Research Report

NO. 82-5

FINAL REPORT ON

COMMUNICATIONS CONTROLLER

FOR

WORD PROCESSORS

PREPARED FOR THE DEPARTMENT OF COMMUNICATIONS
UNDER DSS CONTRACT NO. 05U81-00115

BY

H. T. MOUFTAH

MARCH 1982

Queen's University at Kingston
Department of Electrical Engineering

P
91
C655
M68
1982

FINAL REPORT ON COMMUNICATIONS CONTROLLER FOR WORD PROCESSORS

Prepared for the Department of Communications

Under Contract No. 05U81-00115

by H. T. Mouftah
Department of Electrical Engineering
Queen's University
Kingston, Ontario.

March 1982

Research Report No. 82-5

Industry Canada
LIBRARY
JUL 20 1998
BIBLIOTHÈQUE
Industrie Canada

COMMUNICATIONS CANADA
SEP 30 1983
LIBRARY – BIBLIOTHÈQUE

DD 3961716
DL 3966431

P
91
C655
M68
1982

COMMUNICATIONS CANADA
LIBRARY - BIBLIOTHÈQUE

ACKNOWLEDGEMENTS

The author would like to thank the Research Assistants E. Krause and S. Bhatia for developing, debugging and running the computer simulation program. The cooperation of Mrs. Marion Rose in typing this report is also greatly appreciated.

Table of Contents

	Page
Acknowledgements	(i)
Table of Contents	(ii)
List of Figures	(iii)
Abstract	1
1. Introduction	2
Objectives	3
2. The Word Processing Communications Controller	3
2.1 The Hardware Structure	4
2.2 The Software Model	5
2.3 Assumptions	6
3. The WPCC Computer Simulation Program	8
3.1 Computer Program Description	8
3.1.1 Subroutine NPINR	10
3.1.2 Subroutine PROCES	12
3.1.3 Subroutine HISPT	14
3.1.4 Subroutine NPINT	14
3.1.5 Subroutine NEXT	15
3.2 Input Parameters	16
3.2.1 General input parameters	16
3.2.2 Device parameters	17
3.2.3 Line parameters	17
3.2.4 Control parameters (switching matrix)	18
3.3 Output Parameters	19
3.4 Intermediate Global Variables	21
3.5 Results and Discussions	24
4. Summary and conclusions	27
5. References	29
Appendix A: Computer Simulation Program Listing.	30
Appendix B: A Typical Output File Listing.	67
Appendix C: A Typical VAX 11/750 LOG FILE Listing.	71

List of Figures:

Figure 1 System Configuration of the Word Processing Communications Controller

Figure 2 WPCC Software Simulation Model

Figure 3 Queueing Model of the WPCC

Figure 4 The Structure of the WPCC Computer Simulation Package

Figure 5 System Flow Chart of the WPCC Computer Simulation Program

Figure 6 Typical Input Parameters for the WPCC Computer Simulation Program.

Figure 7 System Throughput versus Line Utilization for Dissimilar Protocols and Different Buffer Size.

Figure 8 Average Message Delay versus Line Utilization for Dissimilar Protocols and Different Buffer Size.

Figure 9 System Throughput versus Mean Message Lenth for Different Buffer Size.

Figure 10 Probability of Data Loss versus Buffer Size for Different Mean Message Length.

Figure 11 System Throughput versus Data Rate of the Word Processors Terminals for Different Buffer Size.

Abstract

A computer simulation package has been developed as a tool to determine the hardware/software configurations of a microprocessor-based communications controller for word processors. The Word Processing Communications Controller (WPCC) is a microprocessor-based device that is designed essentially to connect a number of stand-alone word processing machines that could be of dissimilar types to each other, to a smaller number of local common resource equipment and, via a number of shared lines, to data bases through private or public data networks.

The computer program evaluates the performance of the WPCC under different traffic characteristics and different hardware/software configurations. The performance evaluators are the system throughput and the average message delay with its variance. The results of the performance evaluation study is useful for both the Designer and the Systems manager.

1. Introduction:

This report presents the results of a study performed by Queen's University under a DOC University Research Contract entitled "Communications Controller for Word Processors".

Recent years have witnessed widespread use of word processing machines in offices in both government and industry. A wide variety of word processing products has become available to service this expanding market. As business office automation continues and with the increasing trend towards organizational decentralization, it is expected that the need for communications between such devices will grow.

However, no industry-wide standards exist at this time for defining communications protocols appropriate to the word processing environment. As a result, each manufacturer has developed its communications capabilities on an ad hoc basis making the connection to word processors of other manufacturers very difficult. This imposes considerable limitations on the amount of information that can be transferred directly between dissimilar word processors. Then, to allow two dissimilar word processors to talk to each other a translation package that contains look up tables for code translation is required.

If a word processor is to communicate with many different types of word processing equipment, translation tables will be required for each possible connection, making the system complex and costly. A translation box would relieve each word processor of the translation task by assuming responsibility for maintaining all required information about various machines.

It is desirable that this translation box be able also to permit a

number of stand-alone word processor terminals to share communication lines besides the capability of communicating between themselves. This box becomes then a communications controller rather than just a code translator.

The function of the communications controller may be also extended to providing shared access to local common resources such as high speed printers, optical character readers, and mass storage devices. This will avoid the cost of duplicating these expensive devices that are not utilized frequently by a single word processor.

Objectives:

The main objective of this study is then to provide a design tool for defining the system requirements and technical feasibility of an intelligent communications controller for word processors.

The word processing communications controller should:

1. permit a number of word processor terminals to share communication lines.
2. provide communication between the terminals themselves.
3. provide protocol translation to allow communication between different models of word processor.
4. provide shared access to common resource equipments.
5. comply with the principles of the Open System Interconnection Model.

2. The Word Processing Communications Controller:

The realization of a Word Processing Communications Controller (WPCC) that can perform all the functions mentioned above may be feasible only if it contains an intelligent element. The WPCC is then a microprocessor-based device that is designed essentially to connect a

number of stand-alone word processing machines (N) that could be of dissimilar types, to each other, to a smaller number of local common resource equipments (M) and via a number of shared lines (L) to data bases through private or public data network. The general configuration of the system is shown in Figure 1.

Two versions of WPCC have been investigated. The WPCC Version I is the basic, simple type of WPCC that is designed for small to medium word processing environment where numbers N, L and M are not greater than 8, 1 and 4 respectively. In this case a general purpose microprocessor constitutes the heart of this WPCC. The WPCC Version II is provided for medium to large word processing environment where numbers N, L and M are in the order of 20, 4 and 8 respectively. In this case an advanced 16-bit microprocessor is used.

The design tool presented in this report has been development on the form of a computer simulation package. A hardware/software model of the WPCC is first described then the simulation of this model on a computer is presented. Only the WPCC Version I has been considered. Although the hardware structure is different, it is assumed that the software model is the same in both versions. Thus with merely some modifications in the input parameters, the computer simulation package described in this report can be used for both cases.

2.1 The Hardware Structure:

The WPCC is assumed to be built around a single standard microprocessor (the Motorola MC6809) presently available at low cost. A small number (up to 8) of low speed lines interface the individual word processors to the device, one high speed trunk (1200-9600 bps) links the device to a host computer or a node in a computer communications

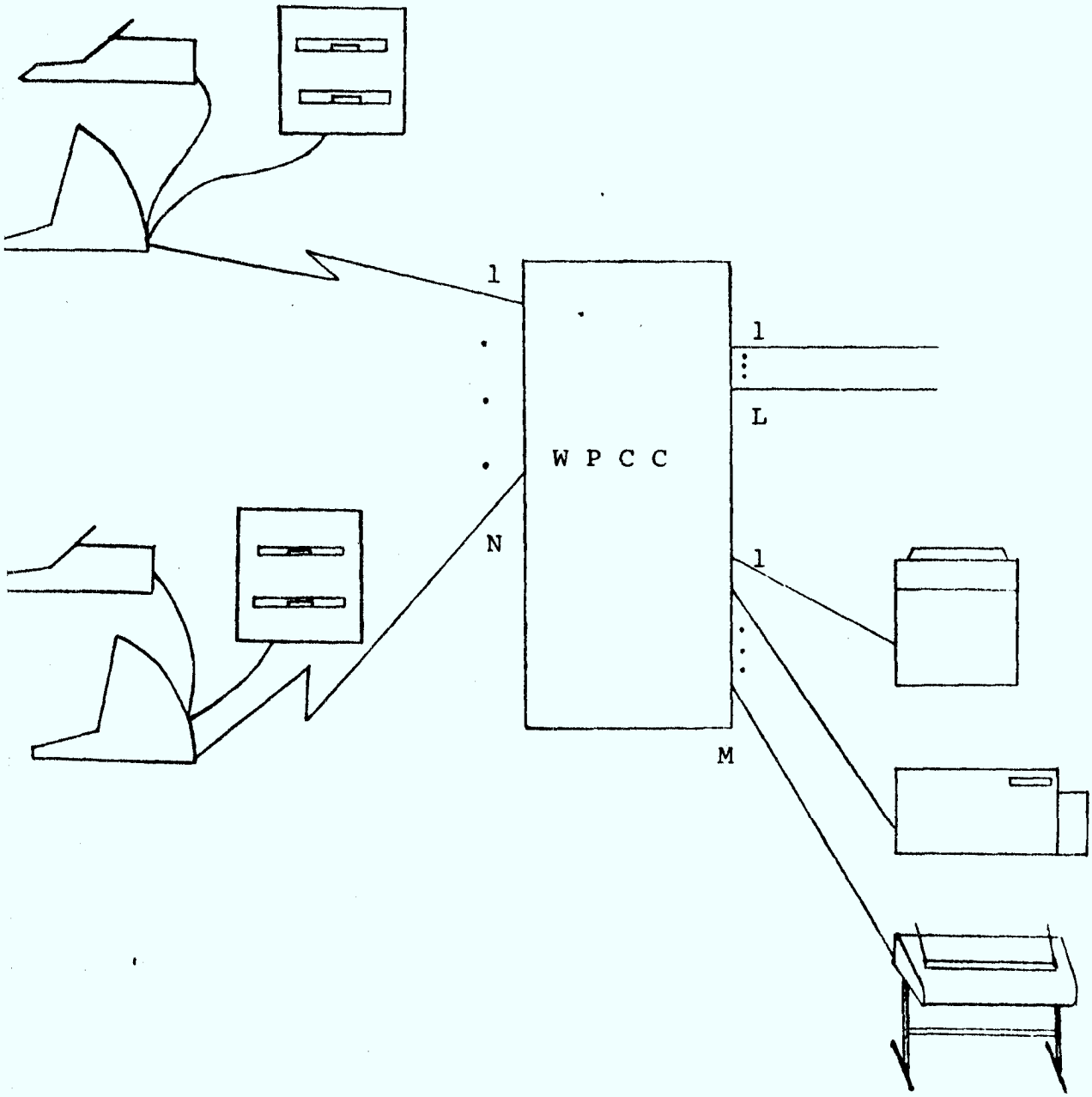

Figure 1. System Configuration of the Word Processing Communications Controller:

network, and up to 4 lines connect the common resource equipments to the device. Besides the microprocessor and its complementary peripheral chips, the WPCC contains 32 kilobytes (i.e. 32KB) of program memory (ROM, PROM or RAM), 4-16KB of buffer RAM, and a number of serial communications interface LSI chips. This basic hardware configuration is very flexible and the configuration can be altered to a certain extent by varying the input parameters to the simulation program.

2.2 The Software Model:

The WPCC software is modelled as three different separate layers: the non-preemptive interrupt handler (NPIH) layer, the preemptive interrupt handler (PIH) layer, and the background (BG) layer. The NPIH layer handles interrupts on a character basis while the PIH layer processes interrupt services on an internal message block basis. Each layer has a receiving and transmitting side.

It has to be noted here that the word "message" is used throughout this report to define a group of characters received or transmitted on a line, and is contained in one packet. On the other hand the term "internal message block" is used to define a group of characters transferred at once from one layer to another internally in the WPCC. A message may be composed of one or more internal message blocks.

The internal message block is composed of 32 words of 16-bits each. The header of each message block reserves 13 (or 1) words leaving only 19 (or 31) words for data information (depending on whether the internal message block is the first one of an incoming message or not). Figure 2 shows the WPCC software model while the queueing model is shown in Figure 3.

Characters are received and transmitted in the NPIH layers.

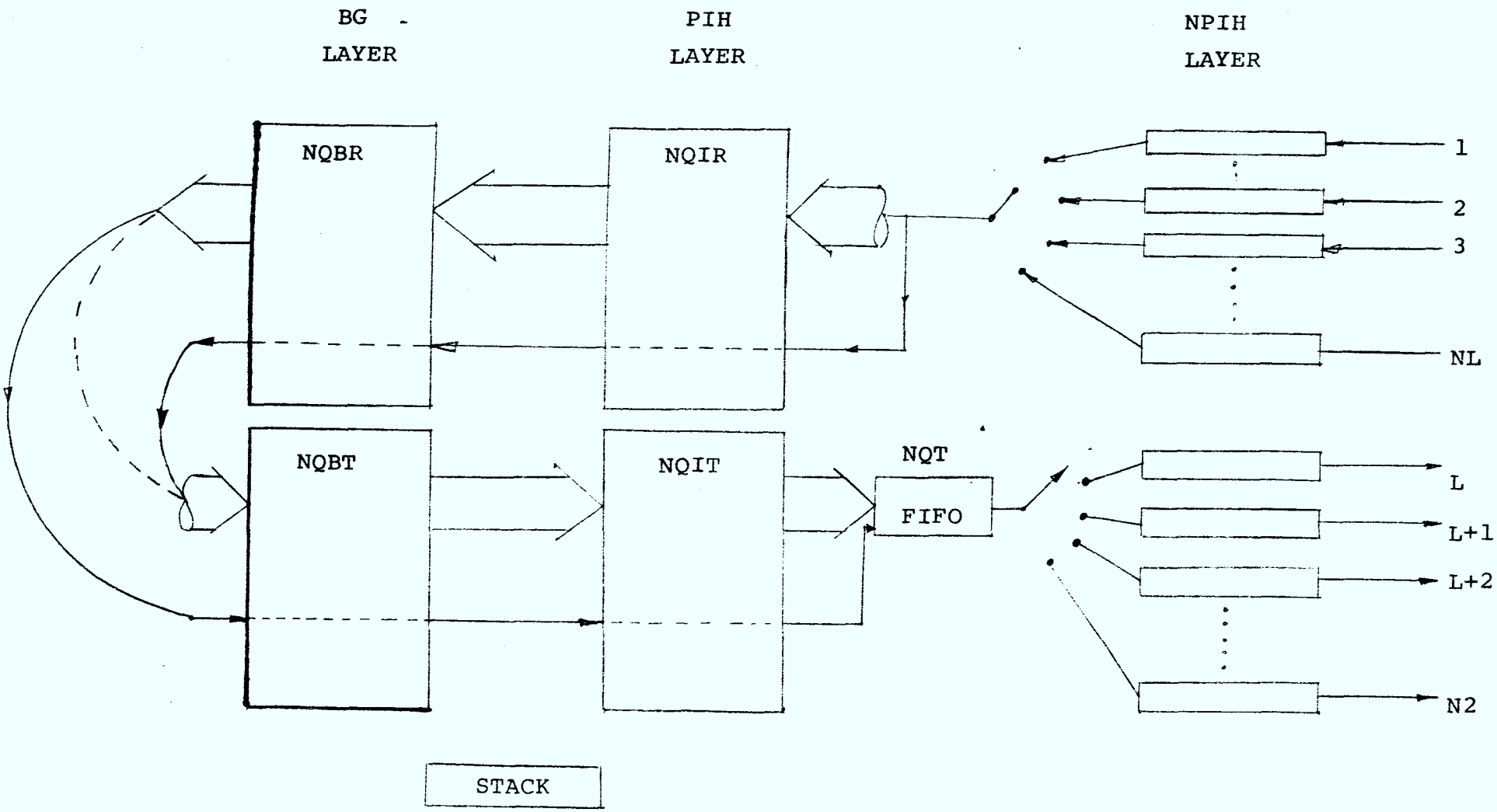

Figure 2. WPCC Software Simulation Model

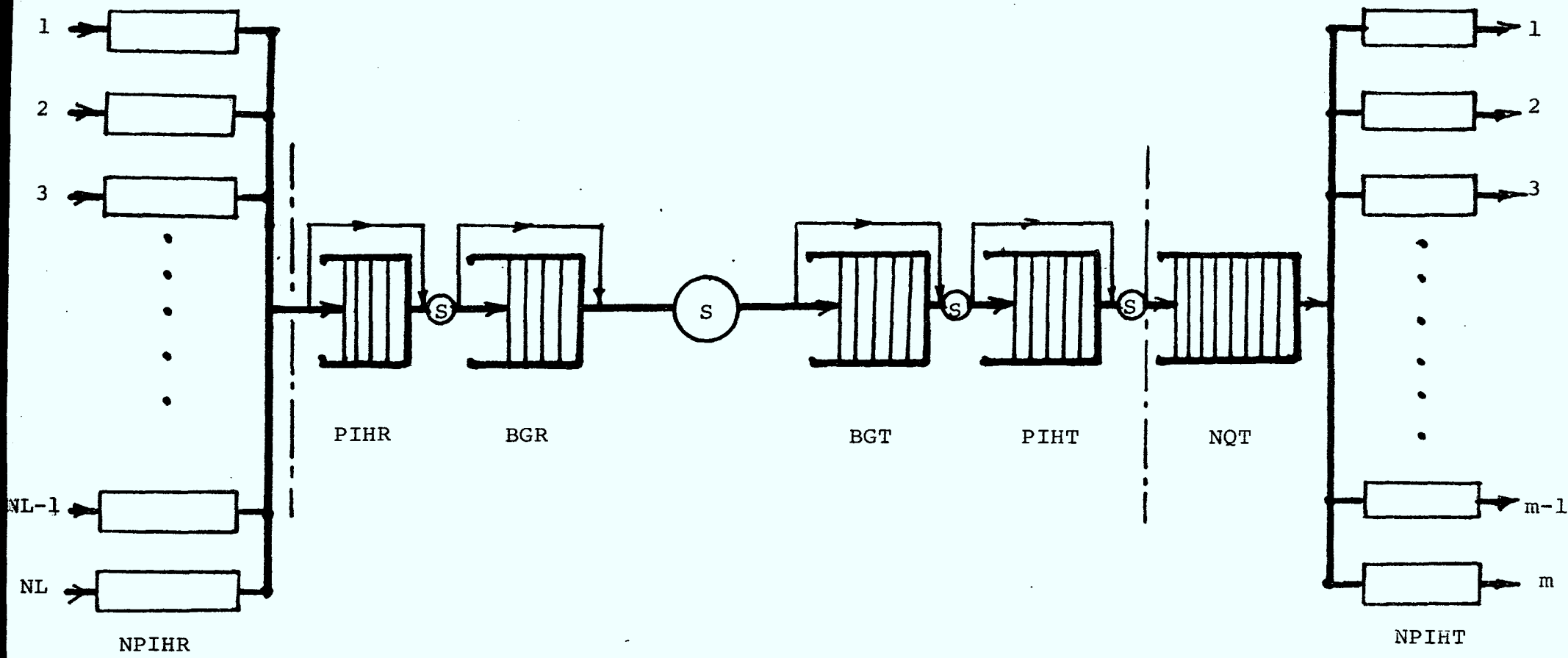

Figure 3. Queueing Model of the WPCC

Received characters are buffered and grouped in the receiving NPIH layer and sent to the PIH layer in the form of message blocks. These message blocks are stored in the PIH layer receiving queue (the NQIR) waiting for processing. After a message block has been processed in the PIH layer it is sent to the BG layer and stored in its receiving queue (the NQBR) waiting for processing.

Line switching occurs in the background layers. The processing time in this layer is determined by any differences in protocols assigned to the two lines as well as by the length of the message. As the difference between protocols is bigger the translation routines will be larger and hence their execution time will be longer.

After a message block is processed in the transmitting side of the BG layer it is stored in the transmitting side of the PIH queue (the NQIT) to await reprocessing in this layer. The message is then placed in the transmitting queue (NQT) to await transmission. Transmitting will begin immediately if the required line is full duplex and is not already transmitting, or if the line is half duplex and is neither receiving nor transmitting. The individual characters can then be processed in the transmitting side of the NPIH layer.

If processing in the PIH or BG layers is interrupted by the next earliest character arrival, the message block being processed will be stored temporarily in a STACK register. Processing will resume at the same point as soon as the character has been processed.

2.3 Assumptions:

The following are the general assumptions considered throughout this study.

1. The user to computer traffic varies with the type of device connected

to the low speed lines, i.e. CRT terminals, word processors, teletypes, disk drives etc. Two types of traffic are considered:

i. Arrival of characters and messages are Poisson processes. Time between arrivals is a continuous random variable with exponential distribution. Message length on either side (trunk or low speed lines) is also a continuous random variable with exponential distribution.

ii. Input traffic is bursty and arrival of messages is a compound Poisson distribution. The message lengths and the burst arrivals are approximated as exponentially distributed.

2. A multipriority system:

i. Software layer priority: three levels; highest priority for NPIH layer, and the lowest for BG layer.

ii. Queues priority: there is one stack register in the system. The stack has the highest priority, then the PIH waiting queue. The BG waiting queue has the lowest priority.

iii. Line service priority: each line has its own priority level. The trunk has the highest priority.

3. All lines are error free and therefore require no retransmission. Retransmission is assumed only due to WPCC failure (overrun, underrun or overflow of queues).

4. Each message sent by an initiating line will require a response or acknowledgement.

5. Files are received synchronously on any line even though this line can receive messages asynchronously (in the active mode). The number of sync characters preceding each data character arrival is generated using an exponential distribution having a mean that is inversely

proportional to the synchronous utilisation of the line.

3. The WPCC Computer Simulation Program:

3.1 Computer Program Description:

The WPCC computer simulation program is composed of a main program and ten subroutines structured as shown in Figure 4. FORTRAN is used in coding the program which is composed of approximately 1.5K FORTRAN lines.

The main program reads input parameters, initializes variables and time clock, and calls subroutines. The following is a brief description of the different subroutines grouped into three groups according to their level in the hierarchy.

a. Level I:

1. Subroutine NPINR: simulates the NPIH layer on the receiving side of each line.

2. Subroutine NPINT: simulates the NPIH layer on the transmitting side of each line.

3. Subroutine GASTAT: gathers statistics and prints outputs.

4. Subroutine ERLST: determines the earliest event to be serviced in the NPIH layer.

b. Level II:

1. Subroutine PROCES: simulates the PIH layer and BG layers.

2. Subroutine NEXT: determines the next earliest event to be serviced in the NPIH layer.

c. Level III:

1. Subroutine HISPT: selects a message from the NQT Queue for transmission.

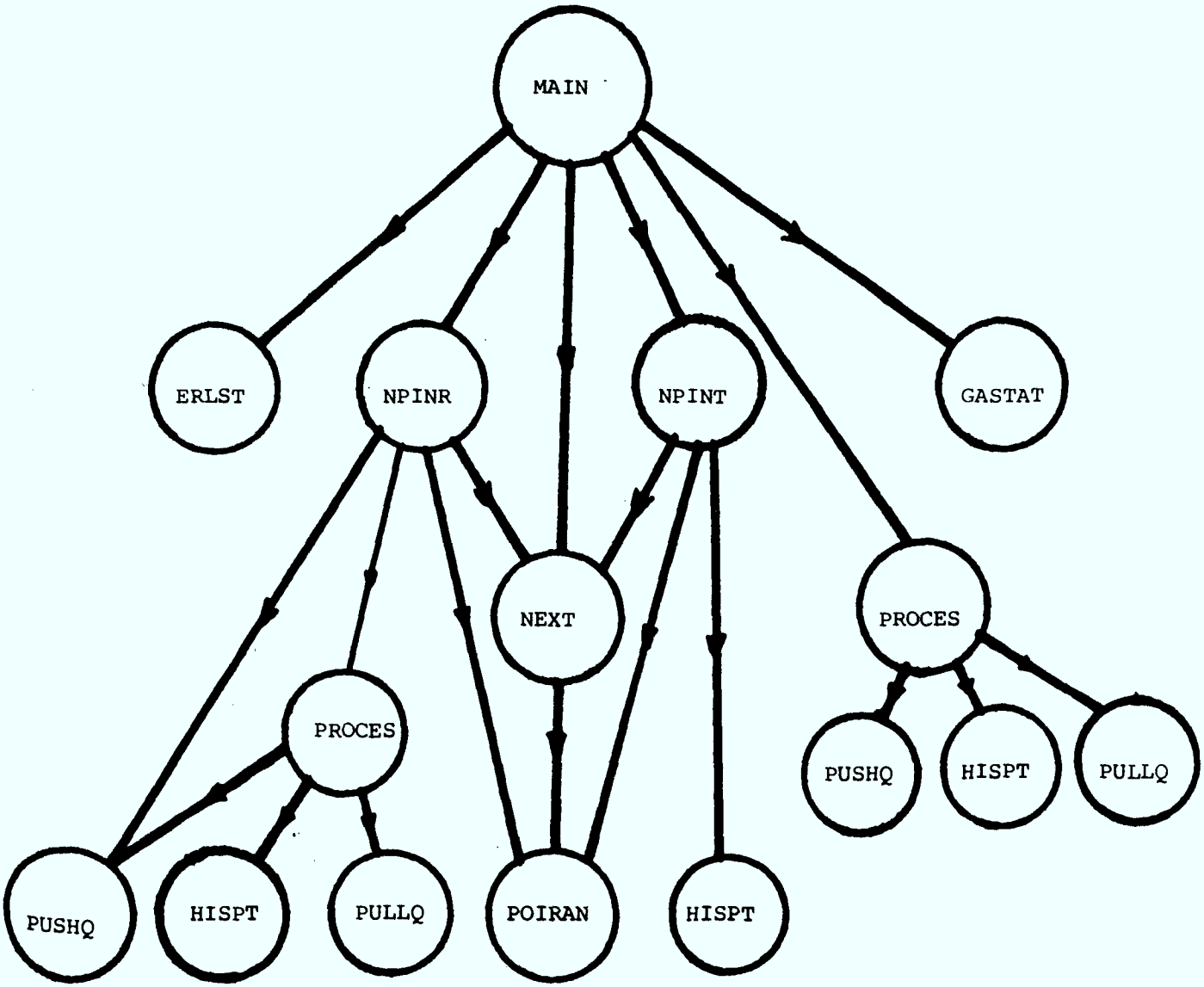

Figure 4. The Structure of the WPCC Computer Simulation Package.

2. Subroutine PUSHQ: pushes an internal message block in the prioritized queues NQIR, NQBR, NQBT and NQIT.

3. Subroutine PULLQ: pulls an internal message block from the prioritized queues NQIR, NQBR, NQBT and NQIT.

4. Subroutine POIRAN: generates a random number variable that follows an exponential distribution.

The input data is divided into four classes; the general system parameters, the device parameters, the line parameters and the control parameters. The system parameters supply general information such as the number of lines and the simulation time. The device parameters give information specific to each type of device such as device name, mode of transmission: half or full duplex. Information related to a specific line such as traffic characteristics, processing time in each layer, and line service priority, is given by the line parameters. The type of interactions which may occur, its relative probability, the number of loops (messages per interaction), waiting times and the message length and file size sent in each direction are variables specified by the control parameters.

The initialization of variables precedes each simulation. The arrival time of the first character in the message must also be generated for each line that is capable of initiating an interaction.

Subroutine ERLST is then called to detect the arrival time of the first character due for processing in the Non-Preemptive Interrupt Handler Layer. If a character was not picked up during the time that it was available due to the processing of the last character, an overrun or underrun will be registered. Otherwise subroutine NPINR is called if the character is on a receiving line or NPINT is called if the character

is on a transmitting line. In the NPIH layer characters are processed and are buffered so that they may be grouped into blocks and processed in the PIH and BG layers after the entire message has been received.

After returning from NPINR or NPINT, subroutine PROCES is called to check if any message blocks are waiting to be processed in the NQIR, NQBR, NQBT, NQIT or NQT queues. After returning from this subroutine, control is returned to the beginning of the loop to await the next character arrival if the simulation time or number of messages processed does not exceed the specified values. Otherwise the first simulation is completed by calling subroutine GASTAT to gather statistics and to prepare the output. Figure 5 shows the system flow chart of the WPCC Computer Simulation Program.

The following is the functional description of some subroutines that constitute the important part of the package (subroutines NPINR, PROCES, HISPT, NPINT and NEXT).

3.1.1 Subroutine NPINR:

The simulation clock is advanced by the time required to process a character in this layer and cannot be interrupted even if this means that the next earliest character will be missed (overrun or underrun). If this is the first character received from a line that is attempting to initiate an interraction, then the type of device to which the message will be sent must be chosen before continuing. This is done using the set of probabilities for interractions between different devices, supplied in the data switching table. The number of loops, message length, file length, and the maximum waiting time (when the chosen line(s) is busy) are also generated at this time.

The number of characters yet to be received on this line (message

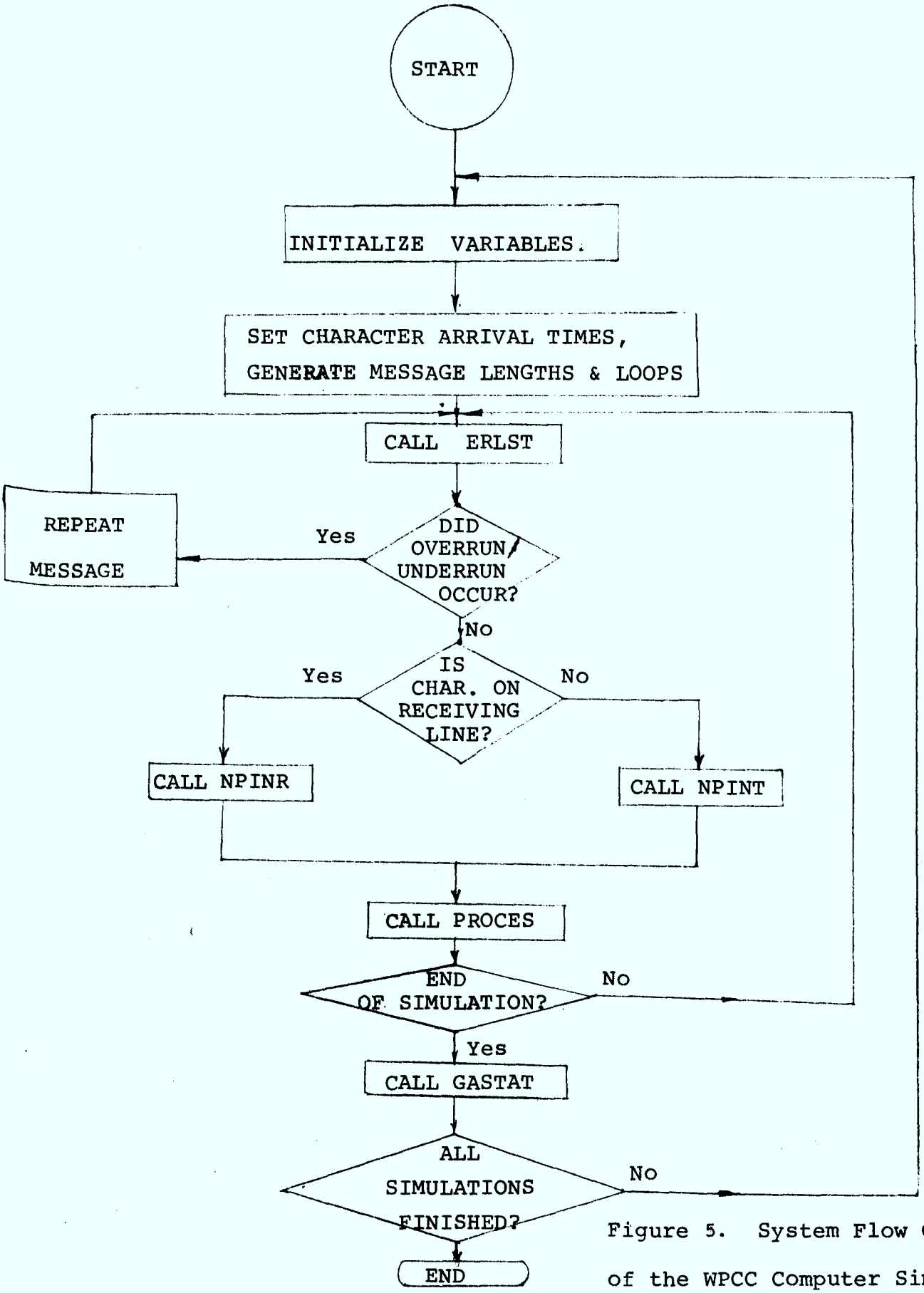

Figure 5. System Flow Chart of the WPCC Computer Simulation Program

length) is decremented and if more characters are expected, the arrival time of the next character is generated before returning to the main program.

After receiving an entire message the status of the interraction is again checked. When initiating, a destination line number will need to be selected from those lines that are of the device type already chosen. If the selected line is already involved in an interraction with some other line, then a different line may be selected only if NCARE (i) (i is the initiating device number) is zero, provided that another line exists. If no lines are available the initiator has the choice of waiting or beginning a new interraction. This decision is made by comparing the time that has been spent waiting and was generated after the first character arrived, to the maximum permitted waiting time.

Before a message can be sent it must be subdivided into blocks. The first block in a message has room for 38 message characters while following blocks, if required, have a capacity of 62. Each block is placed in the first processing queue (NQIR) by calling subroutine PUSHQ (after setting Q=0) for each message block. If however, there had not been enough room in this queue to hold each of the message blocks, the message would be repeated. The next time it arrives, the size of NQIR may have been reduced.

A new message and the arrival time of the first character will be generated for this line if:

1. A file is being sent and more lines are to follow
2. The line is full duplex and:

a. messages that are waiting to receive on this line are queued

or

b. no messages are queued but this is the line that initiated the loop and more messages are to be sent before it is terminated.

Q = 1	NQIR	Q = 3	NQBT
Q = 2	NQBR	Q = 4	NQIT

3.1.2 Subroutine PROCES:

Any message blocks that may be waiting in NQIR, NQBR, NQBT, NQIT, NQT or the stack are processed by this subroutine. When PROCES is called, first priority is given to the stack. Otherwise the integer variable Q points to the queue whose messages are being processed. If the stack was empty, Q will be initially set to 1 and any message blocks waiting in NQIR will be processed. Only when this queue is empty or the next queue is full, Q will be incremented so that processing may begin on any message blocks waiting in the next queue. If a message block had been waiting in the stack, Q would be reset to its previous value and processing would resume at the point where it was interrupted. Subroutines PUSHQ and PULLQ perform their respective operations on the NQIR, NQBR, NQBT and NQIT queues. These queues are prioritised and thus the priority assigned to each line must be considered when the push operation is performed. When a queue is pulled, however, the message block waiting in the first queue position is simply fetched.

When a message block is transferred from NQBR to NQBT, the multiplexing or demultiplexing operation of the background layer is simulated. This simply involves switching the line number stored by the variable KK to the number stored by the variable LSLN. LSLN then assumes

the previous value of KK.

The required processing time in the BG layers is dependent on the protocols of the sending and transmitting lines and must be specified in the input data. Also required is the processing time when continuing from the standard protocol to the protocol of the transmitting line. The sum of these two times is equivalent to the total processing time in the receiving and transmitting BG layers. The processing time in the BG layer from the trunk to the standard protocol (or standard to trunk) will be assumed negligible if the trunk is assigned a protocol type of zero. (PROT(j) = 0 where line j is the trunk). The processing time specified in the input data should be the time required to process a message block consisting of a single character since this time will be multiplied by the number of characters in the block to yield the total message block processing time.

Each time the NQT queue is pushed, subroutine HISPT is called so that this message or some other message in the queue may begin transmitting.

Since the PIH and BG layer operations simulated by this subroutine are preemptive, their operations may be interrupted if a character arrives during processing. Therefore, before a message block is processed the time before the next character arrival (PTMI) is compared to the calculated processing time (TLEFT) and if TLEFT is greater than PTMI the block will be placed in the stack and control will return to the main program. Otherwise processing will continue until all queues are empty or when no more blocks can be processed. The latter situation will occur when each message in NQT is waiting for the required line to be free.

3.1.3 Subroutine HISPT:

HISPT is called each time the NQT queue is pushed or whenever a line has finished transmitting. To simplify its operation, each line is immediately checked to determine which are free. A half duplex line is free if it is neither receiving nor transmitting while a full duplex line is free if it is not already transmitting and the queue of messages waiting to receive on the line is not full.

The variable KK stores the line number of the block that was just pushed into the NQT queue or the number of the line that just finished transmitting. If this line is free the NQT queue will be searched for the first message which matches this line number. Otherwise, if the line was not free or if a message was not found, the queue is scanned for the first message that is waiting for one of the available lines. (A message cannot be selected if one or more blocks have not yet arrived in NQT). The selected message blocks are fetched from the queue and the arrival time of the first character to be transmitted is set. If the message is transmitting on the trunk, the message length is adjusted to include a header of 8 characters for each block of the message.

3.1.4 Subroutine NPINT:

The clock is advanced by the time required to process a character in this layer and as in NPINR, processing cannot be interrupted by the arrival of the next earliest character. The number of characters yet to be transmitted on this line is decremented and if more characters are expected, the transmitting time of the next character is generated before returning to the main program.

When the entire message has been transmitted, subroutine HISPT will be called so that a new message waiting in NQT may begin transmitting on

this line. If a file is being transmitted, no action can be taken until the last line is received. Otherwise a response will be generated if:

1. this is not the line that initiated the loop

   or

2. this is the line that initiated the loop, and more messages are to be sent before the loop is terminated.

When a loop is terminated, the arrival time of the first character in the message of the next loop is generated using LAMDA3 for this line, (the mean time between interactions.) This is also done on the line that responded during the last loop unless LAMDA3 for this line is zero.

3.1.5 Subroutine NEXT:

[NEXT(JJ,EVMN,MSMN,IFMN)]

The subroutine may be called to generate a new event on line JJ. The parameters which may be set are the arrival time of the next character (EVBG(JJ),EATW(JJ)), the message length (MSLNTH(JJ)) and the file size (IFT(JJ)).

JJ : line number on which the event will occur

EVMN : <0 next arrival time generated synchronously. -EVMN should be the utilisation. It is used to generate the number of sync characters before the next arrival.
=0 next arrival time unchanged.
>0 time before next arrival generated using EVMN as a mean

MSMN : <0 message length unchanged
=0 message length generated using the input data as specified by ITYPE(JJ)
>0 message length generated using MSMN as a mean

IFMN : <0 number of lines (for files)unchanged
>0 number of lines in file generated using IFMN as a mean

If EVMN is negative the absolute value should be the utilisation of the line. The reciprocal of the utilisation is then used as a mean to

an exponential distribution which generates the number of sync characters before the next data character is received.

3.2 Input Parameters:

3.2.1 General input parameters:

NDEV - (integer*2, I5) number of different types of devices (1-15)

NL - (integer*2, I5) number of lines (1-25)

SIMT - (real*4, F10.3) maximum simulation time in seconds

NMES - (integer*2, I5) maximum number of messages which may be transmitted (the simulation ends when either SIMT or NMES is reached)

MSIM - (integer*2, I5) number of simulations

MAX - (integer*2, I5) the maximum number of message blocks which may be held in any of the following queues: NQIR, NQBR, NQBT, NQIT. The maximum number of message blocks which may be held in the NQT queue will be 2*MAX. MAX can range from 1 to 43, the maximum corresponding to 16KB of memory. Thus MAX is calculated as follows:

$$MAX = \frac{T.M.B.A.}{6xI.M.B.} = \frac{T.M.B.A.}{384}$$

where T.M.B.A. is the Total Memory Buffer Available in bytes, and I.M.B. the Internal Message Block length (64 bytes).

ISEED1, ISEED2 - (integer*2, I5, integer*2, I5) the two seeds required by the number generator

TBEG - (real*4, F12.3) the elapsed time before the DEBUG option will be enabled

PDELF2 - (real*4, F12.9) the delay within the microprocessor before a message is sent to one of the lines in response to some error. In transmitting side it is used alone and in receiving side it is added to PDELF(j).

PDELF2 = NMC x CLK

where NMC is the number of machine cycles that the microprocessor will take to execute all instructions included in the error recovery routines,
and
CLK is the cycle time in seconds of the microprocessor clock.

ISCALE - (integer*2, I4) should be set to some number other than 0 if the sum of probabilities of interactions (PROB(i)) do not

add up to 1 and are to be scaled accordingly.

3.2.2 Device parameters:

NCARE(i) - (integer*2 array (15), I2) specifies whether any differences among the lines of this device type (if more than 1) are of importance when this type of line is selected for an interaction. If NCARE(i) is 0, then it will be assumed that there is no preference for choosing any of the lines of device type i. If NCARE(i) is not equal to zero and the line selected is busy, then another line of the same device type will not be chosen.

IDUP(i) - (integer*2 array (15), I2) specifies whether interactions on all lines of this device type will be full or half duplex.

DVNAME(i,k) - (logical*1 array(15,12), 12A1) the name assigned to this type of device. (Maximum of 12 characters.)

3.2.3 Line parameters:

IDV(j) - (integer*2 array (25), I2) the type of device represented by line j.

LAMDA1(j) - (real*4 array(25), F9.5) the mean time between character arrivals when line j is receiving. If LAMDA1(j) is zero the line is assumed to be synchronous (i.e. for printers, data base and trunk.)

LAMDA2(j) - (real*4 array(25),F9.5) the mean time between messages received on line j once an interaction has already been initiated.

LAMDA3(j) - (real*4 array (25), F9.5) the mean time between interactions initiated by line j. If LAMDA3(j) is zero, no interactions will be initiated by this line (i.e. for printers data base and trunk).

LAMDA4(j) - (real*4 array(25), F9.5) the utilization of line j when receiving or transmitting synchronously.

WIND(j) - (real*4 array (25), F9.6) the reciprocal of the speed of line j when expressed in characters per second.

JPR(j) - (integer*2 array(25), I2) the relative priority of line j in the NQIR, NQBR, NQBT, NQIT queues.

PDELF(j) - (real*4 array(25),F6.1) the delay factor on line j (expressed as a number of windows) used as a mean to generate the time before the first character of a response will be received.

MSMAX(j) - (integer*2 array (25), I4) the maximum number of characters in any message on line j.

MAXH(j) - (integer*2 array (25), I4) The maximum number of message

responses that may wait in the NQH(j) queue. In other words it is the maximum number of messages that have been transmitted on line j and have not yet been responded (acknowledged). The number of messages in the queue will never exceed 1 if the line is half duplex.

PROT(j) - (integer*2 array(25), I2) the type of protocol represented by an integer number assigned to line j. The trunk may be assumed to have type 0, (standard).

NPIHR(j) - (real*4 array(25), F9.6) the time required to process a character receiving on line j in the NPIH.

NPIHT(j) - (real*4 array (25);F9.6) the time required to process a character transmitting on line j in the NPIH.

PIHR(j) - (real*4 array(25), F9.6) the time required to process a message block consisting of a single character in the PIH receiving layer on line j.

PIHT(j) - (real*4 array(25), F9.6) the time required to process a message block consisiting of a single character in the PIH transmitting layer on line j.

BGR(k,j) - (real*4 array(4,25),F9.6) the time required by the BG layer to process a message block consisting of a single character from receiving line j of protocol type k to a standard protocol.

BGT(k,j) - (real*4 array(4,25), F9.6) the time required by the BG layer to process a message block consisting of a single character from a standard protocol to line j (transmitting) having protocol type k.

Note: the time required to process a character or a message block in any layer is calculated as follows:
Processing time = NMC x CLK
where NMC is the number of machine cycles that the microprocessor will take to execute all instructions included in the routine performing task, and CLK is the cycle time in seconds of the microprocessor clock.

3.2.4 Control parameters (switching matrix):

IDV1(i) - (integer*2 array(25), I2) the initiating device number in row i.

IDV2(i) - (integer*2 array(25),I2) the responding device number in row i.

PROB(i) - (real*4 array(25), F8.5) the probability that the data in row i will be selected when initiating an interaction.

Note: if NLOOPS(i), MSLNT1(i), IFT1(i), MSLNT2(i), IFT2(i) and WAIT(i)

are positive, then NLPS(j), MSLNTH(j), IFT(j) and WAITT(j) will be generated using these numbers as a mean. However, if any of the variables are negative, the assumed value will be the absolute value of this number.

NLOOPS(i) - (integer*2 array(25), I4) the number of messages or files which will be sent in one direction during the interaction.

MSLNT1(i) - (integer*2 array(25), I4) the message length on the initiating device in row i.

IFT1(i) - (integer*2 array(25), I4) the number of lines in a file on the initiating device in row i. If it is the interactive mode, IFT(i) should be either 0 or -1.

MSLNT2(i) - (integer*2 array(25), I4) the message length on the responding device in row i.

IFT2(i) - (integer*2 array(25), I4) the number of lines in a file on the responding device in row i. If it is the interactive mode, IFT2(i) should be either 0 or -1.

WAIT(i) - (real*4 array(25), F9.3) the length of time that the initiating device in row i will wait before a specific line (or any line if NCARE(j) = 0) of the specified responding device is free.

3.3 Output Parameters:

NMTR - (integer*2) the total number of messages transmitted.

NMTRUN(j,i) - (integer*2 array (25,15)) the number of messages transmitted from line j to some other line having a device number of i.

AVDEL(j,i) - (real*8 array(25,15)) the average delay of all messages transmitted from line j to some other line having a device number of i.

ABORT(i,j) - (integer*2 array(6,25))
i=1 - the number of overruns or underruns caused by line j.
i=2 - the number of times that a message had to be repeated in order to prevent the NQIR queue from overflowing.
i=3 - the number of times that a message block could not be transferred to NQBR in order to prevent its overflow.
i=4 - the number of times that a message block could not be transferred to NQBT in order to prevent its overflow.
i=5 - the number of times that a message block could not be transferred to NQIT in order to prevent its overflow.
i=6 - the number of times that a message block could not be transferred to NQT in order to prevent its overflow.

ABORC(i) - (integer*2 array(6)) the sum of ABORT(i,j) as j is varied from 1 to NL.

ABORTS - (integer*2) the sum of ABORC(i) as i is varied from 1 to 6.

THROPT - (real*8) the system throughput for the simulation

$$THROPT = \frac{NMTR}{SIMT} \quad \text{message/sec.}$$

AVDELY - (real*8) the average delay of all messages transmitted during the simulation.

NARMES(j) - (integer*2 array(25)) the number of complete messages received on line j.

NARTOT(j) - (integer*2 array(25)) the number of message blocks received on line j.

NTRMES(j) - (integer*2 array(25)) the number of messages transmitted onto line j from some other line.

K(j) - (integer*2 array(25)) the number of message blocks that have been multiplexed onto the trunk from line j if line j is not the trunk, or the number of message blocks that have been demultiplexed if line j is the trunk.

NT(j) - (integer*2 array(25)) the number of messages of line j in the NQT queue.

NQH(j) - (integer*2 array(25)) the number of messages waiting in queue to begin receiving on line j. In other words it is the number of messages that have been transmitted on line j and have not yet been responded (acknowledged).

NQIR - (integer*2) the number of message blocks waiting in the NQIR queue.

NQBR - (integer*2) the number of message blocks waiting in the NQBR queue.

NQBT - (integer*2) the number of message blocks waiting in the NQBT queue.

NQIT - (integer*2) the number of message blocks waiting in the NQIT queue.

NQT - (integer*2) the number of message blocks waiting in the NQT queue.

STAK - (integer*2) A flag which is set to 1 if the last message block to be processed was interrupted and placed in the stack.

TOTTHR - (real*8) the throughput averaged over all simulations

TOTDEL - (real*8) the average of the average delay for each

simulation.

VAVDLY - (real*8) the variance of the average delay of all messages transmitted during the simulation

$$VAVDLY = \frac{\sum_{j=1}^{NL} (TRTIME(j))^2}{NMTR} - (AVDELY)^2$$

3.4 Intermediate Global Variables:

NDV(i) - (integer*2 array*15) the number of lines of device type number i

NSW - (integer*2) the number of rows in the data switching table

NCB1 - (integer*2) the maximum number of message characters allowed in the first block of a message.

NCB2 - (integer*2) the maximum number of message characters allowed in any block other than the first.

NCTR - (integer*2) the number of characters in each header included with all messages sent on the trunk.

N2 - (integer*2) 2*NL

TIME - (real*8) the elapsed simulation time

EVBG(j) - (real*8) the time of the next arrival on line j

EATW(j) - (real*8) EVBG(j) + WIND(j)

TIMEST(j) - (real*4 array*25) the time that the first character was received on line j. If the message must be repeated for some reason, TIMEST(j) will not be adjusted. If another line cannot be selected when initiating a loop TIMEST(j) is used to determine when the specified time has been exceeded.

NLPS(j) - (integer*2 array *25) the number of loops generated using NLOOPS(j). This is the number of messages that will be sent from line j to its corresponding line.

LSLNT(j) - (integer*2 array*25) the number of the line that is presently corresponding with line j. A negative value of LSLNT(j) after a message has been transmitted onto line j indicates that a file is being sent and more lines can be expected.

ITYPE(j) - (integer*2 array*25) marks the row in the data switching table which contains the information for generating the parameters for a specific line. A negative value of ITYPE(j) indicates that line j is the responding line rather than the initiator

MSLNTH(j) - (integer*2 array*25) the number of characters in a message on line j which have not yet arrived

IFT(j) - (integer*2 array*25) the number of lines in a file which have been generated on line j

WAITT(j) - (real*4 array*25) the allowed waiting time on line j (when line j is attempting to initiate a loop) generated from WAIT(i)

RESETM(j) - (integer*2 array*25) stores the initial message length on line j so that MSLNTH(j) may be reset if the message needs to be repeated

NDEB - (integer*2) a flag that is set when the debug option has not been selected.

RCTIME(j) - (real*8 array*25) accumulates the total NPINR processing time for the message being received on line j

TRTIME(j) - (real*8 array*25) total message delay on line j
= time that message on line j was pulled from last queue (NQT)
- time that the message entered the first queue (NQIR)
+ total NPINR processing time of message
+ total NPINT processing time for all characters transmitted

Since TRTIME(j) accumulates the NPINT processing time while transmitting, it will be equal to the delay of the message by the time that the last character is transmitted.

NR(j) - (integer*2 array*25) the number of internal message blocks required by the message on line j

LC(j) - (integer*2 array*25) the number of characters that will be carried by the last block of the message on line j

PTMI - (real*8) the amount of time before the next earliest character is due to arrive in the NPIH layer

TLEFT - (real*8) the time required by the PIH or BG layers to process a specific message block

KK - (integer*2) the line number of which a character or message block is presently being processed

TIMIN - (real*8) temporarily stores the time that a message block

entered the first queue (NQIR) minus the total NPINR processing time for that message

NCR - (integer*2) temporarily stores the number of characters in a message block

NBP - (integer*2) temporary
= 0 if the message being processed is contained within a single block
= 1 if the block being processed is the first in a message requiring more than one block
= 2 if the block being processed is neither the first nor the last in a message requiring more than two blocks
= 3 if the block being processed is the last in a message requiring more than one block

ITYP - (integer*2) marks the row in the data switching table that contains the information required to generate the various parameters required by a particular line (temporary)

LSLN - (integer*2) temporarily stores the line number to which the message is to be switched

Q - (integer*2) points to the queue (1 to 4) whose message blocks are presently being processed by subroutine PROCES

KKQ(q,i) - (integer*2 array*4*43) stores the line number (KK) in the NQIR, NQBR, NQBT, NQIT queues (q=1 to 4)

TIMEQ(q,i) - (real*8 array*4*43) stores the value TIMIN in the NQIR, NQBR, NQBT, NQIT queues (q=1 to 4)

NCRQ(q,i) - (integer*2 array*4*43) stores the value of NCR in the NQIR, NQBR, NQBT, NQIT queues (q=1 to 4)

NBPQ(q,i) - (integer*2 array*4*43) stores the value of NBP in the NQIR NQBR, NQBT, NQIT queues (q=1 to 4)

ITYPQ(q,i) - (integer*2 array*4*43) stores the value of ITYP in the NQIR, NQBR, NQBT, NQIT queues (q=1 to 4)

LSLNQ(q,i) - (integer*2 array*4*43) stores the value of LSLN in the NQIR, NQBR, NQBT, NQIT queues (q=1 to 4)

KKT(i) - (integer*2 array*86) stores the line number (KK) in the NQT queue

TIMET(i) - (real*8 array*86) stores the value of TIMIN in the NQT queue

NCRT(i) - (integer*2 array*86) stores the value of NCR in the NQT queue

NBPT(i) - (integer*2 array*86) stores the value of NBP in the NQT queue

ITYPT(i) - (integer*2 array *86) stores the value of ITYP in the NQT

queue

LSLNT(i) - (integer*2 array *86) stores the value of LSLN in the NQT queue

KKS - (integer*2) stores the line number (KK) in the stack

TIMES - (real*8) stores the value of TIMIN in the stack

NCRS - (integer*2) stores the value of NCR in the stack

NBPS - (integer*2) stores the value of NBP in the stack

ITYPS - (integer*2) stores the value of ITYP in the stack

LSLNS _ (integer*2) stores the value of LSLN in the stack

NQH(j) - (integer*2 array *25) the number of messages waiting in queue to begin receiving on line j. In other words it is the number of messages that have been transmitted on line j and have not yet been responded (acknowledged).

EVBH(j) - (real*4 array*25) the arrival time of the first character of the message that is first in queue to receive on line j (if the line becomes free before this time)

ITYPH(j,i) - (integer*4 array*25*8) stores the value of ITYP for each message waiting in queue to begin receiving on line j

LSLNH(j,i) - (integer*4 array*25*8) stores the value of LSLN for each message waiting in queue to begin receiving on line j

3.5 Results and Discussions:

A typical set of input parameters used to get some of the sample results reported in this section is given in Figure 6. For all the reported results it is assumed that the WPCC interfaces 10 word processor terminals, 2 line printers, one local data base, one optical character reader and one high speed line.

The buffer behaviour of the WPCC was studied under various traffic conditions such as the input line utilization, mean message length, half/full duplex mode and standard/mixed protocols for different word processors. Figure 7 shows the relationship between the system throughput and average input line utilization for word processors having

```
  5   15   900.000  900    1    5 4567 8934    9999.999 0.000100000   1
0 0 TRUNK
0 1 TERMINAL
0 1 PRINTER
1 1 DATA BASE
0 1 OCR
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
3  0.00000  0.10000  0.00000  0.80000 0.008332 2  30.0  80  25 3
3  0.00000  0.10000  0.00000  0.80000 0.008332 2  30.0  80  25 3
4  0.00000  0.10000  0.00000  0.80000 0.008333 2  30.0  80  25 4
5  0.10000  0.10000 25.00000  0.80000 0.033333 2  30.0  80  25 4
1  0.00000  0.10000  0.00000  1.00000 0.000833 1  25.0 256  25 1
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000040 0.000040 0.000008 0.000008 0.000008 0.000008 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
2 1 0.18000  -1  30   0  30   0    0.000
2 3 0.12000   1  40 100  -2   0    2.000
2 4 0.12000   1  60 100  -2   0   15.000
2 4 0.12000   1  -2   0  60 100   15.000

2 2 0.12000  -1  10   0  10   0   10.000
2 2 0.17000  -1  50  50  -2   0   40.000
2 2 0.17000  -1  -2   0  50  50   40.000
5 2 0.50000  -1  30  80  -2   0  100.000
5 4 0.50000  -1  30  80  -2   0  100.000
```

Figure 6. Typical Input Parameters for the WPCC Computer Simulation Program

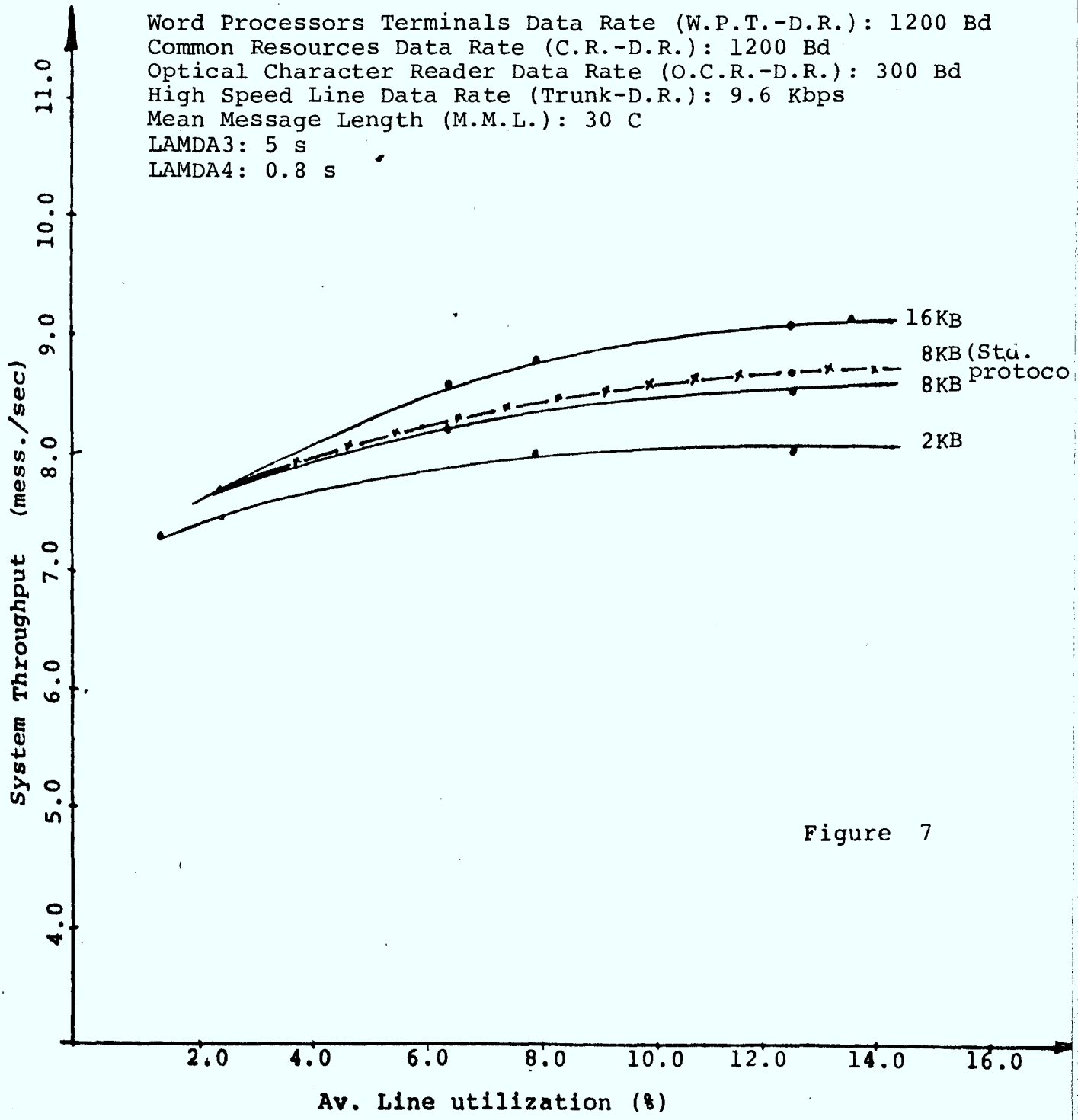
Word Processors Terminals Data Rate (W.P.T.-D.R.): 1200 Bd
Common Resources Data Rate (C.R.-D.R.): 1200 Bd
Optical Character Reader Data Rate (O.C.R.-D.R.): 300 Bd
High Speed Line Data Rate (Trunk-D.R.): 9.6 Kbps
Mean Message Length (M.M.L.): 30 C
LAMDA3: 5 s
LAMDA4: 0.8 s
16KB
8KB (Std. protoco
8KB
2KB
System Throughput (mess./sec)
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
Av. Line utilization (%)

Figure 7

dissimilar protocols. The curves show that the system throughput increases with the increase in the line utilization (which is expressed by WIND/LAMDA1). The rate of increase in throughput depends on the memory buffer available in the WPCC. The reason for this increase in throughput is that the messages (which caused overflow of buffer with 2KB of memory) are accommodated in the queues and processed when the memory size is increased to 16KB. It is also observed that the rate of increase in throughput decreases when the line utilization is increased more than 10%. This is due to the congestion occurring in the system causing more number of failures. A very similar relationship was observed between the average message delay and the average line utilization for different buffer size (Figure 8). The average message delay increases directly as the average line utilization increases and the rate of increase depends on the memory buffer available for date storage. The reason for this is that as the line utilization increases (mean character interarrival time decreases) a greater number of interrupts occur in the PIH and BG layers which results in an increase in the holding time of a message in the system. Figures 7 and 8 also show the comparison of both throughput and average message delay versus average line utilization in the case of WPCC having all the word processors of same (standard) protocols and WPCC having WP's of different protocols. The curves show that for a fixed memory size of 8KB, lower throughput and higher message delay is obtained in case of the WPCC having WP's of different protocols. This is due to the protocol translation which is necessary for communication among dissimilar word processors. The protocol translation requires a higher processing time in the Background layer which causes an increase in

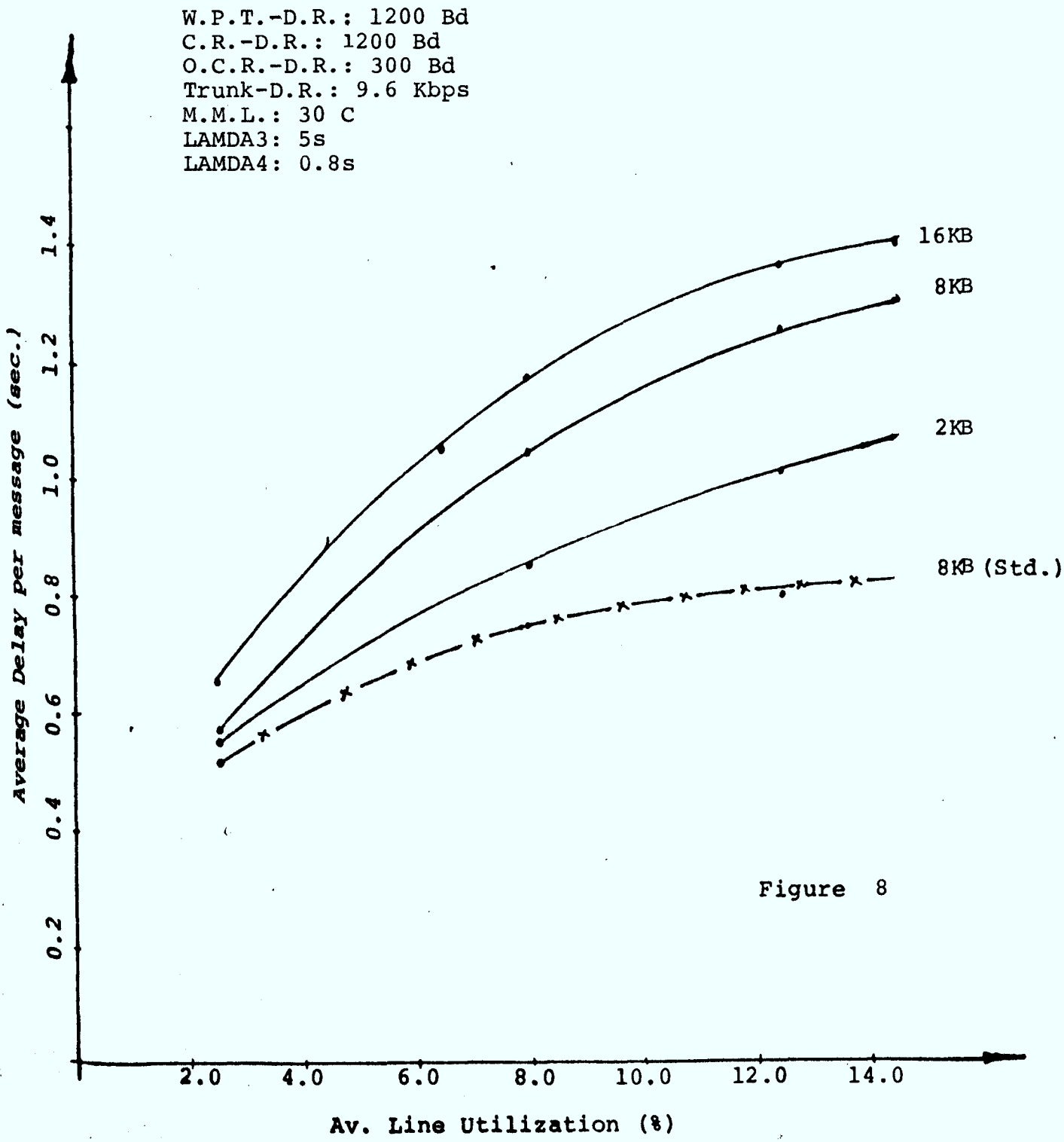
W.P.T.-D.R.: 1200 Bd
C.R.-D.R.: 1200 Bd
O.C.R.-D.R.: 300 Bd
Trunk-D.R.: 9.6 Kbps
M.M.L.: 30 C
LAMDA3: 5s
LAMDA4: 0.8s
16KB
8KB
2KB
8KB (Std.)
Average Delay per message (sec.)
0.2
0.4
0.6
0.8
1.0
1.2
1.4
2.0
4.0
6.0
8.0
10.0
12.0
14.0
Av. Line Utilization (%)

Figure 8

average message delay which subsequently decreases the system throughput.

Using the same set of input parameters given above and keeping the character interarrival time on low speed lines at 0.05 S with all the WP's having the same protocols, the variation in the system throughput with the increase in the mean message length is shown in Figure 9. The curves reflect that the system throughput decreases with the increase in the message length. However, the rate of decrease depends on the size of memory available. The reason for this decrease is that the processing and transmitting time of a message varies directly with the length of the message, which causes the decrease in system throughput. As the graph indicates the rate of fall of system throughput decreases with the increase in the buffer size. This is due to the fact that the messages (which were rejected when the memory size was 0.5KB) are processed when the memory is increased to 2KB. Figure 10 shows the relationship between the probability of data loss and the memory size for different mean message length.

Probability of data loss = number of messages lost/(number of messages lost + number of messages transmitted),

i.e.

probability of data loss = (ABORC(1)+ABORC(2))/(ABORC(1)+ABORC(2)+NMTR).

The curves show that the probability of data loss decreases exponentially as the memory size is increased. At a certain value of memory size (4KB in this case) the number of aborts reduces to zero resulting in the probability of data loss to be less than 10^{-6}. Figure 11 shows the relation between the system throughput and the data rates of the WP's terminals for different buffer size. The speeds of trunk

W.P.T.-D.R.: 1200 Bd
C.R.-D.R.: 1200 Bd
O.C.R.-D.R.: 300 Bd
Trunk-D.R.: 2.4 Kbps
LAMDA1, LAMDA2: 0.05 s
LAMDA3: 5 s
LAMDA4: 0.8 s

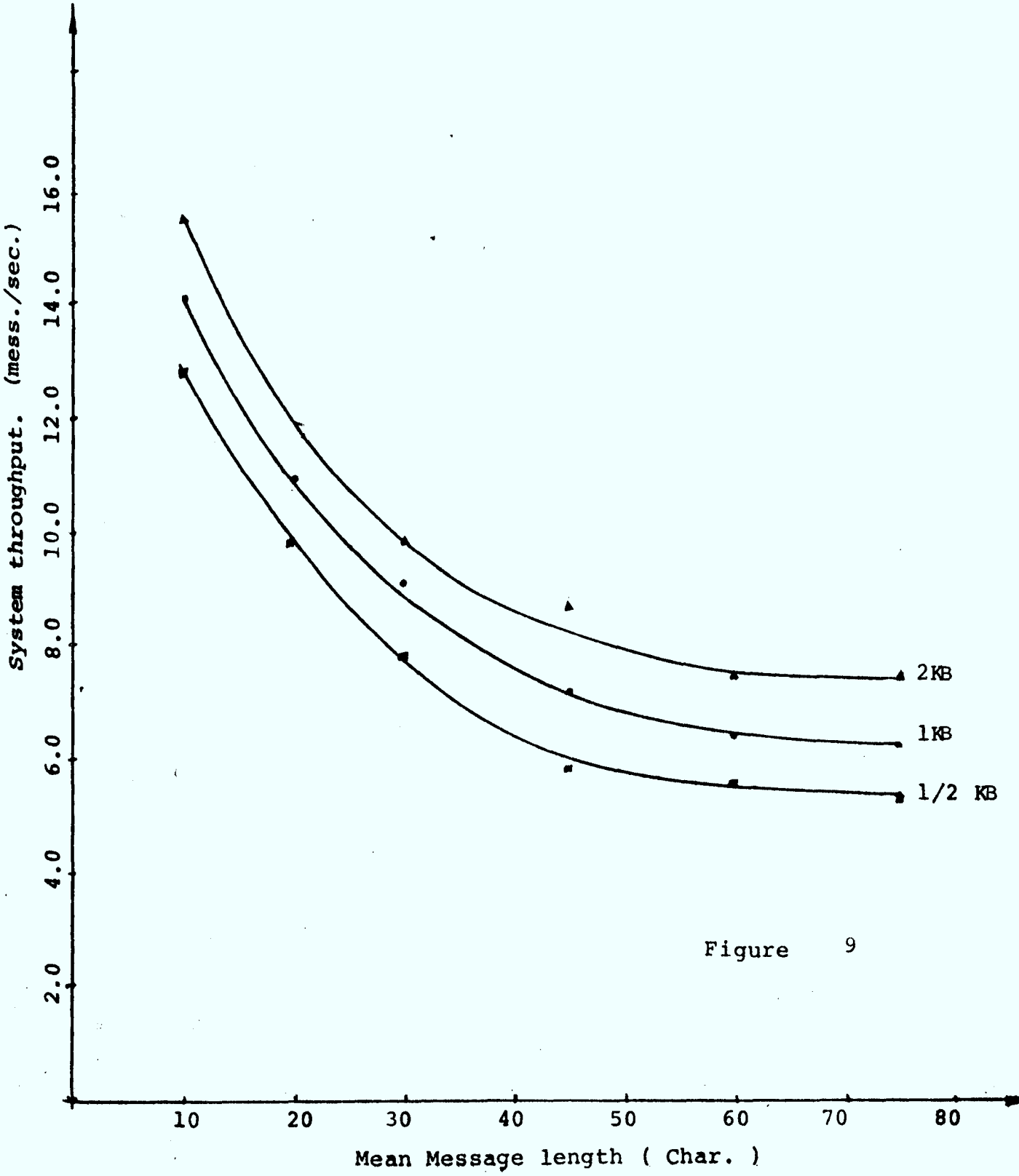

Figure 9

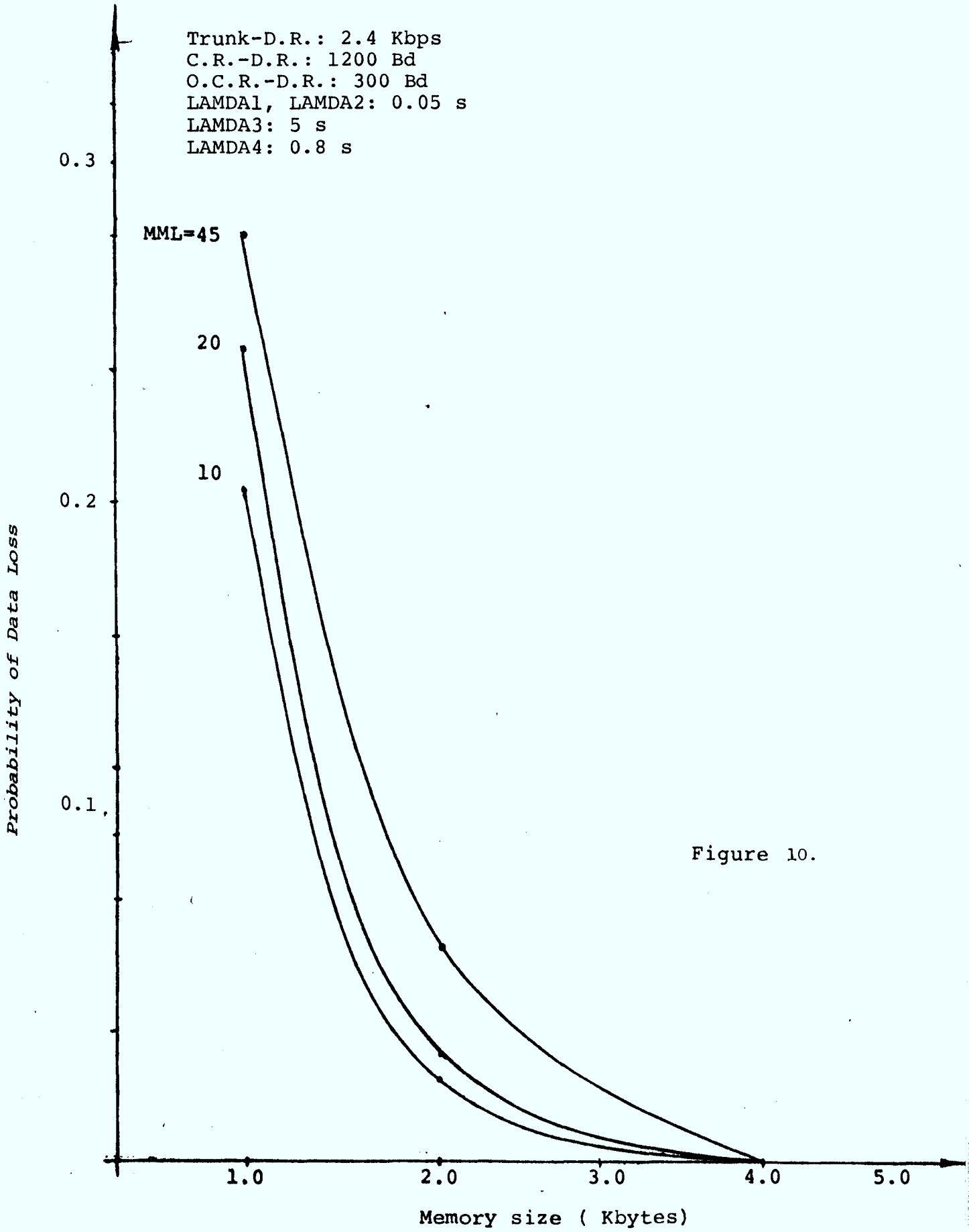
Trunk-D.R.: 2.4 Kbps
C.R.-D.R.: 1200 Bd
O.C.R.-D.R.: 300 Bd
LAMDA1, LAMDA2: 0.05 s
LAMDA3: 5 s
LAMDA4: 0.8 s
MML=45
20
10
0.3
0.2
0.1
Probability of Data Loss
1.0
2.0
3.0
4.0
5.0
Memory size (Kbytes)

Figure 10.

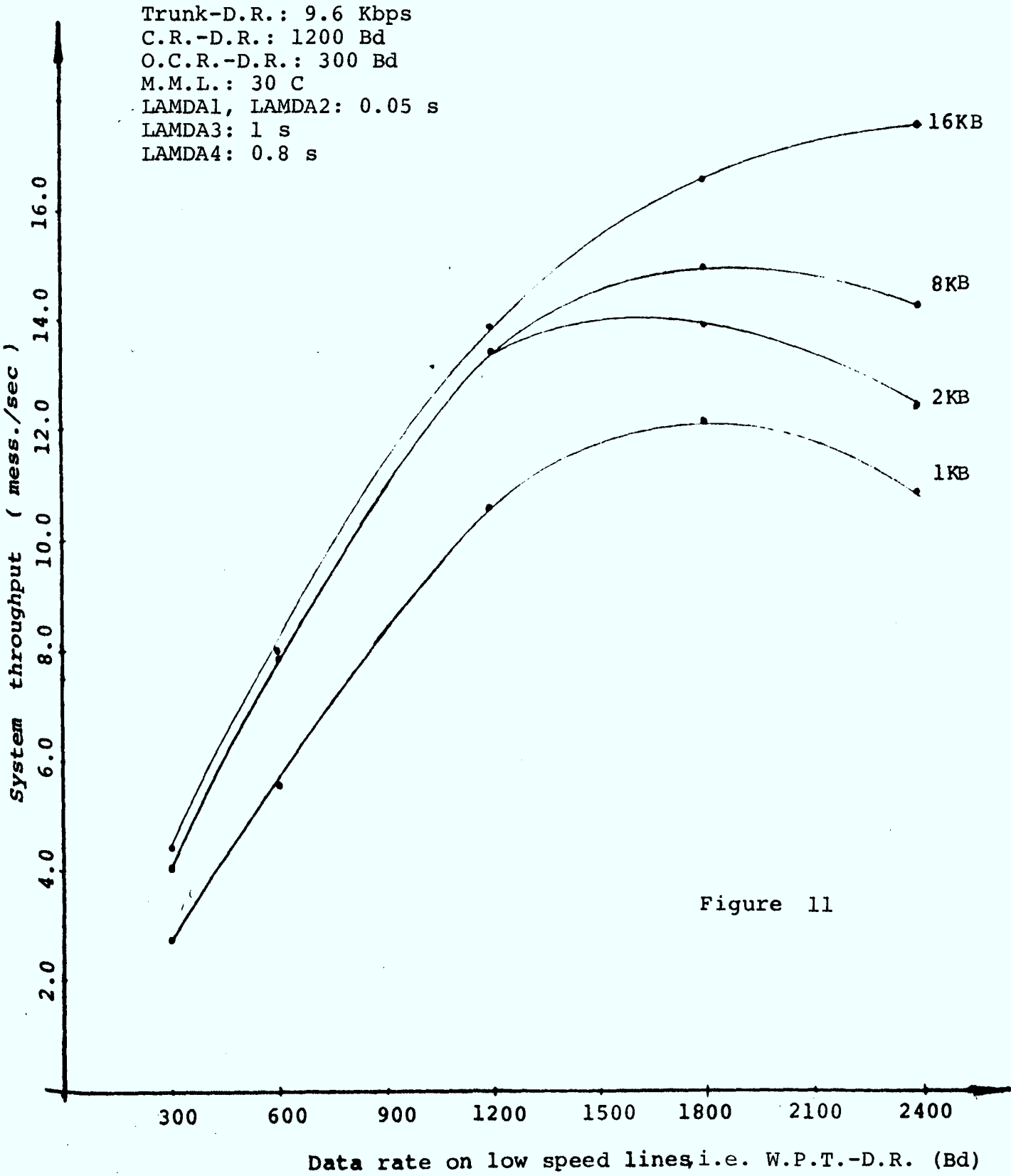
Trunk-D.R.: 9.6 Kbps
C.R.-D.R.: 1200 Bd
O.C.R.-D.R.: 300 Bd
M.M.L.: 30 C
LAMDA1, LAMDA2: 0.05 s
LAMDA3: 1 s
LAMDA4: 0.8 s
16KB
8KB
2KB
1KB
System throughput (mess./sec)
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
300
600
900
1200
1500
1800
2100
2400
Data rate on low speed lines, i.e. W.P.T.-D.R. (Bd)

Figure 11

line, database link, line printers and OCR were 9.6 Kbps, 1200Bd, 1200Bd and 300Bd respectively. The curves depict that for a given buffer size, the system throughput initially increases with the increase of data rate on the low speed lines (word processors) but later starts to decrease when the data rate is increased beyond 1800Bd. For example, at a memory size of 1KB the throughput decreases sharply when the data rate is increased beyond 1800Bd but considerably higher throughput is achieved when the memory is increased to 16KB. This is due to the fact that for small buffer size of 1KB the system cannot handle 10 WP's of 1800Bd each because the traffic intensity increases to such an extent that congestion occurs in the system resulting in the overflow of queues and thus fall in the throughput. However, when more memory is available (i.e. 16kB), data rates higher than 1800Bd can be handled. These curves are very useful to systems engineers for developing an optimal system configuration with high throughput and reasonable low delay per message.

Summary and Conclusions:

A computer simulation package has been developed as a tool to determine the hardware/software configurations of a microprocessor-based communications controller for word processors. The computer program evaluates the performance of this communications controller under different conditions. The results of the performance evaluation study would help both the Designer and Systems Manager. For the Designer the computer program can define for him the most suitable hardware/software configuration for a specific type of application. This includes defining the type of microprocessor to be used, the maximum number of different lines and their type of interfaces (e.g. synchronous or

asynchronous, and half duplex or full duplex) that this specific microprocessor can handle, and the minimum buffer requirements for this hardware/software configuration. For the Systems Manager who would like to buy a similar system the computer simulation program can compare the different communications controllers available on the market that can implement the WPCC functions. Also, once the Systems Manager would have such equipment in operation, this computer program can define for him the system requirements of the WPCC. These include the maximum number of word processors and common resource equipments and the maximum data rate and traffic on all lines that this specific WPCC can handle reliably.

5. References:

[1] R.J.A. Buhr and D.A. Mackinnon, "Word Teleprocessing Interface", MSDL Tech. Note 80-1, prepared for the Department of Communications, March 31, 1980.

[2] C. Weitzman, "Distributed Micro-minicomputer Systems", Prentice-Hall Inc., N.J. 1980.

[3] D. Ferrari, "Computer Systems Performance Evaluation", Prentice-Hall Inc., N.J. 1978.

[4] M. Schwartz, "Computer-communications Network Design and Analysis", Prentice-Hall Inc., N.J. 1977.

[5] G.G. Iazeolla, E. Martinelli and O. Tedone, "Production and Representation of Computer System Simulation Models", Symposium of the Simulation of Computer Systems IV, August 1976, pp. 205-211.

[6] P.D. Cherry, "Word Processing System Design for High Throughput", Computer Design, August 1980, pp. 95-99.

[7] H.J. Hindin, "Controlling the Electronic Office: PBXs Make Their Move", Electronics, April 1981, pp. 139-148.

[8] H.J. Hindin, "Controlling Data Communications: Statistical Multiplexers Move In", Electronics, July 1981, pp. 141-148.

[9] D.R. McGlynn, "Distributed Processing and Data Communications", John Wiley and Sons, N.Y. 1978.

[10] W.W. Chu, "Buffer Behaviour for Batch Poisson Arrivals and Single Constant Output", IEEE Transactions on Communications, Vol. COM-18, October 1970, pp. 613-618.

[11] L. Kleinrock, "Queueing Systems: Vol. II", John Wiley and Sons, N.Y., 1976.

[12] T.H. Gordon, J.I. Meltzer and R.J. Pilc, "Design and Performance of a Statistical Multiplexer", Conference Record ICC '70, pp.28.9-28.21.

APPENDIX A

Computer Simulation Program Listing

```
C          WORD PROCESSOR COMMUNICATION CONTROLLER  SIMULATION
C                                ===============
C                                SWITCH - VERSION 3
C
C
C
C    Maximum message length specified in input data.
C The first block of each message may contain a max. of 38 chars.
C Following blocks may contain upto 62 characters.
C Double precision temporary variables
C
C If a block will cause an overflow in the next queue, it will not be
C processed untill this queue is no longer full. An attempt is made
C to reduce the size of the queue by temporarilly adjusting the
C normal level of priorities.
C
C
C the delay per message  on a particular line is now calculated by
C noting the time that a message is put into the first queue and by
C subtracting this number from the time the message is taken out of
C the last queue. this number is added to the NPINR and NPINT processing
C times to give the system delay of the message.
C A new set of control variables must now be included in the input data
C to specify all types of interractions that may occur.
C the receiving and transmitting message blocks in the nqi and nqb queue
C have now been separated into receiving and transmitting sections:
C NQIR, NQBR, AND NQIT AND NQBT. Operations are performed on these
C four queues by subroutines PUSHQ and PULLQ.
C Each line of a particular type of DEVICE may be specified as full or
C half duplex.
C When a line is transmitting SYNCHRONOUSLY, sync characters may be sent
C depending on the utilisation of the line.
C
C VERSION 3 UPDATED BY PROVIDING VARIANCE OF MESSAGE DELAY 1 MAY 82
C
C=================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      REAL VAVDEL(25,10)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
```

```
      DIMENSION NQH(25),MAXH(25),ITYPH(25,25),LSLNH(25,25),EVBH(25)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTM1,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
      COMMON /BLK2/ TOTTHR,TOTDEL
C
      COMMON /BLK3/ NQH,MAXH,ITYPH,LSLNH,EVBH
C
      COMMON /BLK5/ VAVDEL,TVARDY
C=====================================================================
C LOCAL VARIABLES
      DOUBLE PRECISION TOTTHR,TOTDEL
      REAL DMY(3),PRSUM(15)
      LOGICAL*1 YES,ANSWER
C
      DATA YES/'Y'/
C THE KEYBOARD IS BY DEFAULT UNIT 5
C THE SCREEN IS BY DEFAULT UNIT 6
C ASSIGN THE UNIT NUMBER 7 TO LOGICAL DEVICE LP: (THE PRINTER)
      WRITE(6,7041)
 7041 FORMAT(' ENTER THE NAME OF THE MAIN OUTPUT FILE.',/)
      CALL ASSIGN(7,,-1,'NEW')
C A PLOT FILE IS USED TO ACCUMULATE THE RESULTS FOR MULTIPLE RUNS
      WRITE(6,135)
  135 FORMAT(' ENTER THE NAME OF THE PLOT FILE.',/)
C THIS NAME MUST ALREADY BE INCLUDED IN THE DIRECTORY
      CALL ASSIGN(8,,-1,'OLD')
      WRITE(6,7040)
 7040 FORMAT(' ENTER THE INPUT DATA FILE NAME.',/)
C ASSIGN THE UNIT NUMBER 4 TO A DATA FILE FOR INPUT ONLY.
C THE FILENAME OF THE DATA FILE MUST BE ENTERED FROM THE KEYBOARD.
      CALL ASSIGN(4,,-1,'RDO')
C THE DEBUG OUTPUT IS PRINTED ON THE DEVICE ASSIGNED TO UNIT 6
C SO THAT THE PROGRAM PROCEDURE MAY BE TRACED
      NDEB=1
      CALL DATE(DMY)
      WRITE(7,3744) DMY
      WRITE(6,3744) DMY
 3744 FORMAT(2X,2A4,A1,/)
C INPUT GENERAL SYSTEM PARAMETERS
      READ(4,400)NDEV,NL,SIMT,NMES,MSIM,MAX,ISEED1,ISEED2,TBEG,PDELF2,
     +ISCALE
      WRITE(7,400)NDEV,NL,SIMT,NMES,MSIM,MAX,ISEED1,ISEED2,TBEG,PDELF2,
     +ISCALE
      IF(NDEB.EQ.0)WRITE(6,400)NDEV,NL,SIMT,NMES,MSIM,MAX,ISEED1,
     +ISEED2,TBEG,PDELF2,ISCALE
C INPUT DEVICE PARAMETERS (ONE LINE FOR EACH TYPE OF DEVICE)
```

```
      DO 4622 I=1,NDEV
      READ(4,3001)NCARE(I),IDUP(I),(DVNAME(I,J),J=1,12)
      WRITE(7,3001)NCARE(I),IDUP(I),(DVNAME(I,J),J=1,12)
      IF(NDEB.EQ.0)WRITE(6,3001)NCARE(I),IDUP(I),(DVNAME(I,J),J=1,12)
      NDV(I)=0
 4622 CONTINUE
 3001 FORMAT(2I2,1X,12A1)
      NPROT=0
      DO 10 I=1,NL
C     READ IN THE PARAMETERS FOR EACH LINE
      READ(4,401) IDV(I),LAMDA1(I),LAMDA2(I),LAMDA3(I),LAMDA4(I),
     1 WIND(I),JPR(I),PDELF(I),MSMAX(I),MAXH(I),PROT(I)
C     WRITE THEM AGAIN FOR CHECKING.
      WRITE(7,401) IDV(I),LAMDA1(I),LAMDA2(I),LAMDA3(I),LAMDA4(I),
     1 WIND(I),JPR(I),PDELF(I),MSMAX(I),MAXH(I),PROT(I)
      NDV(IDV(I))=NDV(IDV(I))+1
      IF(PROT(I).GT.NPROT)NPROT=PROT(I)
   10 CONTINUE
  401 FORMAT(I2,4F9.5,F9.6,I2,F6.1,2I4,I2)
C
      N=2
      IF(N.GT.NPROT)N=NPROT
      DO 20 I=1,NL
      READ(4,402) NPIHR(I),NPIHT(I),PIHR(I),PIHT(I),(BGR(J,I),BGT(J,I),
     + J=1,N)
      WRITE(7,402)NPIHR(I),NPIHT(I),PIHR(I),PIHT(I),(BGR(J,I),BGT(J,I),
     + J=1,N)
   20 CONTINUE
      IF(N.EQ.NPROT)GO TO 3852
      DO 8213 I=1,NL
      READ(4,402) (BGR(J,I),BGT(J,I),J=3,NPROT)
      WRITE(7,402) (BGR(J,I),BGT(J,I),J=3,NPROT)
 8213 CONTINUE
 3852 CONTINUE
  402 FORMAT(8F9.6)
C BY INSERTING THE FOLLOWING CODE THE INPUT DATA WILL BE ECHOED TO
C TO THE SCREEN ONLY IF DEBUG IS CHOSEN.
      IF(NDEB.NE.0)GO TO 1003
      WRITE (6,403)
      DO 30 I=1,NL
      WRITE(6,9000) I,(DVNAME(IDV(I),J),J=1,12),LAMDA1(I),LAMDA2(I),
     $LAMDA3(I),LAMDA4(I),WIND(I)
   30 CONTINUE
 9000 FORMAT(I4,3X,12A1,4F8.5,F9.6)
      WRITE(6,404)
      DO 40 I=1,NL
      WRITE(6,9001) I,JPR(I),PDELF(I),MSMAX(I),MAXH(I),NPIHR(I),NPIHT(I)
   40 CONTINUE
 9001 FORMAT(2I5,F7.1,I7,I6,2F10.6)
      WRITE(6,2388) (I,I,I=1,3)
 2388 FORMAT('     PIHR     PIHT',3(5X,'BGR',I1,5X,'BGT',I1))
      N=3
      IF(N.GT.NPROT)N=NPROT
      DO 2112 I=1,NL
      WRITE(6,402) PIHR(I),PIHT(I),(BGR(J,I),BGT(J,I),J=1,N)
 2112 CONTINUE
  400 FORMAT (2I5,F10.3,5I5,F12.3,F12.9,I4)
```

```
  403 FORMAT (' LINE  DEVICE        LAMDA1  LAMDA2  LAMDA3  LAMDA4
     $   WINDOW')
  404 FORMAT (' LINE  JPR  PDELF  MSMAX  MAXH     NPIHR     NPIHT')
 1003  CONTINUE
C READ IN CONTROL PARAMETERS (ONE LINE FOR EACH TYPE OF INTERRACTION)
       IF(NDEB.EQ.0)WRITE(6,9292)
       DO 3933 I=1,NDEV
 3933  PRSUM(I)=0
       I=1
 9289  READ(4,9290,END=83)IDV1(I),IDV2(I),PROB(I),NLOOPS(I),MSLNT1(I),
     + IFT1(I),MSLNT2(I),IFT2(I),WAIT(I)
 9290  FORMAT(2I2,F8.5,5I4,F9.3)
       IF(PROB(I).EQ.0)GO TO 83
       PRSUM(IDV1(I))=PRSUM(IDV1(I))+PROB(I)
       WRITE(7,9290)IDV1(I),IDV2(I),PROB(I),NLOOPS(I),MSLNT1(I),IFT1(I),
     + MSLNT2(I),IFT2(I),WAIT(I)
       IF(NDEB.NE.0)GO TO 9293
       WRITE(6,9291) (DVNAME(IDV1(I),J),J=1,12),(DVNAME(IDV2(I),J),J=1,
     + 12),PROB(I),NLOOPS(I),MSLNT1(I),IFT1(I),MSLNT2(I),IFT2(I),WAIT(I)
 9293  CONTINUE
 9292  FORMAT(' INIT.DEVICE  RESP.DEVICE   PROB  NLOOPS  MSLNT1  IFT1
     +  MSLNT2  IFT2     WAIT')
 9291  FORMAT(1X,12A1,2X,12A1,F6.3,2I8,I6,I8,I6,F9.3)
       I=I+1
       GO TO 9289
C
   83  NSW=I-1
       IF(ISCALE.EQ.0)GO TO 2947
       WRITE(7,6543)
 6543  FORMAT(' PROBABILITIES AFTER SCALING:')
       DO 2948 I=1,NSW
       PROB(I)=PROB(I)/PRSUM(IDV1(I))
 2948  WRITE(7,5432) PROB(I)
 5432  FORMAT(F10.6)
C
C      INITIALIZE VARIABLES AND TIMES PRIOR TO EACH SIMULATION.
C
 2947  NCB1=38
       NCB2=62
       NCTR=8
C
C INITIALIZE THE VARIABLES FOR AVERAGING OVER MSIM.
       TOTDEL=0.0
       TOTTHR=0.0
       TVARDY=0.
C
       N1=NL+1
       N2=NL*2
C LINE NUMBERS N1 TO N2 ARE USED TO REPRESENT LINES 1 TO NL WHEN
C  TRANSMITTING
C ENTER THE LOOP WHICH IS TO BE REPEATED FOR EACH SIMULATION
       DO 100 MSI=1,MSIM
       WRITE(7,431)MSI
       WRITE(6,431)MSI
  431  FORMAT (5X,'SIMULATION NO.',I5)
       TIME=0.
       TIMT=0.
```

```
22900   C
23000   C TIMT is used to notify the user each time 10 seconds of simulation
23100   C time has elapsed.
23200   C All transmitting lines will be idle at the beginning of the
23300   C simulation therefore the next character arrival time is set
23400   C to a high number so that it will not be detected by SUBROUTINE ERLST
23500   C
23600         DO 110 J=N1,N2
23700         EVBG(J)=1E5
23800         EATW(J)=1E5
23900     110 CONTINUE
24000   C
24100   C     INITIALIZE VARIABLES
24200   C
24300         DO 810 J=1,NL
24400         NT(J)=0
24500         NARMES(J)=0
24600         NTRMES(J)=0
24700         NARTOT(J)=0
24800         K(J)=0
24900         DO 1462 I=1,NDEV
25000         NMTRUN(J,I)=0
25100         VAVDEL(J,I)=0.
25200    1462 AVDEL(J,I)=0.
25300     810 CONTINUE
25400         STAK=0
25500         DO 65 I=1,4
25600      65 NQ(I)=0
25700         NQT=0
25800         NMTR=0
25900         NMTC=0
26000         NRCTR=0
26100         DO 46 J=1,N2
26200         DO 66 I=1,6
26300      66 ABORT(I,J)=0
26400      46 CONTINUE
26500         DO 830 J=1,NL
26600         NQH(J)=0
26700         TIMEST(J)=0
26800         NLPS(J)=0
26900         LSLNT(J)=0
27000         ITYPE(J)=0
27100         MSLNTH(J)=0
27200         RCTIME(J)=0
27300   C  IF LAMDA3(J) IS ZERO, LINE J WILL NOT INITIATE A LOOP AND MUST WAIT
27400   C   SELECTED BY SOME OTHER LINE
27500         IF(LAMDA3(J).LE.0.)GO TO 2824
27600         CALL NEXT(J,LAMDA3(J),-1,-1)
27700         GO TO 830
27800    2824 EVBG(J)=1E5
27900         EATW(J)=1E5
28000     830 CONTINUE
28100       1 IF(TIME.LT.TIMT)GO TO 9051
28200   C  NOTIFY THE USER HOW MUCH TIME HAS ELAPSED AND HOW MANY MESSAGES HAVE
28300   C   BEEN TRANSMITTED
28400         WRITE(6,9052)TIME,NMTR
28500    9052 FORMAT(' TIME =',F8.0,5X,'NMTR =',I5)
```

```
      TIMT=TIMT+10.
 9051 CONTINUE
      IF(NMTR.LT.NMTC)GO TO 318
C PRINT A TABLE OF THE INTERRACTIONS TAKING PLACE IF THE DEBUG OPTION HAS
C  BEEN CHOSEN
      IF(NDEB.NE.0)GO TO 612
      WRITE(6,319)
  319 FORMAT('    LINE  LSLNT   NLPS   IFT  MSLNTH  ITYPE')
      DO 321 J=1,NL
      IF(NLPS(J).LE.0)GO TO 321
      WRITE(6,327) J,LSLNT(J),NLPS(J),IFT(J),MSLNTH(J),ITYPE(J)
  321 CONTINUE
  327 FORMAT(6I7)
  612 NMTC=NMTC+5
  318 CONTINUE
C CALL SUBROUTINE ERLST TO WAIT UNTILL THE FIRST CHARACTER ARRIVAL IS
C DETECTED ON EITHER A RECEIVING OR TRANSMITTING LINE
      CALL ERLST(N2,KK,EVBG)
C
C KK IS THE SUBSCRIPT OF THE LINE WITH THE EARLIEST EVENT
C RETURNED FROM ERLST
C
      IF(NDEB-1)1005,1006,1006
 1005 WRITE(6,9053)KK,EVBG(KK)
 9053 FORMAT(' *****',I3,F15.8,' *****')
 1006 CONTINUE
C IF EATW(KK)  (time of arrival plus window) IS LESS THAN THE PRESENT TIME
C  THE CHARACTER CAN NO LONGER BE PICKED UP.
      IF(EATW(KK).GE.TIME)GO TO 6012
C AN OVERRUN OR UNDERRUN HAS OCCURRED
C  THE ENTIRE MESSAGE MUST BE REPEATED
      IF(NDEB.EQ.1)GO TO 6014
      WRITE(6,6015) KK
 6015 FORMAT(' *** OVER-RUN OR UNDER-RUN ON LINE',I3,' ***')
      WRITE(6,6016)
 6016 FORMAT(' REPEATING MESSAGE')
 6014 ABORT(1,KK)=ABORT(1,KK)+1
      KI=KK
      IF(KI.GT.NL)KI=KI-NL
      IF(ITYPE(KI).EQ.0.AND.KK.LE.NL)GO TO 114
      MSLNTH(KK)=RESETM(KK)
C
C Reset message length. Use the delay factors to set
C the time when the message will begin re-sending the message.
C
  114 EVART=PDELF2
      IF(KK.GT.NL)GO TO 6013
      EVART=EVART+PDELF(KI)*WIND(KI)
 6013 EVBG(KK)=TIME+EVART
      EATW(KK)=EVBG(KK)+WIND(KI)
      GO TO 1
 6012 CONTINUE
      IF(TIME.GE.TBEG)NDEB=0
C
C When time gets bigger than TBEG then start to debug.
C If the time or number of messages transmitted are greater than
C the specified values in the simulation.
```

```
C
      IF(TIME.GT.SIMT .OR. NMTR.GE.NMES) GO TO 81
       IF(TIME.LT.EVBG(KK))TIME=EVBG(KK)
C IF NEXT EVENT IS ON A RECEIVING LINE
       IF(KK.GT.NL)GO TO 4
C
C Call NPIH for recieving chars.
C
      CALL NPINR
      GO TO 102
C
C Call NPIH for the transmitting side.
C
    4 CALL NPINT
C
C Subroutine ERLST is called to determine how much time will elapse
C (PTMI) before the next earliest character will arrive.
C
C
  102 CALL ERLST(N2,KK,EVBG)
      PTMI=EVBG(KK)-TIME
      CALL PROCES
      GO TO 1
C
C     GATHER STATISTICS
C
   81 IF(NDEB.EQ.0)WRITE(6,423)
  423 FORMAT (5X,'STATISTICS')
C
      CALL GASTAT
C
  100 CONTINUE
      TOTDEL=TOTDEL/MSIM
      TVARDY=TVARDY/MSIM

      WRITE(7,9054) MAX,ABORTS,TOTDEL,TOTTHR
 9054 FORMAT(' MAX',I4,' ABORTS',I8,' T.DELAY',F10.6,' T.THRUPUT',
     + F12.6,/)
      WRITE(7,9056) TVARDY
 9056 FORMAT(' VAR. OF MESSAGE DELAY',F23.7)
      CALL CLOSE(7)
      STOP
      END
```

```
      SUBROUTINE NPINR
C
C=================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      DIMENSION NQH(25),MAXH(25),ITYPH(25,25),LSLNH(25,25),EVBH(25)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
      COMMON /BLK3/ NQH,MAXH,ITYPH,LSLNH,EVBH
C
C=================================================================
C
C                                   Processing time of this layer NPIHR
C                                   cannot be interrupted.
C
      TIME=TIME+NPIHR(KK)
C                                   Processing time of each character
C                                   in a message is added to calculate
C                                   the message delay.
C
      RCTIME(KK)=RCTIME(KK)+NPIHR(KK)
C
C                                   If ITYPE(KK) is 0,a row in the data
C                                   switching table is selected before
C                                   before continuing.
C
      IF(ITYPE(KK).NE.0)GO TO 3050
      RNDM1=RAN(ISEED1,ISEED2)
      RNDM2=0
```

```
      ITYP=1
 3031 IF(IDV1(ITYP).NE.IDV(KK))GO TO 3032
      RNDM2=RNDM2+PROB(ITYP)
      IF(RNDM1.LT.RNDM2)GO TO 3033
 3032 ITYP=ITYP+1
      IF(ITYP.LE.NSW)GO TO 3031
      EVBG(KK)=100000.
      EATW(KK)=100000.
      GO TO 39
C THE ROW THAT HAS BEEN SELECTED IS ITYP
 3033 ITYPE(KK)=ITYP
C                                    LSLNT=0 indicates destination is yet
C                                    to be selected.
      LSLNT(KK)=0
      TIMEST(KK)=EVBG(KK)
      WAITT(KK)=ABS(WAIT(ITYP))
      IF(WAIT(ITYP).GT.0)WAITT(KK)=POIRAN(WAIT(ITYP),0.,2*WAIT(ITYP))
C
C                                    Call NEXT to generate message length
C                                    an file size.
C
      CALL NEXT(KK,0.,0,0)
      IF(NDEB.NE.0)GO TO 3050
      WRITE(6,682) KK
  682 FORMAT(' NEW MESSAGE RECEIVING ON LINE',I3)
      WRITE(6,683)
  683 FORMAT(' DESTINATION  ITYP  MSLNTH  IFT')
      WRITE(6,684)  (DVNAME(IDV2(ITYP),J),J=1,12),ITYP,MSLNTH(KK),
     +IFT(KK)
  684 FORMAT(1X,12A1,I5,I8,I5)
C                                    Check if a file is received,if so
C                                    the interarrival time is the wind of
C                                    the line.
 3050 EVMN=LAMDA1(KK)
C CHECK IF A FILE IS BEING RECEIVED
      IF(ITYPE(KK).GT.0)IFTT=IFT1(ITYPE(KK))
      IF(ITYPE(KK).LT.0)IFTT=IFT2(-ITYPE(KK))
C IF RECEIVING A FILE THE CHARACTER INRERARRIVAL TIME MUST BE GENERATED
C  USING THE WINDOW. -EVMN MUST BE THE UTILISATION
      IF(IFTT.GT.0.OR.EVMN.LE.0)EVMN=-LAMDA4(KK)
      CALL NEXT(KK,EVMN,-1,-1)
      MSLNTH(KK)=MSLNTH(KK)-1
      IF(MSLNTH(KK).GT.0)GO TO 39
C
C END OF MESSAGE
C
      IF(NDEB.EQ.0)WRITE(6,280) (DVNAME(IDV(KK),J),J=1,12),KK
  280 FORMAT(' MESSAGE ARRIVAL ON ',12A1,', LINE',I3)
      NARMES(KK)=NARMES(KK)+1
      NARTOT(KK)=NARTOT(KK)+NR(KK)
      ITYP=ITYPE(KK)
C IF LSLNT(KK) IS ZERO, A LINE (WHERE THE MESSAGE WILL BE SENT) MUST BE
C  SELECTED BEFORE CONTINUING
      IF(LSLNT(KK).GT.0)GO TO 3051
C IRNDM IS THE OCCURENCE (EX. FIRST,SECOND,...) OF A LINE OF THE SPECIFIEI
C  DEVICE TYPE THAT WILL BE SELECTED, WHEN SEARCHING FROM 1 TO NL.
      IRNDM=1
```

```
C IF NCARE FOR THE DEVICE IS NOT 0, IRNDM WILL BE GENERATED RANDOMLY
      IF(NCARE(IDV2(ITYP)).NE.0)IRNDM=1+NDV(IDV2(ITYP))*RAN(ISEED1,
     +ISEED2)
C LSLNT(KK) WILL BE NEGATIVE IF LINE KK IS WAITING FOR A PARTICULAR LINE
C  BE FREE. THIS SAME LINE CAN BE CHECKED AGAIN BY SETTING IRNDM EQUAL TO
C  THE ABSOLUTE VALUE OF LSLNT(KK).
      IF(LSLNT(KK).LT.0)IRNDM=-LSLNT(KK)
  382 I=0
      DO 3017 J=1,NL
C CHECK IF DEVICE TYPE OF LINE J MATCHES THAT WHICH HAS ALREADY BEEN
C  SELECTED
      IF(IDV(J).NE.IDV2(ITYP))GO TO 3017
      I=I+1
C IF I IS NOT EQUAL TO IRNDM, THEN THIS IS NOT THE ONE WE WANT
      IF(I.LT.IRNDM)GO TO 3017
      IF(J.NE.KK)GO TO 412
C A LINE CANNOT INTERRACT WITH ITSELF THEREFORE A NEW LINE MUST BE CHOSEN
      IF(NCARE(IDV2(ITYP)).EQ.0)GO TO 3018
      IRNDM=1+NDV(IDV2(ITYP))*RAN(ISEED1,ISEED2)
      GO TO 382
  412 IF(NDEB.EQ.0)WRITE(6,670) J
  670 FORMAT(' LINE',I3,' SELECTED')
C THE SELECTED LINE IS ALREADY INTERRACTING WITH SOME OTHER LINE IF NLPS(
C  IS GREATER THAN 0
      IF(NLPS(J).NE.0)GO TO 3018
C LINE KK WILL SEND TO LINE J
      LSLNT(KK)=J
C GENERATE THE NUMBER OF LOOPS IN THE INTERRACTION
      NLPS(KK)=IABS(NLOOPS(ITYP))
      IF(NLOOPS(ITYP).GT.0)NLPS(KK)=POIRAN(FLOAT(NLPS(KK)),1.,
     +2.*NLPS(KK))
C THE DEFAULT VALUE FOR NLPS IS 1 LOOP
      IF(NLPS(KK).EQ.0)NLPS(KK)=1
C IF THE SELECTED LINE IS THE TRUNK, WE DO NOT WANT TO SET NLPS(J) SINCE
C  THIS WOULD PREVENT OTHER LINES FROM SENDING TO THE TRUNK AT THE SAME
C  TIME. THE TRUNK SHOULD ALWAYS BE FREE.
      IF(IDV(J).EQ.1)GO TO 3051
      NLPS(J)=NLPS(KK)
      LSLNT(J)=KK
      EVBG(J)=1E5
      EATW(J)=1E5
      GO TO 3051
 3018 IF(NDEB.EQ.0)WRITE(6,671)
  671 FORMAT('  LINE IS BUSY')
C THIS LINE IS BUSY THEREFORE CHECK IF ANOTHER LINE CAN BE SELECTED INSTE
      IF(NCARE(IDV2(ITYP)).NE.0.OR.IRNDM.GE.NDV(IDV2(ITYP)))GO TO 3026
C TRY ANOTHER LINE OF THE SAME DEVICE TYPE
      IRNDM=IRNDM+1
 3017 CONTINUE
 3026 RCTIME(KK)=0
      WTST=TIME-TIMEST(KK)
C CHECK IF LINE KK IS WILLING TO WAIT
      IF(WTST.GT.WAITT(KK))GO TO 3027
      IF(NDEB.EQ.0)WRITE(6,373)
  373 FORMAT(' WAITING')
C THE NEXT LINE IS EXECUTED IF THE SAME LINE IS TO BE CHECKED AFTER THE
C  MESSAGE IS REPEATED
```

```
      IF(NCARE(IDV2(ITYP)).NE.0)LSLNT(KK)=-IRNDM
C REPEAT MESSAGE
      MSLNTH(KK)=RESETM(KK)
C SET ARRIVAL TIME OF FIRST CHARACTER
      CALL NEXT(KK,LAMDA2(KK),-1,-1)
      GO TO 39
 3027 IF(NDEB.EQ.0)WRITE(6,372)
  372 FORMAT(' NOT WAITING')
C TRY TO INITIATE A NEW INTERRACTION WITH PERHAPS A DIFFERENT DEVICE
      ITYPE(KK)=0
C GENERATE ARRIVAL TIME OF FIRST CHARACTER
      CALL NEXT(KK,LAMDA3(KK),-1,-1)
      GO TO 39
 3051 NRB=NR(KK)
      NCRR=LC(KK)
      LSLN=LSLNT(KK)
      NRCTRT=NRCTR+NRB
      IFLG=0
C CHECK IF THERE IS ENOUGH ROOM IN THE NQIR QUEUE TO HOLD EACH BLOCK OF
C  THE MESSAGE
      NQIR=NQ(1)+NRB
      IF(NQIR.LE.MAX)GO TO 2
C MESSAGE CANNOT BE ACCEPTED SINCE THIS WOULD CAUSE THE QUEUE TO OVERFL
      IF(NDEB.EQ.0)WRITE(6,31)
   31 FORMAT('**** ABORT2 ****')
      ABORT(2,KK)=ABORT(2,KK)+1
C IF THE NUMBER OF MESSAGE BLOCKS IS GREATER THAN THE SIZE OF THE QUEUE
C  THERE IS NO POINT IN REPEATING THE MESSAGE
      IF(NRB.LE.MAX)GO TO 93
      IF(NDEB.EQ.0)WRITE(6,13)
   13 FORMAT(' **** MESSAGE TOO LONG. GENERATING NEW MESSAGE ****')
C GENERATE NEW MESSAGE LENGTH AND ARRIVAL TIME OF FIRST CHARACTER
      CALL NEXT(KK,LAMDA2(KK),0,-1)
      EVBG(KK)=EVBG(KK)+PDELF2
      EATW(KK)=EATW(KK)+PDELF2
      GO TO 39
   93 IF(NRCTRT.LE.(6*MAX).OR.(IDUP(IDV(KK)).NE.0.AND.NQH(KK).LT.MAXH(KK
     + )))GO TO 1
C ALL QUEUES ARE FULL
C  CHECK IF A MESSAGE IN THE NQT QUEUE IS WAITING TO TRANSMIT ON THIS LI
      IF(NT(KK).EQ.0)GO TO 1
      WRITE(6,81)
   81 FORMAT(' **** HOLDING MESSAGE WHILE QUEUES ARE PROCESSED WITHOUT
     + INTERRUPTION')
C
C                                   Since all messages in the NQT queue
C                                   are waiting to transmit on line KK
C                                   the no. of messages in the system
C                                   will not be reduced until the line
C                                   is free.Therefore the message will
C                                   not be repeated and the line is
C                                    released so that a message can be
C                                    removed from the system. If a file
C                                    is being received the remaining
C                                    PROCES is called to move messages
C                                    forward into the next queues.
C                                    lines are discarded.Subroutine
```

```
C
      IF(IFT(KK).GT.1)WRITE(6,6377) (IFT(KK)-1)
 6377 FORMAT(' LAST',I4,' LINES OF FILE MUST BE DISCARDED')
      KKX=KK
      ITYPX=ITYP
      LSLNX=LSLN
      IF(IDUP(IDV(KK)).NE.0)GO TO 255
C RELEASE LINE KK SO THAT A MESSAGE IN NQT CAN BEGIN TRANSMITTING ON TH
C  LINE
      EVBG(KK)=1E5
C IT IS REQUIRED THAT THIS BE THE LAST LINE (IF A FILE)
      IFT(KK)=1
      GO TO 954
C FULL DUPLEX: GO TO 942 SO THAT A MESSAGE WAITING IN THE NQH QUEUE CAN
C  BEGIN RECEIVING ON THIS LINE. BUT FIRST SET A FLAG SO THAT CONTROL C
C  BE RETURNED TO 954 AFTER THIS HAS BEEN DONE.
  255 IFLG=1
      GO TO 942
C SET PTMI TO A LARGE NUMBER SO CHARACTER ARRIVALS WILL BE IGNORED WHIL
C  PROCESSING
  954 PTMI=1E5
      CALL PROCES
C RESET KK, ITYP, AND LSLN SINCE THEIR VALUES MAY HAVE BEEN CHANGED WHE
C  OTHER SUBROUTINES WERE CALLED
      KK=KKX
      ITYP=ITYPX
      LSLN=LSLNX
      NRCTRT=NRCTR+NRB
      NQIR=NQ(1)+NRB
      IF(NQIR.LE.MAX)GO TO 2
C THERE ARE STILL TOO MANY MESSAGE BLOCKS TO FIT IN THE QUEUE. THEREFOR
C  MESSAGE WILL BE TRUNCATED
      NDIS=NQIR-MAX
      NRB=NRB-NDIS
      NRCTRT=NRCTR+NRB
      NCRR=NCB1
      IF(NRB.GT.1)NCRR=NCB2
      WRITE(6,4891) NDIS
 4891 FORMAT(' *** MESSAGE TOO LONG. LAST',I2,' BLOCKS DISCARDED ***')
      GO TO 2
C PREPARE TO REPEAT MESSAGE
    1 IF(NDEB.EQ.0)WRITE(6,6019)
 6019 FORMAT(' REPEATING MESSAGE')
      MSLNTH(KK)=RESETM(KK)
      EVART=PDELF(KK)*WIND(KK)+PDELF2
      EVBG(KK)=TIME+EVART
      EATW(KK)=EVBG(KK)+WIND(KK)
      GO TO 39
    2 NRCTR=NRCTRT
      IF(NDEB.EQ.0)WRITE(6,3845) NRCTR
 3845 FORMAT(I5,' MESSAGE BLOCKS NOW IN SYSTEM')
      TIMIN=TIME-RCTIME(KK)
      RCTIME(KK)=0
      IF(NDEB.NE.0)GO TO 396
      WRITE(6,397)
  397 FORMAT(' SENDING TO   (LINE)  NLPS  IFT   NR   LC  ITYPE    TIMES
     +     TIMIN')
```

```
      WRITE(6,398)(DVNAME(IDV(LSLN),J),J=1,12),LSLN,NLPS(KK),IFT(KK),
     +NRB,NCRR,ITYP,TIMEST(KK),TIMIN
  398 FORMAT(1X,12A1,2I6,3I5,I7,2F10.5)
  396 CONTINUE
      IF(IFLG.NE.0)GO TO 668
C DECREMENT THE NUMBER OF LINES IN THE FILE THAT HAVE NOT YET BEEN RECEIVE
      IFT(KK)=IFT(KK)-1
C IF LSLN IS NEGATIVE, THE RECEIVING LINE WILL KNOW THAT THERE ARE MORE
C  LINES TO COME. (WHEN THE MESSAGE HAS BEEN TRANSMITTED ONTO THIS LINE)
      IF(IFT(KK).GT.0)LSLN=-LSLN
  668 DO 1800 I10=1,NRB
      IF(NRB.NE.1)GO TO 5840
      NBP=0
      NCR=NCRR
      GO TO 5820
 5840 IF(I10.NE.1)GO TO 5841
      NBP=1
      NCR=NCB1
      GO TO 5820
 5841 IF(I10.EQ.NRB)GO TO 5842
      NBP=2
      NCR=NCB2
      GO TO 5820
 5842 NBP=3
      NCR=NCRR
C Q IS SET TO 0 SO THAT SUBROUTINE NQIR WILL KNOW THAT THE FIRST QUEUE
C  (NQIR) IS TO BE PUSHED
 5820 Q=0
      IF(NDEB.EQ.0)WRITE(6,348) TIME
  348 FORMAT(' PUSH NQIR TIME =',F13.8)
C PUSH A MESSAGE BLOCK INTO THE NQIR QUEUE
      CALL PUSHQ
 1800 CONTINUE
      IF(NDEB.NE.0)GO TO 941
      WRITE(6,607)
  607 FORMAT('  NQIR  NQBR  NQBT  NQIT   NQT   STAK')
      WRITE(6,9028) (NQ(I),I=1,4),NQT,STAK
 9028 FORMAT(5I6,I7)
C IF THE FLAG IS SET, THE VALUE OF IFT(KK) WILL CORRESPOND TO A DIFFERENT
C  MESSAGE. IN THIS CASE, HOWEVER, A RESPONSE HAS ALREADY BEING SENT.
C  THEREFORE EXIT.
  941 IF(IFLG.NE.0)GO TO 39
      IF(IFT(KK).EQ.0)GO TO 942
C GENERATE THE MESSAGE LENGTH OF THE NEXT LINE IN THE FILE
      CALL NEXT(KK,0.,0,-1)
      GO TO 39
  942 IF(ITYP.LT.0)GO TO 236
      NLPS(KK)=NLPS(KK)-1
      IF(NLPS(KK).EQ.0.AND.(IDUP(IDV(KK)).EQ.0.OR.
     + IDV(LSLN).NE.1))NLPS(KK)=-1
C CHECK IF ANY MESSAGES ARE IN QUEUE WAITING TO RECEIVE ON THIS LINE
  236 IF(NQH(KK).EQ.0)GO TO 242
C TAKE INFORMATION FROM QUEUE AND USE TO GENERATE ARRIVAL TIME OF FIRST
C  CHARACTER, MESSAGE LENGTH, AND FILE SIZE
      ITYPE(KK)=ITYPH(KK,1)
      LSLNT(KK)=LSLNH(KK,1)
      EVBG(KK)=TIME+EVBH(KK)
```

```
      EATW(KK)=EVBG(KK)+WIND(KK)
      CALL NEXT(KK,0.,0,0)
      IF(NDEB.EQ.0)WRITE(6,401) LSLNT(KK)
  401 FORMAT(' SENDING A RESPONSE TO LINE',I3)
      NQH(KK)=NQH(KK)-1
      IF(NQH(KK).EQ.0)GO TO 666
C SQUEEZE NQH QUEUE
      DO 244 I=1,NQH(KK)
      J=I+1
      ITYPH(KK,I)=ITYPH(KK,J)
  244 LSLNH(KK,I)=LSLNH(KK,J)
C GENERATE DELAY FOR NEXT MESSAGE IN QUEUE
      EVBH(KK)=POIRAN(LAMDA2(KK),WIND(KK),2*LAMDA2(KK))
  666 IF(IFLG.NE.0)GO TO 954
      GO TO 39
  242 IF(IDUP(IDV(KK)).EQ.0.OR.ITYP.LT.0)GO TO 239
      IF(NLPS(KK).LE.0)GO TO 240
C                                        Send next mess. without waiting
C                                        for a response to the previous o
      CALL NEXT(KK,LAMDA2(KK),0,0)
      GO TO 39
  240 IF(IDV(LSLN).EQ.1) THEN
C                                        Initiate new interaction.
        ITYPE(KK)=0
        CALL NEXT(KK,LAMDA3(KK),-1,-1)
C
      ELSE
C                                        Wait for next message.
  239   EVBG(KK)=1E5
        EATW(KK)=1E5
C
      ENDIF
C
   39 RETURN
      END
```

```
      SUBROUTINE NPINT
C
C=================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25)
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43)
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86)
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      REAL VAVDEL(25,10)
C
      DIMENSION NQH(25),MAXH(25),ITYPH(25,25),LSLNH(25,25),EVBH(25)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIH
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARME
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
      COMMON /BLK3/ NQH,MAXH,ITYPH,LSLNH,EVBH
C
      COMMON /BLK5/ VAVDEL,TVARDY
C=================================================================
C
      KI=KK-NL
C                                   TRTIME accumulates the time to proc
C                                   a character in this layer.
      TRTIME(KI)=TRTIME(KI)+WIND(KI)
C                                   Simulation time incremented by NPIH
      TIME=TIME+NPIHT(KI)
C                                   Decrement the message length.
      MSLNTH(KK)=MSLNTH(KK)-1
      IF(MSLNTH(KK).GE.1)GO TO 49
C
C                                   End of message.
C
      IF(NDEB.EQ.0)WRITE(6,70) KI
   70 FORMAT(' MESSAGE HAS BEEN TRANSMITTED ONTO LINE',I3)
```

```
      LSLN=IABS(LSLNT(KK))
      ITYP=ITYPE(KK)
      NMTR=NMTR+1
      NTRMES(KI)=NTRMES(KI)+1
      NMTRUN(LSLN,IDV(KI))=NMTRUN(LSLN,IDV(KI))+1
C THE MESSAGE DELAY IS ACCUMULATED
      AVDEL(LSLN,IDV(KI))=AVDEL(LSLN,IDV(KI))+TRTIME(KI)
C
      VAVDEL(LSLN,IDV(KI))=VAVDEL(LSLN,IDV(KI))+(TRTIME(KI)*TRTIME(KI
      IF(DEBUG.LT.1)GO TO 7979
      WRITE(6,7980)KI,TRTIME(KI),AVDEL(LSLN,IDV(KI)),
     +          VAVDEL(LSLN,IDV(KI))
 7980 FORMAT(' KI',I5,' TRTIME',F12.6,' AVDEL',F12.6,' VAVDEL',F12.6,/
 7979 EVBG(KK)=1E5
      EATW(KK)=1E5
C                                The line is released.
C                                If LSLNT<0,a file is received and more
C                                lines are expected.No action can be tai
C                                until the last line is Xmitted.
      IF(LSLNT(KK).LT.0)GO TO 3041
C                                 If ITYP is less than 0,KI is respondir
C                                 line.
      IF(ITYP.LT.0)GO TO 60
      IF(IDV(LSLN).NE.1)GO TO 61
      IF(IDUP(IDV(KI)).NE.0)GO TO 3041
      IF(NLPS(KI).GT.0)GO TO 60
      GO TO 65
C                                Decrement the number of messages sent
C                                from responding line to initiating ln.
   61 NLPS(LSLN)=NLPS(LSLN)-1
      IF(NLPS(KI).LE.0)GO TO 62
      IF(IDUP(IDV(KI)).NE.0)GO TO 3041
   60 CONTINUE
C A RESPONSE TO THE MESSAGE JUST TRANSMITTED IS PLACED IN QUEUE
      IF(NQH(KI).GT.0)GO TO 80
      EVBH(KI)=PDELF(KI)*WIND(KI)
      EVBH(KI)=POIRAN(EVBH(KI),WIND(KI),2*EVBH(KI))
   80 NQH(KI)=NQH(KI)+1
      LSLNH(KI,NQH(KI))=LSLN
      ITYPH(KI,NQH(KI))=ITYP
      IF(EVBG(KI).GE.1E4)GO TO 71
      IF(NDEB.EQ.0)WRITE(6,72) KI,NQH(KI)
   72 FORMAT(' RESPONSE IS QUEUED.  NQH(',I2,') =',I3)
      GO TO 3041
C LINE KI IS NOT ALREADY RECEIVING THEREFORE THE FIRST MESSAGE WAITING
C  IN QUEUE MAY BEGIN
   71 ITYPE(KI)=ITYPH(KI,1)
      LSLNT(KI)=LSLNH(KI,1)
C GENERATE MESSAGE LENGTH, FILE SIZE AND ARRIVAL TIME OF FIRST CHARACT
      EVBG(KI)=TIME+EVBH(KI)
      EATW(KI)=EVBG(KI)+WIND(KI)
      CALL NEXT(KI,0.,0,0)
      IF(NDEB.EQ.0)WRITE(6,73) LSLNT(KI)
   73 FORMAT(' SENDING A RESPONSE TO LINE',I3)
      NQH(KI)=NQH(KI)-1
      IF(NQH(KI).EQ.0)GO TO 3041
C
```

```
C                                   Squeeze queue.
C
      DO 67 I=1,NQH(KI)
      J=I+1
      ITYPH(KI,I)=ITYPH(KI,J)
   67 LSLNH(KI,I)=LSLNH(KI,J)
C
C                                   Set delay for first message in the Queue
C
      EVBH(KI)=POIRAN(LAMDA2(KI),WIND(KI),2*LAMDA2(KI))
      GO TO 3041
   62 IF(NLPS(LSLN).GT.0)GO TO 3041
C
C                                   Prepare to initiate next interaction
C                                   on the responding line.
C
      ITYPE(LSLN)=0
      IF(LAMDA3(LSLN).LE.0)GO TO 68
      CALL NEXT(LSLN,LAMDA3(LSLN),-1,-1)
      GO TO 65
   68 EVBG(LSLN)=1E5
      EATW(LSLN)=1E5
C
C                                     Prepare to initiate next interaction
C                                     on the initiating line.
C
   65 ITYPE(KI)=0
      NLPS(KI)=0
      IF(NDEB.EQ.0)WRITE(6,74) KI,LSLN
   74 FORMAT(' LINES',I3,' AND',I3,' ARE NOW DISCONNECTED')
      CALL NEXT(KI,LAMDA3(KI),-1,-1)
 3041 IF(NDEB.GE.1) GO TO 2033
      WRITE(6,1411) LSLN,TRTIME(KI)
 1411 FORMAT(' DELAY ON LINE',I3,' =',F18.7)
      WRITE(6,1412) KI,EVBG(KI)
 1412 FORMAT(' NEXT ARRIVAL ON LINE',I3,' AT',F18.7)
      WRITE(6,2034)
 2034 FORMAT(5X,'HISPT TIME',3X,'KK',3X,'NQT')
      WRITE(6,5006)TIME,KK,NT(KI),NQT
 5006 FORMAT(F15.4,I5,2I6)
 2033 CONTINUE
C
C                                     HISPT is called to check for other
C                                     messages waiting in NQT for Xmission
C                                     on this line.
C
      CALL HISPT
C
      IF(NDEB.GE.1)GO TO 52
      WRITE(6,7012)KK,NT(KI),NQT
 7012 FORMAT(3I6)
      GO TO 52
   49 CONTINUE
C                                     Generate arrival time of next char.
C                                     in the message.
C
      CALL NEXT(KK,-LAMDA4(KI),-1,-1)
```

```
17000        52 RETURN
17100           END
```

```
      SUBROUTINE GASTAT
C
C=====================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      DIMENSION NQH(25),MAXH(25),ITYPH(25,25),LSLNH(25,25),EVBH(25)
C
       REAL VAVDEL(25,10)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
      COMMON /BLK2/ TOTTHR,TOTDEL
C
      COMMON /BLK3/ NQH,MAXH,ITYPH,LSLNH,EVBH
C
       COMMON /BLK5/ VAVDEL,TVARDY
C
C=====================================================================
C LOCAL VARIABLES
      DOUBLE PRECISION AVD,C1,C2,DEL(15),TOTTHR,TOTDEL
      INTEGER NM(15),ABORR(50)
C
      DATA C1/'[ NUMBER'/
      DATA C2/' DELAY ]'/
      WRITE(7,1)
    1 FORMAT(' ')
C PRINT THE TIME
      ISEC=TIME
      MIN=0
    3 IF(ISEC.LT.60)GO TO 2
```

```
      ISEC=ISEC-60
      MIN=MIN+1
      GO TO 3
    2 WRITE(7,4) MIN,ISEC
    4 FORMAT(I5,' MINUTES AND',I3,' SECONDS')
      WRITE(7,1)
      AVDELY=0.
      VAVDLY=0.
C PRINT A TABLE OF MESSAGES TRANSMITTED AND THEIR AVERAGE DELAYS
C  USING NMTRUN AND AVDEL
      DO 5 I=1,NDEV
      NM(I)=0
    5 DEL(I)=0.
      ND=NDEV
      IF(ND.GT.5)ND=5
      WRITE(7,603) ((DVNAME(I,J),J=1,12),I=1,ND)
      WRITE(7,1622) ((C1,C2),I=1,ND)
  603 FORMAT('                          TOTAL      AVERAGE',4X,5(5X,12A1))
 1622 FORMAT('        LINE        TRANSMITTED   DELAY  ',4X,5(A8,1X,A8))
      DO 240 J=1,NL
      NMT=0
      AVD=0.
      VAVD=0.
      DO 1006 I=1,NDEV
      NMT=NMT+NMTRUN(J,I)
      AVD=AVD+AVDEL(J,I)
      VAVD=VAVD+VAVDEL(J,I)
      NM(I)=NM(I)+NMTRUN(J,I)
      DEL(I)=DEL(I)+AVDEL(J,I)
      IF(NMTRUN(J,I).NE.0)AVDEL(J,I)=AVDEL(J,I)/NMTRUN(J,I)
      IF(NMTRUN(J,I).NE.0)VAVDEL(J,I)=VAVDEL(J,I)/NMTRUN(J,I)
 1006 CONTINUE
      AVDELY=AVDELY+AVD
      VAVDLY=VAVDLY+VAVD
      IF(NMT.NE.0)AVD=AVD/NMT
      IF(NMT.NE.0)VAVD=VAVD/NMT
      WRITE(7,1382)J,(DVNAME(IDV(J),I),I=1,12),NMT,AVD,
     +(NMTRUN(J,I),AVDEL(J,I),I=1,ND)
  240 CONTINUE
 1382 FORMAT(I4,1X,12A1,I7,F15.8,2X,5(I6,F11.6))
      DO 6 I=1,ND
    6 IF(NM(I).NE.0)DEL(I)=DEL(I)/NM(I)
      AVDELY=AVDELY/NMTR
      VAVDLY=VAVDLY/NMTR
      VAVDLY=VAVDLY-(AVDELY*AVDELY)
      WRITE(7,1)
      WRITE(7,7)NMTR,AVDELY,(NM(I),DEL(I),I=1,ND)
    7 FORMAT(' TOTAL',I18,F15.8,2X,5(I6,F11.6))
   14 WRITE(7,1)
      IF(ND.EQ.NDEV)GO TO 8
      ND=ND+5
      NO=ND-4
      IF(ND.GT.NDEV)ND=NDEV
      WRITE(7,9) ((DVNAME(I,J),J=1,12),I=NO,ND)
      WRITE(7,10) ((C1,C2),I=NO,ND)
    9 FORMAT(31X,5(5X,12A1))
   10 FORMAT(32X,5(A8,1X,A8))
```

```
      DO 11 J=1,NL
   11 WRITE(7,12) J,(NMTRUN(J,I),AVDEL(J,I),I=NO,ND)
      WRITE(7,1)
      WRITE(7,13) (NM(I),DEL(I),I=NO,ND)
   12 FORMAT(I4,26X,5(I6,F11.6))
   13 FORMAT(' TOTAL',24X,5(I6,F11.6))
      GO TO 14
    8 ABORTS=0
      DO 83 I=1,6
   83 ABORC(I)=0
      DO 84 J=1,N2
      ABORR(J)=0
      DO 85 I=1,6
      ABORR(J)=ABORR(J)+ABORT(I,J)
   85 ABORC(I)=ABORC(I)+ABORT(I,J)
   84 ABORTS=ABORTS+ABORR(J)
      WRITE(7,3222) ABORTS
 3222 FORMAT(' ABORTS =',I6)
      IF(ABORTS.EQ.0)GO TO 8435
      WRITE(7,604)
  604 FORMAT(' LINE   OVER/UNDER-RUNS  NQIR FAIL.  NQBR FAIL.
     +  NQBT FAIL.  NQIT FAIL.  NQT FAIL.')
      DO 250 J=1,N2
      IF(ABORR(J).EQ.0)GO TO 250
      WRITE(7,5008) J,(ABORT(I,J),I=1,6)
  250 CONTINUE
 5008 FORMAT(I4,I14,5I12)
      WRITE(7,1)
      WRITE(7,826) (ABORC(I),I=1,6)
  826 FORMAT(' TOTAL',6I12)
      WRITE(7,1)
 8435 THROPT=NMTR/TIME
      WRITE(7,605)
  605 FORMAT(5X,'THROUGHPUT',5X,'AVERAGE DELAY PER MESSAGE',5X,
     +            'VARIANCE OF AV. MESS. DELAY',/)
      WRITE(7,5009) THROPT,AVDELY,VAVDLY
 5009 FORMAT(F15.8,F23.8,F23.8,/)
C ACCUMULATE THROUGHPUT AND DELAY FOR EACH SIMULATION
      TOTDEL=TOTDEL+AVDELY
      TVARDY=TVARDY+VAVDLY
C
      TOTTHR=TOTTHR+THROPT
      WRITE(7,1)
C PRINT THE NUMBER OF MESSAGES ARRIVED AND TRANSMITTED
      WRITE(7,606)
  606 FORMAT(5X,'LINE',3X,'NARMES',3X,'NARTOT',3X,'NTRMES',3X,
     $'K',3X,'NT',3X,'NQH')
      DO 608 J=1,NL
      WRITE(7,5010) J,NARMES(J),NARTOT(J),NTRMES(J),K(J),NT(J),NQH(J)
  608 CONTINUE
 5010 FORMAT(4I9,I4,I5,I6)
      WRITE(7,1)
C PRINT THE NUMBER OF MESSAGES LEFT IN EACH QUEUE AT THE END OF THE
C  SIMULATION
      WRITE(7,607)
  607 FORMAT('  NQIR  NQBR  NQBT  NQIT   NQT   STACK')
      WRITE(7,9028) (NQ(I),I=1,4),NQT,STAK
```

```
7200      9028 FORMAT(5I6,I8)
7300           WRITE(7,1)
7400           RETURN
7500           END
```

```
      SUBROUTINE ERLST(N2,KK,EVBG)
      DOUBLE PRECISION EVBG(50),EVE
C
C                                   Finds the line number(KK) on which
C                                   the next earliest event occurs.
C                                   finds the earliest in EVBG.
C
      EVE=EVBG(1)
      KK=1
      DO 500 J=2,N2
      IF(EVE.LE.EVBG(J))GO TO 500
      EVE=EVBG(J)
      KK=J
  500 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PROCES
C
C=================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
C=================================================================
C LOCAL VARIABLES
      REAL QNAME(5)
C
      DATA QNAME/'NQBR','NQBT','NQIT','NQT ','STAK'/
C
C                                   Check for message in the stack.
C
      IF(STAK.EQ.0)GO TO 13
      STAK=0
C                                   Fetch from stack.
      KK=KKS
      TIMIN=TIMES
      NCR=NCRS
      NBP=NBPS
      ITYP=ITYPS
      LSLN=LSLNS
C                                   Q is set to the value it had when
C                                   it was interrupted.
      Q=IQ
C                                   IREP=1 so that cycle can be repeated.
      IREP=-1
      KI=KK
```

```
5800          IF(KI.GT.NL)KI=KI-NL
5900          GO TO 6
6000    C                                  If queue empty go to next queue.
6100       11 IF(NQ(Q).GT.0)GO TO 12
6200       15 Q=Q+1
6300          IF(Q.LE.4)GO TO 11
6400    C                                   End of cycle. Next only if IREP=-1.
6500       16 IF(IREP.GE.0)GO TO 10
6600    C                                   Begin cycle.
6700       13 Q=1
6800          IREP=0
6900          GO TO 11
7000       12 IF(Q.EQ.4)GO TO 14
7100          IF(NQ(Q+1).LT.MAX)GO TO 5
7200    C                                   Next queue is full.
7300    C
7400          ABORT((Q+2),KKQ(Q,1))=ABORT((Q+2),KKQ(Q,1))+1
7500          IF(NDEB.EQ.0)WRITE(6,72) (Q+2)
7600       72 FORMAT(' **** ABORT',I2,' ****')
7700    C ADJUST IREP:  0 => 1, 1 => 1, -1 => -1
7800          IREP=1+IREP-IABS(IREP)
7900    C GO TO NEXT QUEUE
8000          GO TO 15
8100       14 IF(NQT.LT.(2*MAX))GO TO 5
8200    C                                  Queue is full.
8300          ABORT(6,KKQ(4,1))=ABORT(6,KKQ(4,1))+1
8400          IF(NDEB.EQ.0)WRITE(6,72) 6
8500    C                                  Call HISPT to check if one of the
8600    C                                  messages in NQT can be transmitted.
8700    C                                  Set KK=0 so that higher priority is
8800    C                                  given to messages which arrived first.
8900    C
9000          KK=0
9100    C
9200          CALL HISPT
9300    C
9400          IF(NQT.EQ.(2*MAX))GO TO 16
9500    C                                 Else message block can be processed.
9600    C ADJUST IREP: 0 => 0, 1 => -1, -1 => -1
9700        5 IREP=-IABS(IREP)
9800    C FETCH FROM QUEUE
9900          CALL PULLQ
10000         KI=KK
10100         IF(KI.GT.NL)KI=KI-NL
10200   C                                 Calculate time to process the message blk.
10300   C
10400         IPROT=PROT(KI)
10500         PROC=0
10600         IF(Q.EQ.1)PROC=PIHR(KI)
10700         IF(Q.EQ.2.AND.IPROT.GT.0)PROC=BGR(IPROT,KI)
10800         IF(Q.EQ.3.AND.IPROT.GT.0)PROC=BGT(IPROT,KI)
10900         IF(Q.EQ.4)PROC=PIHT(KI)
11000         TLEFT=PROC*NCR
11100       6 IF(NDEB.EQ.0)WRITE(6,19) KK,NCR,TLEFT,PTMI
11200      19 FORMAT(' LINE=',I2,'     NCR=',I2,'     PROCES TIME=',F10.7,
11300        +'     ALLOWED TIME=',F14.7)
11400   C
```

```
C                                   Check if there is enough time to process
C                                   a message block before a new char. arrives
C
      IF(TLEFT.LE.PTMI)GO TO 7
C
C                                   Interrupt processing & store message block
C                                   in the stack.
C
      IF(NDEB.EQ.0) WRITE(6,18) QNAME(5),TIME
   18 FORMAT(' PUSH ',A4,' TIME =',F13.8)
      STAK=1
      KKS=KK
      TIMES=TIMIN
      NCRS=NCR
      NBPS=NBP
      ITYPS=ITYP
      LSLNS=LSLN
C
C                                     Store value of Q to enable processing
C                                     from the same point.
      IQ=Q
C
C                                     TLEFT is the processing time still
C                                     required on this block.
C
      TLEFT=TLEFT-PTMI
      IF(NDEB.NE.0)GO TO 10
      WRITE(6,607)
      WRITE(6,9028) (NQ(I),I=1,4),NQT,STAK
      GO TO 10
C ADJUST THE TIME AND THE TIME BEFORE THE NEXT CHARACTER ARRIVAL
    7 TIME=TIME+TLEFT
      PTMI=PTMI-TLEFT
C IF RECEIVING IN THE BG LAYER, SWITCH TO THE NEW LINE
      IF(Q.NE.2)GO TO 9
      JJ=KI
      KI=LSLN
      KK=IABS(KI)+NL
      LSLN=JJ
      IF(KI.GT.0)GO TO 30
      KI=-KI
      LSLN=-LSLN
   30 ITYP=-ITYP
      IF(IDV(JJ).EQ.1.OR.IDV(KI).EQ.1)K(JJ)=K(JJ)+1
    9 IF(NDEB.EQ.0)WRITE(6,18) QNAME(Q),TIME
      IF(Q.EQ.4)GO TO 17
C
      CALL PUSHQ
C
      IF(NDEB.NE.0)GO TO 11
      WRITE(6,607)
      WRITE(6,9028) (NQ(I),I=1,4),NQT,STAK
      GO TO 11
   17 IF(NBP.EQ.0.OR.NBP.EQ.3)NT(KI)=NT(KI)+1
      NQT=NQT+1
C                                     Push NQT queue.
      KKT(NQT)=KK
```

```
      TIMET(NQT)=TIMIN
      NCRT(NQT)=NCR
      NBPT(NQT)=NBP
      ITYPT(NQT)=ITYP
      LSLNQT(NQT)=LSLN
      IF(NDEB.NE.0)GO TO 20
      WRITE(6,607)
      WRITE(6,9028) (NQ(I),I=1,4),NQT,STAK
  607 FORMAT('  NQIR  NQBR  NQBT  NQIT   NQT   STACK')
 9028 FORMAT(5I6,I8)
      WRITE(6,23)
   23 FORMAT(' NQT QUEUE:')
      WRITE(6,24)
   24 FORMAT('   KK  NCR  NBP',4X,'TIMIN  ITYP  LSLN')
      DO 25 J=1,NQT
   25 WRITE(6,26) KKT(J),NCRT(J),NBPT(J),TIMET(J),ITYPT(J),LSLNQT(J)
   26 FORMAT(3I5,F9.4,2I6)
C
   20 CALL HISPT
C
      GO TO 11
   10 RETURN
      END
```

```
      SUBROUTINE NEXT(JJ,EVMN,MSMN,IFMN)
C
C=================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
C=================================================================
C
C  JJ   : LINE NUMBER ON WHICH THE EVENT WILL OCCUR
C
C  EVMN : <0  NEXT ARRIVAL TIME GENERATED SYNCHRONOUSLY USING AN EXPONENTIAL
C              DISTRIBUTION. -EVMN IS ASSUMED TO BE THE UTILISATION AND IS
C              USED TO DETERMINE IF A DATA OR SYNC CHARACTER WILL BE GENERATED
C         =0  NEXT ARRIVAL TIME UNCHANGED
C         >0  TIME BEFORE NEXT ARRIVAL GENERATED USING EVMN AS A MEAN
C
C  MSMN : <0  MESSAGE LENGTH UNCHANGED
C         =0  MESSAGE LENGTH GENERATED USING THE INPUT DATA AS SPECIFIED
C              BY ITYPE(JJ)
C         >0  MESSAGE LENGTH GENERATED USING MSMN AS A MEAN
C
C  IFMN : <0  NUMBER OF LINES (FOR FILES) UNCHANGED
C         =0  NUMBER OF LINES GENERATED USING THE INPUT DATA AS SPECIFIED
C              BY ITYPE(JJ)
C         >0  NUMBER OF LINES IN FILE GENERATED USING IFMN AS A MEAN
C
      KI=JJ
      IF(KI.GT.NL)KI=JJ-NL
```

```
C GENERATE CHARACTER ARRIVAL TIME
C A COMPARISON TOLERANCE OF 1E-5 IS ALLOWED WHEN CHECKING FOR ZERO
      IF(ABS(EVMN).LT.1E-5)GO TO 1
      IF(EVMN.LT.0)GO TO 2
      EVART=POIRAN(EVMN,WIND(KI),2*EVMN)
      EVBG(JJ)=TIME+EVART
      GO TO 3
C GENERATE I, THE NUMBER OF SYNC CHARACTERS BEFORE THE NEXT ARRIVAL
    2 I=POIRAN(-1/EVMN,0.,-2/EVMN)
C SET THE ARRIVAL TIME OF THE NEXT DATA CHARACTER
      EVBG(JJ)=EVBG(JJ)+(I+1)*WIND(KI)
    3 EATW(JJ)=EVBG(JJ)+WIND(KI)
C GENERATE MESSAGE LENGTH
    1 IF(MSMN.LT.0)GO TO 4
      IF(MSMN.EQ.0)GO TO 5
      MSLNTH(JJ)=POIRAN(FLOAT(MSMN),1.,FLOAT(MSMAX(JJ)))
      GO TO 6
C ITYPE(JJ) WILL BE NEGATIVE IF THIS IS THE RESPONDING LINE AND NOT THE
C  INITIATOR
    5 IF(ITYPE(JJ).LT.0)GO TO 7
      MSLNTH(JJ)=-MSLNT1(ITYPE(JJ))
      GO TO 8
    7 MSLNTH(JJ)=-MSLNT2(IABS(ITYPE(JJ)))
    8 IF(MSLNTH(JJ).LT.0)MSLNTH(JJ)=POIRAN(-1.*MSLNTH(JJ),1.,
     + FLOAT(MSMAX(KI)))
C DETERMINE HOW MANY MESSAGE BLOCKS WILL BE REQUIRED (NR(JJ) )
C AND HOW MANY CHARACTERS WILL BE IN THE LAST BLOCK (LC(JJ) )
    6 NCB0=NCB1
      NR(JJ)=1
      LC(JJ)=MSLNTH(JJ)
    9 IF(LC(JJ).LE.NCB0)GO TO 10
      NR(JJ)=NR(JJ)+1
      LC(JJ)=LC(JJ)-NCB0
      NCB0=NCB2
      GO TO 9
C ADD A HEADER TO THE MESSAGE LENGTH (FOR EACH BLOCK) IF LINE IS THE TRUNK
   10 IF(IDV(JJ).EQ.1)MSLNTH(JJ)=MSLNTH(JJ)+NR(JJ)*NCTR
C STORE THE MESSAGE LENGTH IN CASE THE MESSAGE WILL NEED TO BE REPEATED
      RESETM(JJ)=MSLNTH(JJ)
C GENERATE HOW MANY LINES WILL BE IN THE FILE (IF MORE THAN ONE)
    4 IF(IFMN.LT.0)GO TO 11
      IF(IFMN.EQ.0)GO TO 12
      IFT(JJ)=POIRAN(FLOAT(IFMN),1.,2.*IFMN)
      GO TO 11
C ITYPE(JJ) WILL BE NEGATIVE IF THIS IS THE RESPONDING LINE AND NOT THE
C  INITIATOR
   12 IF(ITYPE(JJ).LT.0)GO TO 13
      IFT(JJ)=-IFT1(ITYPE(JJ))
      GO TO 14
   13 IFT(JJ)=-IFT2(IABS(ITYPE(JJ)))
   14 IF(IFT(JJ).LT.0)IFT(JJ)=POIRAN(-1.*IFT(JJ),1.,-2.*IFT(JJ))
C THE DEFAULT FILE SIZE IS 1 LINE
      IF(IFT(JJ).EQ.0)IFT(JJ)=1
   11 RETURN
      END
```

```
      SUBROUTINE HISPT
C
C=====================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      DIMENSION NQH(25),MAXH(25),ITYPH(25,25),LSLNH(25,25),EVBH(25)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
      COMMON /BLK3/ NQH,MAXH,ITYPH,LSLNH,EVBH
C
C=====================================================================
C                                Local variables.
C
      DIMENSION LFREE(25)
C
      IF(NQT.EQ.0)GO TO 2171
C
C                                Message is Xmitted only if line is free.
C                                Line=free if 1) H.duplex & not Xmitting,
C                                or receiving,2) f.duplex & not xmitting
C                                with room in NQH queue.
C
      DO 17 J=1,NL
      LFREE(J)=0
      JI=J+NL
C
C                                If line not free,set LFREE(J)=1.
C
      IF(EVBG(JI).LT.1E4.OR.NQH(J).EQ.MAXH(J))LFREE(J)=1
```

```
5400          IF(IDUP(IDV(J)).EQ.0.AND.EVBG(J).LT.1E4)LFREE(J)=1
5500       17 CONTINUE
5600          IF(KK.EQ.0)GO TO 2181
5700          KI=KK-NL
5710    C
5720    C                                A mess. waiting on ln KK has highest
5730    C                                priority. Check such messages & if line
5740    C                                KK is free. if not, priority is given to
5750    C                                the first message in the queue.
5760    C
6200          IF(NT(KI).EQ.0.OR.LFREE(KI).NE.0)GO TO 2181
6300          DO 2180 J=1,NQT
6400          JY=J
6450    C
6451    C                              If line no. matches KK,check for all blocks
6452    C
6700          IF((KKT(J).EQ.KK).AND.(NBPT(J).EQ.0.OR.NBPT(J).EQ.3))GO TO 2182
6800     2180 CONTINUE
6900    C TRANSMIT THE FIRST MESSAGE IN THE QUEUE REQUIRING A LINE THAT IS FREE
7000     2181 DO 18 J=1,NQT
7100          JY=J
7200          KI=KKT(J)-NL
7300    C IF THE LINE IS FREE, CHECK IF ALL BLOCKS IN THE MESSAGE ARE WAITING IN
7400    C  THE QUEUE
7500          IF((LFREE(KI).EQ.0).AND.(NBPT(J).EQ.0.OR.NBPT(J).EQ.3))GO TO 2182
7600       18 CONTINUE
7700          IF(NDEB.GT.0)GO TO 2171
7800          WRITE(6,4008)
7900     4008 FORMAT(' NO NEW BLOCK TO TRANSMIT HISPT')
8000          GO TO 2171
8100    C                              Fetch each block from NQT.
8200     2182 NBLOCS=1
8300          JJ=KKT(JY)
8400          TIMIN=TIMET(JY)
8500          ITYPE(JJ)=ITYPT(JY)
8600          LSLNT(JJ)=LSLNQT(JY)
8610    C                              Set arr. time of 1st char. to be Xmitted
8620    C                              & adjust time of next char. arrival(PTMI)
8630    C
8900          EVART=0
9000          IF(PTMI.GT.EVART.AND.PTMI.LT.1E4)PTMI=EVART
9100          EVBG(JJ)=TIME+EVART
9200          EATW(JJ)=EVBG(JJ)+WIND(KI)
9300          NCR=0
9400     5453 NBP=NBPT(JY)
9500          NCR=NCR+NCRT(JY)
9600          IF(NBP.LE.1)GO TO 6393
9700          JY=JY-1
9800          NBLOCS=NBLOCS+1
9900          GO TO 5453
10000    6393 IF(NDEB-1) 4024,4025,4025
10100    4024 WRITE(6,4006)
10200    4006 FORMAT(5X,'JJ',5X,'NCR')
10300         WRITE(6,5016)JJ,NCR
10400    5016 FORMAT(2I7)
10500    4025 NT(KI)=NT(KI)-1
10600         NQT=NQT-NBLOCS
```

```
10700          IF(JY.GT.NQT)GO TO 1515
10800    C SQUEEZE QUEUE
10900          DO 2190 JL=JY,NQT
11000          JLL=JL+NBLOCS
11100          TIMET(JL)=TIMET(JLL)
11200          NCRT(JL)=NCRT(JLL)
11300          NBPT(JL)=NBPT(JLL)
11400          KKT(JL)=KKT(JLL)
11500          ITYPT(JL)=ITYPT(JLL)
11600          LSLNQT(JL)=LSLNQT(JLL)
11700     2190 CONTINUE
11800     1515 MSLNTH(JJ)=NCR
11810    C
11820    C                                If xmitting on trunk,add header to each
11830    C                                block of the message.
11840    C
12000          IF(IDV(KI).NE.1)GO TO 14
12100          MSLNTH(JJ)=MSLNTH(JJ)+NCTR*NBLOCS
12200       14 RESETM(JJ)=MSLNTH(JJ)
12300          NRCTR=NRCTR-NBLOCS
12400          IF(NDEB-1) 2172,2173,2173
12500     2172 WRITE(6,5017) JJ,MSLNTH(JJ),TIME
12600     5017 FORMAT(I7,I11,F9.3)
12700          WRITE(6,7428)NRCTR
12800     7428 FORMAT(I5,' MESSAGE BLOCKS NOW IN SYSTEM')
12900     2173 TRTIME(KI)=TIME-TIMIN
13000     2171 KK=JJ
13100          RETURN
13200          END
```

```
      SUBROUTINE PUSHQ
C
C=====================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEP,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
C=====================================================================
C
C  SET KI EQUAL TO KK IF RECEIVING; (KK-NL) IF TRANSMITTING
      KI=KK-(KK/(NL+1))*NL
C Q IS THE NUMBER OF THE QUEUE TO BE PUSHED
      Q=Q+1
C SCAN THE QUEUE TO DETERMINE WHERE TO INSERT NEW MESSAGE BLOCK.
C  BLOCK SHOULD BE PLACED BEFORE THE FIRST MESSAGE BLOCK THAT IS ON A
C  LINE OF LOWER PRIORITY. IT CANNOT BE INSERTED BETWEEN BLOCKS BELONGING
C  TO THE SAME MESSAGE.
      DO 1 I=1,NQ(Q)
      JI=KKQ(Q,I)
      IF(JI.GT.NL)JI=JI-NL
      IF(JPR(KI).LT.JPR(JI).AND.NBPQ(Q,I).LE.1)GO TO 2
    1 CONTINUE
      I=NQ(Q)+1
    2 NQ(Q)=NQ(Q)+1
      NC2=NQ(Q)
    4 IF(NC2.EQ.I)GO TO 3
      NC1=NC2-1
C ALL FOLLOWING BLOCKS IN THE QUEUE ARE SHIFTED UP ONE POSITION
      KKQ(Q,NC2)=KKQ(Q,NC1)
      TIMEQ(Q,NC2)=TIMEQ(Q,NC1)
      NCRQ(Q,NC2)=NCRQ(Q,NC1)
      NBPQ(Q,NC2)=NBPQ(Q,NC1)
      ITYPQ(Q,NC2)=ITYPQ(Q,NC1)
      LSLNQ(Q,N[illegible]  LSLNQ(Q,NC1)
      NC2=NC2-1
```

```
      GO TO 4
C INSERT NEW MESSAGE BLOCK
    3 KKQ(Q,I)=KK
      TIMEQ(Q,I)=TIMIN
      NCRQ(Q,I)=NCR
      NBPQ(Q,I)=NBP
      ITYPQ(Q,I)=ITYP
      LSLNQ(Q,I)=LSLN
C RESET Q TO ITS INITIAL VALUE
      Q=Q-1
      RETURN
      END
```

```
      SUBROUTINE PULLQ
C
C=================================================================
      DOUBLE PRECISION EVBG(50),EATW(50),AVDEL(25,10),RCTIME(25),
     1  TRTIME(25),TIMEQ(4,43),TIMET(86)
C
      DOUBLE PRECISION TIME,PTMI,TLEFT,TIMIN,TIMES
C
      REAL LAMDA1(25),LAMDA2(25),LAMDA3(25),LAMDA4(25),WIND(25),
     1  PDELF(25),NPIHR(25),NPIHT(25),PIHR(25),PIHT(25),BGR(4,25),
     2  BGT(4,25),PROB(25),WAIT(25),TIMEST(25),WAITT(25)
C
      INTEGER STAK,Q,ABORTS
C
      INTEGER NCARE(10),IDUP(10),IDV(25),JPR(25),MSMAX(25),PROT(25),
     1  IDV1(25),IDV2(25),NLOOPS(25),MSLNT1(25),IFT1(25),MSLNT2(25),
     2  IFT2(25),NDV(10),NT(25),NARMES(25),NTRMES(25),NARTOT(25),K(25),
     3  NMTRUN(25,10),ABORT(6,50),NQ(4),NLPS(25),LSLNT(50),ITYPE(50),
     4  MSLNTH(50),IFT(25),RESETM(50),NR(25),LC(25),ABORC(6),KKQ(4,43),
     5  NCRQ(4,43),NBPQ(4,43),ITYPQ(4,43),LSLNQ(4,43),KKT(86),NCRT(86),
     6  NBPT(86),ITYPT(86),LSLNQT(86)
C
      LOGICAL*1 DVNAME(10,12)
C
      COMMON /BLK1/ NDEV,NL,MAX,ISEED1,ISEED2,PDELF2,NCARE,IDUP,DVNAME,
     1  IDV,LAMDA1,LAMDA2,LAMDA3,LAMDA4,WIND,JPR,PDELF,MSMAX,PROT,NPIHR,
     2  NPIHT,PIHR,PIHT,BGR,BGT,IDV1,IDV2,PROB,NLOOPS,MSLNT1,IFT1,
     3  MSLNT2,IFT2,WAIT,NDV,NCB1,NCB2,NCTR,N2,TIME,EVBG,EATW,NT,NARMES,
     4  NTRMES,NARTOT,K,NMTRUN,AVDEL,ABORT,STAK,NQ,NQT,NMTR,NRCTR,
     5  TIMEST,NLPS,LSLNT,ITYPE,MSLNTH,RCTIME,IFT,RESETM,NDEB,PTMI,
     6  TLEFT,NR,LC,NSW,WAITT,TRTIME,Q,ABORC,ABORTS,KK,TIMIN,NCR,NBP,
     7  ITYP,LSLN,KKQ,TIMEQ,NCRQ,NBPQ,ITYPQ,LSLNQ,KKT,TIMET,NCRT,NBPT,
     8  ITYPT,LSLNQT,KKS,TIMES,NCRS,NBPS,ITYPS,LSLNS
C
C=================================================================
C
C FETCH THE FIRST MESSAGE BLOCK IN THE QUEUE DETERMINED BY THE VARIABLE Q
      KK=KKQ(Q,1)
      TIMIN=TIMEQ(Q,1)
      NCR=NCRQ(Q,1)
      NBP=NBPQ(Q,1)
      ITYP=ITYPQ(Q,1)
      LSLN=LSLNQ(Q,1)
      NQ(Q)=NQ(Q)-1
C SQUEEZE QUEUE
      DO 1 I=1,NQ(Q)
      IL=I+1
      KKQ(Q,I)=KKQ(Q,IL)
      TIMEQ(Q,I)=TIMEQ(Q,IL)
      NCRQ(Q,I)=NCRQ(Q,IL)
      NBPQ(Q,I)=NBPQ(Q,IL)
      ITYPQ(Q,I)=ITYPQ(Q,IL)
      LSLNQ(Q,I)=LSLNQ(Q,IL)
    1 CONTINUE
      RETURN
      END
```

```
      REAL FUNCTION POIRAN(MEAN,MIN,MAX)
C THIS FUNCTION RETURNS A POISSON DISTRIBUTION BETWEEN MIN AND MAX
C THE REQUIRED INPUT IS THE MEAN (MEAN) THE
C MAXIMUM AND THE MINNIMUM RETURNABLE VALUES.
      COMMON /SEEDS/ ISEED1,ISEED2
      REAL MEAN,MIN,MAX
C REQUIRED FUNCTIONS
C  RAN-GENERATES A PSEUDO RANDOM NUMBER BETWEEN 0 AND 1 GIVEN TWO SEEDS
C REPEAT
    1 POIRAN=-MEAN*ALOG(RAN(ISEED1,ISEED2))
      IF(POIRAN.GT.MAX .OR. POIRAN.LT.MIN) GO TO 1
C UNTIL POIRAN <= MAX   AND   POIRAN => MIN
      RETURN
      END
```

APPENDIX B

A TYPICAL OUTPUT FILE LISTING

PROBABILITIES AFTER SCALING:
0.180000
0.120000
0.120000
0.120000
0.120000
0.170000
0.170000
0.500000
0.500000

SIMULATION NO. 1

1 MINUTES AND 6 SECONDS

	LINE	TOTAL TRANSMITTED	AVERAGE DELAY	TRUNK NUMBER	TRUNK DELAY	TERMINAL NUMBER	TERMINAL DELAY
1	TERMINAL	77	0.58852000	0	0.000000	10	0.450389
2	TERMINAL	132	0.61390152	0	0.000000	103	0.631711
3	TERMINAL	52	0.73758241	0	0.000000	38	0.826870
4	TERMINAL	112	0.61384886	1	0.011689	110	0.624747
5	TERMINAL	44	0.69849311	0	0.000000	12	0.407807
6	TERMINAL	124	0.62154471	0	0.000000	44	0.595128
7	TERMINAL	0	0.00000000	0	0.000000	0	0.000000
8	TERMINAL	103	0.63279556	2	0.017217	101	0.644985
9	TERMINAL	166	0.68827666	0	0.000000	166	0.688277
10	TERMINAL	7	0.09817486	2	0.028824	5	0.125915
11	PRINTER	1	0.01701400	0	0.000000	1	0.017014
12	PRINTER	2	0.01701400	0	0.000000	2	0.017014
13	DATA BASE	18	0.35880984	0	0.000000	18	0.358810
14	OCR	58	0.21371427	0	0.000000	58	0.213714
15	TRUNK	4	0.16855000	0	0.000000	4	0.168550
TOTAL		900	0.60106665	5	0.020754	672	0.595832

PRINTER NUMBER	PRINTER DELAY	DATA BASE NUMBER	DATA BASE DELAY	OCR NUMBER	OCR DELAY
67	0.609137	0	0.000000	0	0.000000
29	0.550649	0	0.000000	0	0.000000
14	0.495230	0	0.000000	0	0.000000
0	0.000000	1	0.017167	0	0.000000
0	0.000000	32	0.807500	0	0.000000
0	0.000000	80	0.636074	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
0	0.000000	0	0.000000	0	0.000000
110	0.579220	113	0.679142	0	0.000000

ABORTS = 3047

```
   5   15   900.000  900    1    5 4567 8934    9999.999 0.000100000   1
0 0 TRUNK
0 1 TERMINAL
0 1 PRINTER
1 1 DATA BASE
0 1 OCR
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
2  0.10000  0.10000  1.00000  0.80000 0.008332 2  30.0  80  25 1
3  0.00000  0.10000  0.00000  0.80000 0.008332 2  30.0  80  25 3
3  0.00000  0.10000  0.00000  0.80000 0.008332 2  30.0  80  25 3
4  0.00000  0.10000  0.00000  0.80000 0.008333 2  30.0  80  25 4
5  0.10000  0.10000 25.00000  0.80000 0.033333 2  30.0  80  25 4
1  0.00000  0.10000  0.00000  1.00000 0.000833 1  25.0 256  25 1
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000140 0.000100 0.000008 0.000008 0.000008 0.000008 0.000010 0.000010
0.000040 0.000040 0.000008 0.000008 0.000008 0.000008 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
0.000011 0.000011 0.000012 0.000012
2 1 0.18000  -1  30   0  30   0    0.000
2 3 0.12000   1  40 100  -2   0    2.000
2 4 0.12000   1  60 100  -2   0   15.000
2 4 0.12000   1  -2   0  60 100   15.000

2 2 0.12000  -1  10   0  10   0   10.000
2 2 0.17000  -1  50  50  -2   0   40.000
2 2 0.17000  -1  -2   0  50  50   40.000
5 2 0.50000  -1  30  80  -2   0  100.000
5 4 0.50000  -1  30  80  -2   0  100.000
```

LINE	OVER/UNDER-RUNS	NQIR FAIL.	NQBR FAIL.	NQBT FAIL.	NQIT FAIL.	NQT FAIL
16	0	0	0	0	0	339
17	0	0	0	0	0	30
18	0	0	0	0	0	274
19	0	0	0	0	0	162
20	0	0	0	0	0	226
21	0	0	0	0	0	105
22	0	0	0	0	0	165
23	0	0	0	0	0	46
25	0	0	0	0	19	445
26	0	0	0	0	0	639
27	0	0	0	0	0	119
28	0	0	0	0	70	408
TOTAL	0	0	0	0	89	2958

THROUGHPUT	AVERAGE DELAY PER MESSAGE	VARIANCE OF AV. MESS. DELAY
13.52154541	0.60106665	0.05893308

LINE	NARMES	NARTOT	NTRMES	K	NT	NQH
1	77	99	104	0	0	0
2	132	171	41	0	2	0
3	52	67	111	0	0	0
4	112	144	55	1	2	0
5	56	87	92	0	0	0
6	125	167	44	0	0	0
7	0	0	58	0	0	0
8	112	148	44	2	0	0
9	169	218	4	0	0	0
10	51	52	119	2	0	0
11	1	1	67	0	0	0
12	2	2	43	0	0	0
13	21	27	113	0	0	0
14	58	66	0	0	0	0
15	5	7	5	7	0	0

NQIR	NQBR	NQBT	NQIT	NQT	STACK
0	0	0	0	4	0

MAX 5 ABORTS 3047 T.DELAY 0.601067 T.THRUPUT 13.521545

VAR. OF MESSAGE DELAY 0.0589331

APPENDIX C

A Typical VAX 11/750 LOG FILE Listing

```
ASSIGN WPCC.DAT;1 FOR004
@CSW
$ assign NEW.dat for007
 ASSIGN DUMMY SYS$PRINT
$ R WPCC
ENTER THE NAME OF THE MAIN OUTPUT FILE.

ENTER THE NAME OF THE PLOT FILE.

ENTER THE INPUT DATA FILE NAME.

  2-MAY-82

    SIMULATION NO.     1
TIME =       0.      NMTR =    0
TIME =      10.      NMTR =  103
TIME =      20.      NMTR =  227
TIME =      30.      NMTR =  385
TIME =      40.      NMTR =  516
TIME =      50.      NMTR =  681
TIME =      60.      NMTR =  812
FORTRAN STOP
ASSIGN WPCC.DAT;2 FOR004
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
@CSW
$ assign NEW.dat for007
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
 ASSIGN DUMMY SYS$PRINT
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
$ R WPCC
ENTER THE NAME OF THE MAIN OUTPUT FILE.

ENTER THE NAME OF THE PLOT FILE.

ENTER THE INPUT DATA FILE NAME.

  2-MAY-82

    SIMULATION NO.     1
TIME =       0.      NMTR =    0
TIME =      10.      NMTR =  102
TIME =      20.      NMTR =  223
TIME =      30.      NMTR =  353
TIME =      40.      NMTR =  486
TIME =      50.      NMTR =  651
TIME =      60.      NMTR =  778
FORTRAN STOP
ASSIGN WPCC.DAT;3 FOR004
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
@CSW
$ assign NEW.dat for007
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
 ASSIGN DUMMY SYS$PRINT
  PREVIOUS LOGICAL NAME ASSIGNMENT REPLACED
$ R WPCC
ENTER THE NAME OF THE MAIN OUTPUT FILE.
```

```
ENTER THE NAME OF THE PLOT FILE.

ENTER THE INPUT DATA FILE NAME.

  2-MAY-82

    SIMULATION NO.    1
TIME =      0.      NMTR =    0
TIME =     10.      NMTR =   98
TIME =     20.      NMTR =  203
TIME =     30.      NMTR =  306
TIME =     40.      NMTR =  415
TIME =     50.      NMTR =  549
TIME =     60.      NMTR =  665
TIME =     70.      NMTR =  775
TIME =     80.      NMTR =  888
FORTRAN STOP
  BHATIA       Job terminated at  2-MAY-1982 17:13:09.59

  Accounting information:
  Buffered I/O count:            63      Peak working set size:   180
  Direct I/O count:              35      Peak virtual size:       259
  Page faults:                 3242      Mounted volumes:           0
  Elapsed CPU time:     0 00:05:26.41    Elapsed time:     0 00:09:28.75
```

CACC / CCAC
80170

MOUFTAH, H.T.
--Communications controller for word processors: final report.

P
91
C655
M68
1982

DATE DUE
DATE DE RETOUR

DEC 15	1983		

LOWE-MARTIN No. 1137