GO TO 20
      CALL ERROR(RTYPE)
C
      COMMON
      DIMENSION
      DO 5 I=1,NEVENT
      SERVER I=NOW
      D(I,1)=0
      D(I,2)=0
    5 CONTINUE
C
      FIND TOTAL NUMBER OF RESOURCES
C
      NRES=0
      DO 6 I=1,RTYPE
          NRES=NRES+POOL(I)
    6 CONTINUE
C
      INITIALISE ALL RESOURCES AS DOING NOTHING
      DO 15 I=0,NRES
            RES(I,1)=0
            RES(I,2)=INF
            RES(I,3)=INF
            RES(I,4)=0
            RES(I,5)=0
   15 CONTINUE
```

```
[illegible]
[illegible]
[illegible]

[illegible] THEN
  WRITE([illegible]) 'TOO MANY RESOURCE TYPES- INCREASE PARAMETER [illegible]
  [illegible]
[illegible]

[illegible] THEN
  WRITE(NOUT,*) 'TOO MANY BUSSES- INCREASE PARAMETER N[illegible]
  [illegible]
[illegible]

IF(NPROC.GT.[illegible]) THEN
  WRITE([illegible],*) 'TOO MANY PROCESSORS- INCREASE PARAMETER N[illegible]
  [illegible]
[illegible]

[illegible]
[illegible]


SUBROUTINE CHANGE(NIN,[illegible],NBUS,INRATE,[illegible],[illegible],PFLAG)
IMPLICIT INTEGER(A-W)

PARAMETER STATEMENT FOR ARRAY DIMENSIONS
PARAMETER ([illegible]=2[illegible],N2=15,N3=5,N4=5[illegible])

[illegible] [illegible](N1),[illegible](N1),POOL(N2),[illegible](N2),TYPE([illegible]N1),BFREE(N3),
*[illegible] [illegible](N1),[illegible](N1),[illegible](N1),TRANS([illegible]N4,N1+[illegible]),
*[illegible] [illegible],LIST(N4+1),BUTIL(N3,2)

PAUSE

[illegible]

WRITE(NOUT,*)
WRITE(NOUT,*)          'TIME IS ',[illegible]
WRITE(NOUT,*)
WRITE(NOUT,*)          'WHAT DO YOU WISH TO DO?
WRITE(NOUT,*)          'OPTIONS ARE:
WRITE(NOUT,*)             1 CHANGE INPUT RATE
WRITE(NOUT,*)             2 CHANGE NUMBER OF RESOURCES IN SYSTEM
WRITE(NOUT,*)             3 CHANGE EXECUTION TIME [illegible] RESOURCE
WRITE(NOUT,*)             4 CHANGE NUMBER OF BUSSES
WRITE(NOUT,*)             5 STOP PROGRAM
WRITE(NOUT,*)             [illegible] CONTINUE SIMULATION
WRITE(NOUT,*)
WRITE(NOUT,*)          ENTER OPTION NUMBER
READ(NIN,*) OPTION

 IF(OPTION.EQ.1) THEN
   WRITE(NOUT,*) 'GIVE NEW I/P RATE'
   READ(NIN,*) INRATE

 ELSEIF(OPTION.EQ.2) THEN
   WRITE(NOUT,*) 'WHICH RESOURCE ALLOCATION DO YOU WISH TO CHANGE'
   READ(NIN,*) IRES
   WRITE(NOUT,*) 'SPECIFY CHANGE IN RESOURCE POOL FOR RESOURCE ',IRES
   WRITE(NOUT,*) 'POSITIVE NUMBER TO INCREASE POOL, NEGATIVE TO DECREASE
   READ(NIN,*) CRES
   IF(POOL(IRES)+CRES.LT.[illegible]) THEN
```

```
[illegible]
[illegible]

[illegible]
  WRITE [illegible] [illegible] SCHEDULE EXECUTION TIME [illegible] WISH TO CHANGE
  [illegible]
  [illegible] EXECUTION TIME
  READ([illegible]) EXTIME [illegible]

[illegible] IF [illegible] THEN
  WRITE ([illegible],*) HOW MANY BUSSES DO YOU WANT
  READ(NI[illegible],*) NBUS1
  IF ADDING MORE BUSSES INITIALISE BUS PARAMETERS
  IF (NBUS1.GT.NBUS) THEN
    DO 2[illegible] I=NBUS+1,NBUS1
          BUTIL(I,1)=0
          BUTIL(I,2)=NOW
          BFREE(I)=0
    CONTINUE
  ENDIF
  SET NUMBER OF BUSSES TO NEW VALUE
  [illegible]

[illegible] IF ([illegible].EQ.5) THEN
  [illegible]

[illegible] IF ([illegible].EQ.[illegible]) THEN
  WRITE (NOUT,*) 'GIVE TIME FOR NEXT BREAK'
  READ ([illegible],*) BREAK
  [illegible]
  [illegible]

ENDIF

GO TO 10

END
```


NI
DATE