222

A USER TRANSPARENT DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

by

Richard Dale Housh

B.S., Kansas State University, Manhattan, Kansas, 1976

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

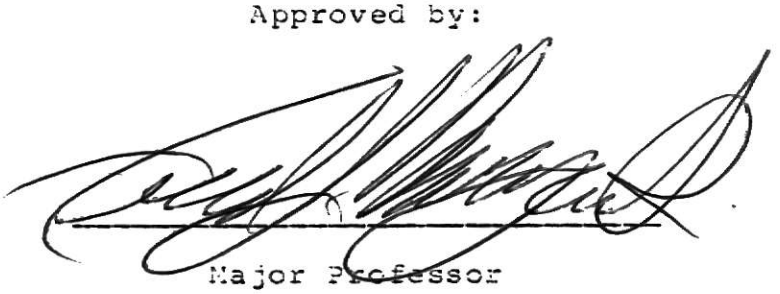
Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1978

Approved by:



Major Professor

TABLE OF CONTENTS

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

FIGURES

ACKNOWLEDGEMENTS

I would like to thank Dr. Fred Maryanski for all the help and encouragement he gave me during the course of this project. Several others also deserve my thanks - David Schmidt for his help with the interface routines and COBOL in general, Alan Skidmore for his help in 'learning the ropes' on the Interdata, and finially Jim Ratliff for his help with the operation of the message system.

I. INTRODUCTION

Data base management has received a great deal of emphasis in the last few years as business and government alike became aware of the role a data base plays in computer applications. Several data base management systems (IMS, System 2000, IDMS, TOTAL, ADABAS, etc. (1)) have been implemented to fulfill the needs of various organizations.

In spite of their power, existing data base management systems have some drawbacks.. One problem is that data base management manipulations require large amounts of computer resources, which can disrupt the performance of the system as a whole. The problem of data base security has received some attention in most implementations, but the application program, data base management system, and the data base itself still reside on the same machine. With a little ingenuity, it may be possible to bypass a data base management system's security provisions or access the data base without the use of the data base management system at all. An organization would also like to protect the physical media upon which the data base is kept. Some organizations keep multiple copies of a data base on machines which may be separated by great distances, but the major problem with this method is keeping all copies of the data base correctly updated. These are only a few of the problems facing current data base management systems.

The purpose of this report is to provide a description of a prototype distributed data base management system which will alleviate some of the problems described above.

Chapter two gives basic definitions used in this report and compares other work in distributed data base management systems to the methods used in this report. Chapter three discusses the hardware used in the system and the major software modules. The fourth chapter describes the software in some detail, while the fifth chapter discusses results obtained in this implementation. Chapter six lists several proposals intended to improve the system, and chapter seven concludes the report.

II. BACKGROUND

Distributed data bases and associated problems have been widely discussed (2-16) but relatively little has been implemented. For this report, the following definitions will be used:

1. Front-end - acts as user interface, receives input, transmits output.
2. Host - executes application program.
3. Back-end - controls data access by execution of data base operations.
4. Bi-functional - combines host and back-end functions.

These are the same definitions used in the technical report authored by Maryanski, et al. (17).

Canaday, et al. (18) implemented an experimental distributed data base management system called XDMS which is similar to the implementation discussed in this report. Canaday's implementation consisted of a single back-end (Digital Scientific META-4) and a UNIVAC 1108 as the host computer, whereas this implementation will use the Interdata 8/32 as a bi-functional machine and the Interdata 7/32 as a back-end. A second difference is that the XDMS software as written to conform to the CODASYL DBTG standards, while this implementation utilizes TOTAL (which does not conform completely to the CODASYL DBTG standards) as the data base management system. A separate message system and a series of interface routines between the application program and the data base management system complete the implementation

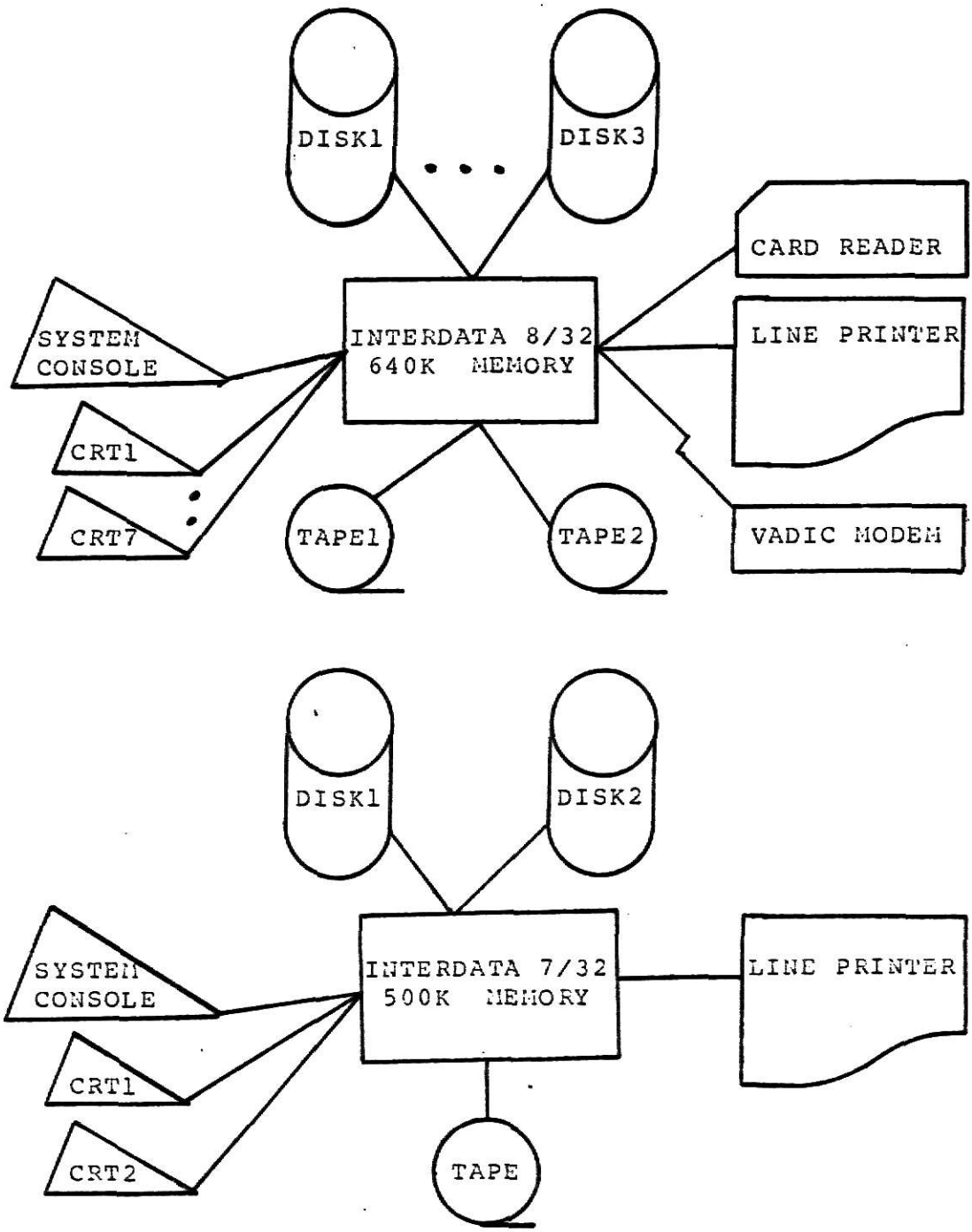
to make it a distributed data base management system. Canaday achieves a multi-user system by using re-entrant code, whereas this implementation achieves a multi-user system by including the small interface routines in each application program's task. The last difference is that Canaday's system allows concurrent changes to the data base and provides rollback and recovery features. Since this implementation is based on the version of TOTAL available on Interdata machines, concurrent update, rollback, and recovery are not found in this implementation.

III. SYSTEM ORGANIZATION

The hardware utilized in a distributed data base management system could range from mainframes for hosts, front-ends, and back-ends, a combination of mainframes and minicomputers, or a system based on minicomputers only. The use of minicomputers in a distributed data base system provides an economical and reliable alternative to the use of mainframes (4, 8, 17). This implementation is based upon an Interdata 8/32 and an Interdata 7/32. the Interdata 8/32 acts as both host, front-end, and back-end, while the Interdata 7/32 acts solely as a back-end. There is nothing in the hardware or software which would prevent the Interdata 7/32 from acting as host or front-end if so desired. The hardware associated with each computer and the lines connecting them are shown in FIGURE 1.

The software can be divided into four categories: the data base management system, the interface routines, the message system, and the application program. The concepts, functions, and relationships of these software modules are discussed in references (2, 17, 21). An attempt was made to use existing software in the implementation. A commercial data base system, TOTAL, by CINCOM, INC. (20), was chosen for the data base management system. The message system is a generalized system implemented in Concurrent PASCAL, which was developed for the MIMICS network (19). The interface routines are the host interface (HINT) and the back-end interface (BINT), both of which are written in COBOL. These interfaces allow the application program, also written in

FIGURE 1


DISK1
DISK3
CARD READER
SYSTEM CONSOLE
INTERDATA 8/32
640K MEMORY
LINE PRINTER
CRT1
CRT7
TAPE1
TAPE2
VADIC MODEM
DISK1
DISK2
SYSTEM CONSOLE
INTERDATA 7/32
500K MEMORY
LINE PRINTER
CRT1
CRT2
TAPE

COBOL, to access any data base in the network via the message system without the user knowing upon which machine the data base resides. A third interface routine named MESSINFC, written in Interdata Common Assembler Language, is necessary to allow HINT and BINT to communicate with the message system.

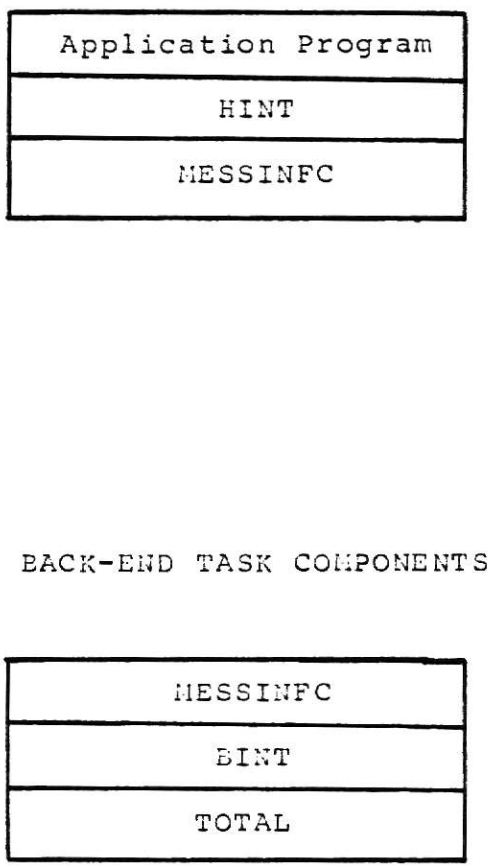
The following steps are taken when a task on the front-end (host) machine issues a data base command:

1. A data base command in the application program produces a call to the host interface (HINT).
2. HINT formats the data base command into a message and through MESSINFC instructs the message system to send the message to the appropriate back-end interface (BINT).
3. The message system transmits the message from HINT.
4. BINT receives the message through MESSINFC and unpacks the message.
5. BINT calls the data base management system to perform the operation specified.
6. The data base management system executes the data base command.
7. The data base management system returns the data (if any) and the status to BINT.
8. The results returned by the data base management system to BINT are repacked and sent back through MESSINFC to the message system for return to HINT.
9. MESSINFC receives the returned message which HINT unpacks before returning control to the application

program.

These steps are almost identical to those given in reference (17). A diagram of the software components of each front-end and back-end task is shown is FIGURE 2.

FIGURE 2

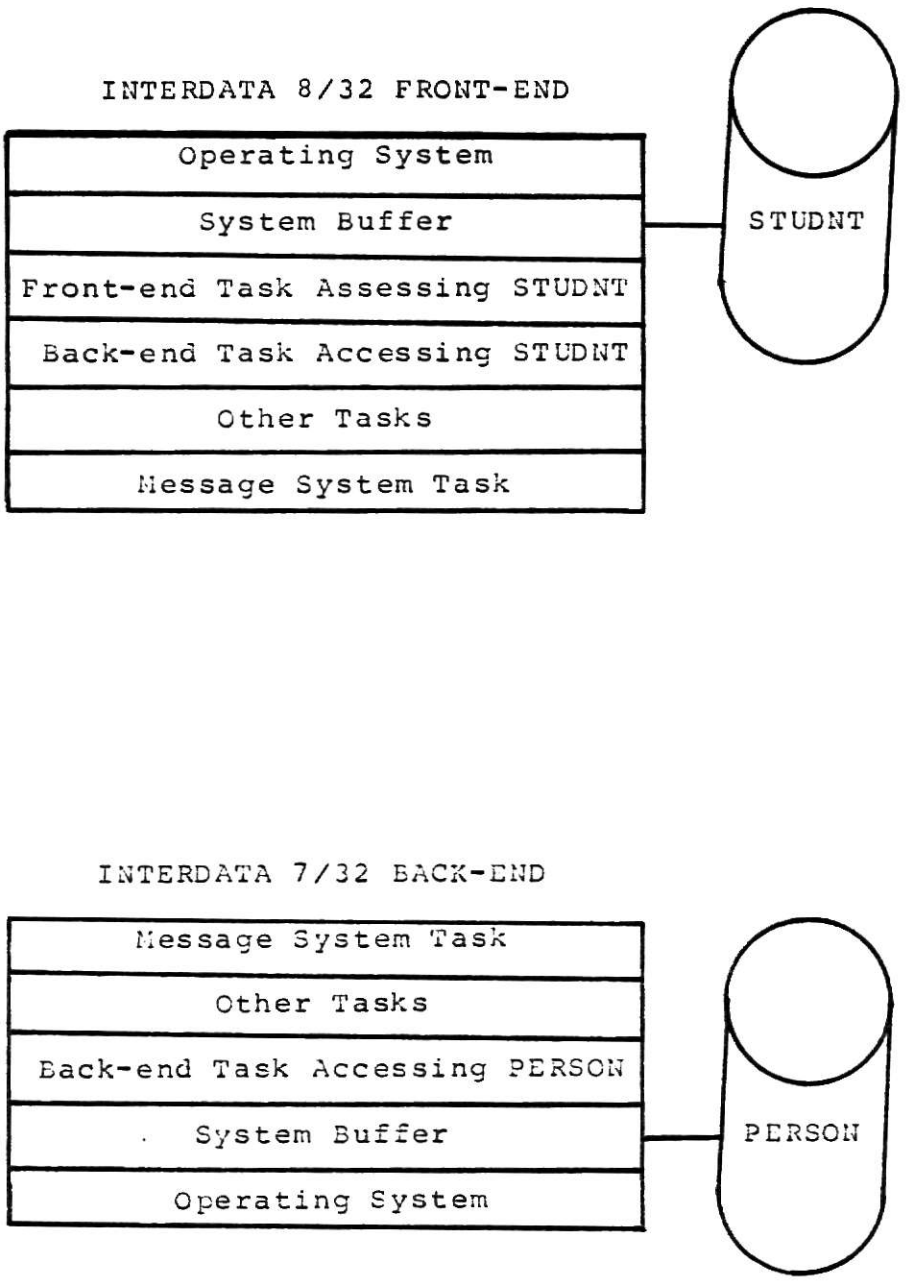

FRONT-END TASK COMPONENTS
Application Program
HINT
MESSINFC
BACK-END TASK COMPONENTS
MESSINFC
BINT
TOTAL

IV. SOFTWARE DESIGN

There were two major objectives in mind when the software for the distributed data base management system was written. First of all, a high degree of portability was desired so that a variety of computers could be incorporated into the distributed data base system network. To achieve this objective the application programs , back-end interface (BINT), host interface (HINT), and the message system are written in high-level languages. Only a small amount of code, called MESSINFC (see APPENDIX F), is written in machine dependent assembler language. The second objective was to make no changes in the TOTAL data base management system and place minimal restrictions on the application program. Both of these objectives have been met reasonably well.

Two data bases were created in this implementation: PERSON (see APPENDIX B) resides on the Interdata 7/32 and STUDNT (see APPENDIX C) resides on the Interdata 8/32, as shown in FIGURE 3. Both data bases are 'synthetic' in that the contents of each data base is arbitrary, they are used only in proving the feasibility of the system. Note that the normal DBGEN CSS supplied by CINCOM, INC. requires the use of background when generating a data base from the DDL. Because of the configuration of the Interdata machines and the functions they perform, it was desirable to do all data base generations in the foreground. To do this, two new CSS programs, DBGEN1 and DBGEN2 (see APPENDIX A) were written. DBGEN1 must be executed and terminated normally before

FIGURE 3


INTERDATA 8/32 FRONT-END
Operating System
System Buffer
Front-end Task Assessing STUDNT
Back-end Task Accessing STUDNT
Other Tasks
Message System Task
STUDNT
INTERDATA 7/32 BACK-END
Message System Task
Other Tasks
Back-end Task Accessing PERSON
System Buffer
Operating System
PERSON

executing DBGEN2. The remainder of the steps taken during the creation of a TOTAL data base up through and including the DBFORMAT remain the same.

The COBOL source code for I832BINT and I732BINT, the back-end interfaces for the Interdata 8/32 and the Interdata 7/32 respectively, is given in APPENDIX G and APPENDIX H. These routines are nearly identical copies of each other. This was done as a first step towards a generalized BINT.

The actions of I732BINT can be described as follows: upon activation of the back-end task, I732BINT issues a call in paragraph ESTAB-TASK-ID to the message system in order to establish a connection to it. If the return status code of the connection attempt is non-zero, an error message is displayed and the task is terminated. Experience has shown that the cause of a bad connection is generally in the message system itself, so the back-end task should not be re-activated until the message system is cleared. Upon a return code of zero, execution proceeds to paragraph HELLO-THERE where it asks the message system for any requests awaiting it. It will continue to loop in this paragraph until it receives a request or until it has a bad receive. If it is a bad receive, execution branches to paragraph SYS-SINOF where I732BINT disconnects from the message system and terminates execution. With a good receive from the message system, I732BINT proceeds to paragraph BINT-UNPACK to unpack the input parameter, and it executes paragraph BINT-EXECUTE-CALL where it branches to the correct paragraph for executing the requested data base

management function. Should the requested function be invalid, BINT-ERROR is executed. Paragraphs BV1 through BV6 select the parameters required for the requested call to DATBAS, makes the call, and branches to BINT-REPACK. BUF1 is repacked with the data and the status returned by the call to DATBAS (or from BINT-ERROR) in BINT-REPACK in preparation for the return through the message system. Paragraph GOOD-BYE attempts to send the contents of BUF1 back to the front-end task. This paragraph will loop until it successfully sends BUF1 into the message system, in which case it returns to MAIN-LOOP for the next request, or until the return status indicates a bad receive by the message system (which causes I732BINT to disconnect and terminate). When the last user issues a SINOF command to the data base management system, I732BINT will automatically terminate execution.

To create a new version of the back-end interfaces to run in a second task on the same machine, or to accept commands from another front-end task, the FROMID and TOID must be changed to the correct machine and task identifying code. The message system needs these codes to know who sent it the message (FROMID) and where it is to send a message back to (TOID). For example, in I732BINT, FROMID has the character string value 'AAUQ' which indicated the Interdata 7/32 with the characters 'AA' and the particular task is identified by the 'UQ'. The character string value 'ABRH' of TOID indicates the Interdata 8/32 with the characters 'AB' and the particular task with 'RH'.

A second change which may need to be made is the lengths of some elements in the input parameter BUF1. Referring to I732BINT again, the length of the first element is set by the fact that there are only six diffenent types of calls to DATBAS that I732BINT can make. The next five elements in BUF1 have lengths required by TOTAL. The elements BCTRL, BVELEM, and BVAREA,. however, have lengths which may vary as the application programs using the system change. For example, BCTRL (the data base control key) has been given a length of nine even though the data base PERSON has a control key with a length of six. This is because the data base STUDNT had a control key with a length of nine. Should another data base with a control key length of twelve be added to the network of distributed data bases, all of the BINT routines would need to have the length of BCTRL changed to twelve. This is another step taken towards a generalized BINT. Similar changes must be made to BVELEM and BVAREA as necessary. When changes such as these are made, the same type of change should also be made in the parameters CTRL, VELEM, and VAREA which are used in the calls to DATBAS.

Once a data base has been created and necessary changes have been made in the BINT routine which accesses it, another CSS program is used to link MESSINFC, the BINT routine, and the data base together in a single task. This CSS was named COBLEST, and its code may be seen in APPENDIX E. When this CSS is executed, the resulting back-end task is named according to the interface name supplied to it as a

parameter, e.g.-

COBLEST I732BINT,USR6:MESSINFC,TOTAL,PERSON

will create a back-end task named I732BINT.TSK. If several back-end tasks are to be created, the Interdata RENAME command can be used to change the name of this task as needed, e.g.-

REN I732BINT.TSK,PERSBACK.TSK

The COBOL source code for I732HINT and I832HINT, the host interfaces for the Interdata 7/32 and Interdata 8/32 respectively, is given is APPENDIX K and APPENDIX L. As a part of the move towards a generalized HINT, these routines are also nearly identical copies of each other.

The actions of I732HINT can be described as follows: the most often used elements of the message system parameter BUF1 are given values from the parameters passed to it from the application program. Paragraph HINT-CALL-SELECT determines which function the user wants the data base management system to execute and branches to the appropriate paragraph. If an invalid function is requested, an error status is loaded and I732HINT returns control to the application program. The paragraph CALL-HINT-V006 is responsible for connecting the front-end task to the message system or disconnecting it from the message system upon receiving the SINCN or SINOF data base management function request. The paragraph SYS-SINON thru SYS-SINON-EXIT is performed to connect the front-end and the message system. If the connection is not good, I732HINT returns control to the application program. Paragraph SYS-SINOF thru

SYS-SINOF-EXIT is executed when the application program issues a SINOF data base management command. Paragraphs CALL-HINT-V001 thru CALL-HINT-V006 select the parameters required for executing the requested data base management function and places them in BUF1. The number moved to the element CALLER of BUF1 will tell the back-end task which call to use in requesting a function from the data base management system. It then performs CALL-BINT thru CALL-EXIT to pass the contents of BUF1 through the message system to the correct back-end. If BUF1 is not passed to the message system correctly in CALL-BINT1, it will either loop until the message system accepts the parameters correctly or it will display an error message and return to the application program if there is a serious error involving the message system. If the message system accepts the parameters correctly, execution of I732HINT proceeds to the next paragraph, CALL-BINT2, which will loop until it correctly receives the expected message back from the back-end task or it will display an error message and return to the application program if there is a serious error involving the message system. Assuming that BUF1 was transmitted and received with no problems, I732HINT will return control to the application program with whatever data and status the application program would expect from the data base management system.

The application programs are LOADPEOP, which accesses data base PERSON, and LOADSTUD, which accesses data base STUDNT. The COBOL source code for each program is found in

APPENDIX N and APPENDIX O respectively. The application programs accept only the following commands: ADD-M, DEL-M, WRITM, READM, READV, DELVD, ADDVC and WRITV. It is felt that this is a reasonable selection of commands to implement and test. Both application programs have almost identical structure, so only the basic functions of LOADPEOP will be discussed here. Upon initiation of the front-end task, LOADPEOP will attempt to sign on to the data base PERSON through the message system in the paragraph SIGN-ON. If the SINON command is not executed correctly, control passes to paragraph CLOSE-EM and execution terminates. If the SINON command is executed correctly, paragraph GET-COMMAND is executed. The user is expected to enter one of the TOTAL functions listed above (or the word CEASE to terminate execution) and the number of successive times the user expects to use the command. LOADPEOP will then perform paragraph MASTER-TRANS or VARIABLE-TRANS, depending on the function, the specified number of times. The user may enter data or a carriage return as necessary in these paragraphs. When the user has entered the CEASE command, execution branches to paragraph CLOSE-EM to issue the SINOF command to the data base and terminates execution.

Application programs using TOTAL in a 'normal' environment and these programs, designed to utilize the distributed data base management system, are not coded in a radically different way. The differences are listed below:

1. Every call to DATBAS is replaced by a call to HINT.
2. Every call to HINT must have nine parameters, some

of which may by dummy parameters. The ordering of these parameters is important. Using the terminology of CINCOM, INC., the parameters and their order is: FUNCTION, STATUS, FILE, REFERENCE, LINKPATH, CONTROL, ELEMENTS, DATA AREA, END.

3. Some parameters may have to be padded with a FILLER clause or be declared to have a length somewhat longer than what it would have when not utilizing the distributed data base management system.

These are very straight foward changes to make. A pre-processor could be used to scan the call statements in the application program and change the calling name to BINT, add dummy parameters as necessary, and change parameter lengths to those specified by the data base administrator. This would be a good step toward a fully user-transparent distributed data base system. The reason for difference 1 should be obvious at this time. Difference 2 is to provide a single calling format to HINT for all data base management function requests. As previously discussed, HINT and BINT will determine which parameters are needed in the call to DATBAS. Since a generalized HINT and BINT is an objective to be met in the near future, the parameters CONTROL, ELEMENTS, and DATA AREA must be declared to have lengths corresponding to the lengths of these parameters in HINT and BINT. Since the longest control key passing through HINT and BINT in this implementation had a length of nine (for data base STUDNT), all parameters in the application programs, HINT, and BINT corresponding to the CONTROL

parameter must have a length of nine also. Similarly, the longest ELEMENTS parameter passing through HINT and BINT appears in application program LOADSTUD when accessing the variable record CRSE with data base STUDNT. This parameter (CRSE-DATA) has a length of sixty, so all other parameters corresponding to elements in the application programs, HINT, and BINT must also be declared to have this length. Lastly, the longest DATA AREA parameter passing through HINT and BINT appears in application program LOADPEOP when accessing the master record PEOP within data base PERSON. This parameter (PEOP-RECORD) has a length of eighty-two, so all other parameters corresponding to DATA AREA in the application program, HINT, and BINT must also be declared to have this length. These differences are not numerous or difficult to implement.

To create a front-end task using an application program, a HINT routine, and MESSINFC, a CSS program is used. This CSS was named COBESTAB, and its code may be seen in APPENDIX J. When this CSS is executed, the resulting front-end task is named according to the interface name supplied to it as a parameter, e.g.-

```
COBESTAB I732HINT,USR6:RICH3,USR6:MESSINFC
```

will create a front-end task named I732HINT.TSK. If several front-end tasks are to be created, the Interdata RENAME command can be used to change the name of this task as needed, e.g.-

```
REN I732HINT.TSK,PERSBACK.TSK
```

When bringing up the distributed data base management

system, the following steps should be taken in order:

1. Initiate the message system.
2. Initiate the back-end task(s).
3. Initiate the front-end task(s).

Since the message system is vital to the front-end and back-end, it should be started first. The back-end is started second since it must be executing normally before the front-end can be used. The front-end is brought up last since it relies on having both the message system and the back-end executing normally.

To initiate a back-end task, the following steps are taken at a user's console:

```
LOAD back-end task name
ASSIGN 0,CON:
START
```

To initiate front-end tasks, enter the following at the user's console:

```
LOAD front-end task name
ASSIGN 0,CON:
ASSIGN 1,CON:
ASSIGN 2,hard copy device code
START
```

V. RESULTS

This implementation of a distributed data base management system does appear to be a viable approach. Most of the testing of the system was done as changes were being made in the system. As such, no exhaustive system testing has been done at this time. A wide variety of activities, such as execution of COBOL, FORTRAN, PASCAL, and Assembler programs, editing, etc., take place on the Interdata machines. During a majority of the periods when the distributed data base management system was being tested, these other activities represented a significant load on the machines. This may make up for at least part of the loading heavy data base work would create.

When accessing a data base in the normal manner, the response time was instantaneous for all practical purposes - on rare occasions it took perhaps two seconds to return. Response time increased significantly when using the distributed data base management system. A front-end task on the Interdata 8/32 accessing a data base on the same machine had a response time of roughly four seconds. When using a front-end task on the Interdata 8/32 to access a data base on the Interdata 7/32, response time jumped to seventeen seconds. These response times are unacceptable in a commercial environment, but continuing developments in the distributed data base management system are expected to make the response times acceptable. It should be noted, however, that response time would not be further lengthened when running multiple front-end tasks, which would make these

response times seem more reasonable.

VI. FUTURE ENHANCEMENTS

Work is currently underway to incorporate an IBM 370/158 in the system. The data base management system is TOTAL (22) as on the Interdata machines. There are a few differences in TOTAL for the Interdata and TOTAL for the IBM, but these differences are not great. One difference is that on the IBM machine a data base must be opened, signed on, and then closed and signed off, while the Interdata version only has the sign on and sign off to execute. The data base residing on the IBM is called PIECES (see APPENDIX D), and the back-end interface is called IBMBINT (see APPENDIX I). The function of IBMBINT is virtually the same as I732BINT and I832BINT. The message system on the IBM is written in Assembler Language, so no equivalent of MESSINFC and NEWINFC is needed - IBMBINT runs as a sub-task of the message system. Because of this, the method of checking for bad receives and sends is made in a separate call to the message system after the call for a receive or send. The host interface is called IBMHINT (APPENDIX M), which is modified to handle the TOTAL, OPENM, CLOSM, and DEQUE IBM TOTAL functions and to pass the ID's correctly. These ID's must be specified carefully so that the correct ASCII to EBCDIC conversion occurs. The calls to TRANSA and TRANSE provide translation for BUF1 and BUF1LTH. Since the message system does not handle floating point numbers, care should be taken to see that they are avoided in the application programs. The application program LOADPART, listed in APPENDIX P, is also quite similar to the other application

programs. Since this segment of the project is not fully operational, the programs listed in the appendices may be subject to many changes.

A second enhancement to the system will be a directory disk file in which the unique machine identifiers (TOID's and FROMID's) for all tasks and all data base files will be kept. Rather than editing the HINT routines to create a new front-end task or to add a data base to the system, the data base administrator will add the appropriate codes to the directory. The host interface will then search this directory and find the codes it needs.

A third improvement to the system would be the creation of a re-entrant code version of HINT and BINT to reside in system library space. Implementing this and the directory file will eliminate the work now needed to expand the system, improve performance, and perhaps provide some extra system security. With protected directory files, new CSS programs to create front-end and back-end tasks, and CSS programs to initiate tasks, the generalized HINT and BINT would help security by allowing the system user to have virtually no knowledge of the distributed data base management system's software, hardware, and their inter-relationships.

The overall response time of the system could be lowered by both hardware and software modifications. From a hardware point of view, high speed modems and transmission lines would help speed up the system, but the major factor in the high response time seems to be in the message system

software. Greater speed could probably be obtained by coding the message system in Common Assembler Language rather than in PASCAL, but this would be at the expense of portability. These changes would be somewhat harder to make than the preceeding ones.

VII. CONCLUSIONS

This implementation shows distributed data base management systems can be created with minimal changes to existing data base management systems and with little or no additional hardware expense. With further work on the system, it could become competitive with the current non-distributed data base management systems in terms of economy, security, reliability, and ease of use.

A distributed data base management system in which an Interdata 8/32 and Interdata 7/32 could act as bi-functional machines, if so desired, was implemented. Other processors could be connected into the system easily as they are needed. The software utilized TOTAL, a commercially available data base management system, a PASCAL message system previously implemented at Kansas State University, and several front-end and back-end interface routines written in COBOL and Common Assembler Language. The hardware and software implementation details of the distributed data base management system are almost completely hidden from the user.

REFERENCES

1. Tsichritzis, D. C., Lochovsky, F. H., Data Base Management Systems, Academic Press, Inc., New York, New York, 1977.

2. Maryanski, F. J., et al., Distributed Data Base Management Using Minicomputers, Infotech State of the Art Report, 'Minis, Midis, and Mainframes', April 1978.

3. Fisher, P. S., Maryanski, F. J., Design Considerations in Distributed Data Base Management Systems, TR-CS 77-08, Dept. of Computer Science, Kansas State University, Manhattan, Kansas, April 1977.

4. Maryanski, F. J., A Survey of Developments in Distributed Data Base Management Systems, Computer, Vol. 11, No. 2, pp. 28-38, Feb. 1978.

5. Fry, J. P., Deppe, M. E., Distributed Data Bases: A Summary of Research, Computer Networks, Vol. 1, No. 2, 1976.

6. Aschim, F., Data Base Networks - An Overview, Management Informatics, Vol. 3.1, pp. 12-28, Feb. 1974.

7. Booth, G. M., The Use of Distributed Data Bases in Information Networks, Proc. 1st. International Conference on Computer Communication: Impacts and Implications, pp. 371-376, Oct. 1972.

8. Rosenthal, R. S., An Evaluation of Backend Data Base Management Machines, MRI Systems Corporation, P.O. Box 9968, Austin, Texas, 1977.

9. Comba, Paul G., Needed: Distributed Control, Proceedings 1st Conference on Very Large Data Bases, pp. 364-375, Sept. 1975.

10. Peebles, R., Manning, Eric, A Computer Architecture for Large (Distributed) Data Bases, Prodeedings 1st Conference on Very Lagre Data Bases, pp. 405-427, Sept. 1975.

11. Maryanski, F. J., Wallentine, V. E., A Simulation Model of a Back-end Data Base Management System, Proceedings 7th Pittsburgh Symposium on Modeling and Simulation, pp. 252-257, April 1976.

12. Maryanski, F. J., Performance of Multi-Processor Back-end Data Base Systems, Proceedings Conference on Information Science and Systems, pp. 437-441, April 1977.

13. Maryanski, F. J., Kreimer, D. E., Effects of Distributed Processing in a Data Processing Environment, Proceedings 11th Annual Simulation Symposium, pp. 183-197, Mar. 1978.

14. Maryanski, F. J., Fisher, P. S., Rollback and Recovery in Distributed Data Base Management Systems, Proceedings ACM Annual Conference, pp. 33-38, Oct. 1977.

15. Chu, W. W., Ohlmacher, G., Avoiding Deadlock in Distributed Data Bases, Proceedings ACM Annual Conference, pp. 156-160, Nov. 1974.

16. Maryanski, F. J., A Deadlock Prevention Algorithm for Distributed Data Base Management Systems, TR-CS 77-02, Dept. of Computer Science, Kansas State University, Manhattan, Kansas, Feb. 1977.

17. Maryanski, F. J., et al., Distributed Data Base Management: An Overview and Example, TR-CS 78-08, Dept. of Computer Science, Kansas State University, Manhattan, Kansas, Feb. 1978

18. Canaday, R. E., et al., A Back-end Computer for Data Base Management, CACM, Vol. 17, No. 10, pp. 575-582, Oct. 1974.

19. Wallentine, V. E., et al., An Overview of the MIMICS Design, TR-CS 77-04, Dept. of Computer Science, Kansas State University, Manhattan, Kansas, Dec. 1976.

20. CINCOM, INC., Interdata TOTAL Reference Manual, Cincinnati, O., 1977.

21. Maryasnki, F. J., et al., A User-Transparent Mechanism for the Distribution of A CODASYL Data Base Management System, TR-CS 76-22, Dept. of Computer Science, Kansas State University, Manhattan, Kansas, Dec. 1976.

22. CINCOM, INC., OS TOTAL Reference Manual, Cincinnati, Ohio, 1976.

APPENDIX A

```
$COPY
**********************************************
*
*    INTERDATA TOTAL DATA BASE GENERATION
*
**********************************************
$NO
$IFX @1.CAL;DE @1.CAL;$ENDC
$IFX DBGEN.WRK;DE DBGEN.WRK;$ENDC
AL @1.CAL,IN,80/3
AL DBGEN.WRK,IN,256/1
LO DBGEN.TSK
AS 1,@2
AS 2,@3
AS 3,@1.CAL
AS 4,DBGEN.WRK
ST
$EXIT
```

```
LO CAL.TSK
$IFX @1.OBJ;DEL @1.OBJ;$ENDC
AL @1.OBJ,IN,126/1
$IFNX CALS4.SCT;AL CALS4.SCT,IN,86/5;$ENDC
$IFNX CALS5.SCT;AL CALS5.SCT,IN,1024/1;$ENDC
$IFNX CALS6.SCT;AL CALS6.SCT,IN,126/1;$ENDC
$IFNX CALS8.SCT;AL CALS8.SCT,IN,86/5;$ENDC
$IFNX CALS9.SCT;AL CALS9.SCT,IN,86/5;$ENDC
AS 1,@1.CAL
AS 2,@1.OBJ
AS 3,@3
AS 4,CALS4.SCT
AS 5,CALS5.SCT
AS 6,CALS6.SCT
AS 7,NULL:
AS 8,CALS8.SCT
AS 9,CALS9.SCT
ST ,SCRAT,NORXT
$EXIT
```

APPENDIX B

```
BEGIN-DATA-BASE-GENERATION
DATA-BASE-NAME=PERSON
OPTIONS=OUTPUT=Y
SHARE-IO
IOAREA=MAS1
IOAREA=VAR1
END-IO
BEGIN-MASTER-DATA-SET
DATA-SET-NAME=PEOP
IOAREA=MAS1
MASTER-DATA
PEOPROOT=8
PEOPCTRL=6      PIC X(6)    CONTROL KEY FOR PEOP
PEOPLKPE=8                  LINK TO PERS RECORDS (ALL)
PEOPNAME=25     PIC X(25)   NAME
PEOPSOSC=9      PIC X(9)    SOCIAL SECURITY NUMBER
PEOPADDR=20     PIC X(20)   ADDRESS
PEOPBIRT=6      PIC X(6)    BIRTHDAY
PEOPHIRE=6      PIC X(6)    HIRE DATE
PEOPSEXX=1      PIC X(1)    SEX
PEOPDIVN=2      PIC 99      DIVISION
PEOPTELE=10     PIC X(10)   TELEPHONE NUMBER
PEOPFILL=20     PIC X(20)   FILLER
END-DATA
DRIVE=010,21,SYS3
TOTAL-LOGICAL-RECORDS=21
END-MASTER-DATA-SET
BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=PERS
IOAREA=VAR1
BASE-DATA
PERSPEOP=6      PIC X(6)    CONTROL KEY TO PEOP RECORD
PERSLKPE=8=PERSPEOP         LINK FROM PEOP TO PERS
PERSSTRT=6      PIC X(6)    DATE EXPERIENCE STARTED
```

```
PERSEXPR=2      PIC X(2)    EXPERIENCE LEVEL CODE
PERSDATA=21     PIC X(21)   PERSONAL DATA
END-DATA
DRIVE=012,227,SYS3
LOAD-LIMIT=70
TOTAL-LOGICAL-RECORDS=227
END-VARIABLE-ENTRY-DATA-SET
END-DATA-BASE-GENERATION
```

****
*   *
* APPENDIX C *
*   *
****

```
BEGIN-DATA-BASE-GENERATION
DATA-BASE-NAME=STUDNT
OPTIONS=OUTPUT=Y
SHARE-IO
IOAREA=MAS1
IOAREA=VAR1
END-IO
BEGIN-MASTER-DATA-SET
DATA-SET-NAME=STUD
IOAREA=MAS1
MASTER-DATA
STUDROOT=8
STUDCTRL=9      PIC X(9)    CONTROL KEY FOR STUD
STUDLKCR=8                  LINK TO CRSE RECORDS (ALL)
STUDNAME=25     PIC X(25)   NAME
STUDSTAR=6      PIC X(6)    DATE STARTED
STUDGRAD=6      PIC X(6)    DATE TO GRADUATE
STUDYEAR=4      PIC X(4)    YEAR IN SCHOOL
STUDXTRA=10     PIC X(10)   FILLER
END-DATA
DRIVE=010,25,SYS2
TOTAL-LOGICAL-RECORDS=25
END-MASTER-DATA-SET
BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=CRSE
IOAREA=VAR1
BASE-DATA
CRSESTUD=9      PIC X(9)    CONTROL KEY TO STUD RECORD
STUDLKCR=8=CRSESTUD         LINK FROM STUD TO CRSE
CRSENAME=30     PIC X(30)   COURSE NAME
CRSENMBR=6      PIC X(6)    COURSE NUMBER
CRSELINO=4      PIC X(4)    COURSE LINE NUMBER
CRSESMSR=2      PIC X(2)    SEMESTER
CRSETORE=1      PIC X(1)    TAKEN OR ENROLLED
```

```
CRSEHORS=1        PIC X(1)    HOURS
CRSEGRAD=1        PIC X(1)    GRADE
END-DATA
DRIVE=012,150,SYS12
LOAD-LIMIT=70
TOTAL-LOGICAL-RECORDS=150
END-VARIABLE-ENTRY-DATA-SET
END-DATA-BASE-GENERATION
```

APPENDIX D

```
//STEP1    EXEC DUMMY
//DD1      DD DSN=DS2L1.LOADLIB(LOADPART),DISP=(OLD,DELETE)
//DD2      DD DSN=DS2L1.PART,DISP=(OLD,DELETE),UNIT=SYSDA,
//         SPACE=(TRK,1),VOL=REF=DS2L1.TOTAL.LINKLIB
//DD3      DD DSN=DS2L1.SUPL,DISP=(OLD,DELETE),UNIT=SYSDA,
//         SPACE=(TRK,2),VOL=REF=DS2L1.TOTAL.LINKLIB
//STEP2    EXEC DUMMY
//DD1      DD DSN=DS2L1.LOADLIB(LOADPART),VOL=REF=DS2L1.TOTAL.LINKLIB,
//         SPACE=(TRK,(15,15,3)),DISP=(NEW,CATLG),DCB=(BLKSIZE=13030,
//         RECFM=U),UNIT=SYSDA
//DBGEN    EXEC PGM=DBGEN
//STEPLIB DD DSN=DS2L1.TOTAL.LINKLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD DSN=&&DBSOURCE,DISP=(,PASS),UNIT=SYSDA,
//          SPACE=(TRK,(5,5)),DCB=BLKSIZE=3120
//SYSIN    DD *
BEGIN-DATA-BASE-GENERATION;
DATA-BASE-NAME=PIECES
SHARE-IO
IOAREA=MAS1
IOAREA=VAR1
BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=PART
IOAREA=MAS1
MASTER-DATA:
PARTROOT=8
PARTCTRL=9     PIC X(9)     CONTROL KEY FOR PART
PARTLKSU=9                  LINK TO SUPL RECORDS (ALL)
PARTNAME=25    PIC X(25)    NAME
PARTDESC=30    PIC X(30)    PART DESCRIPTION
PARTFILL=20    PIC X(20)    FILLER
END-DATA;
DEVICE=3350
TOTAL-LOGICAL-RECORDS=25
```

```
END-MASTER-DATA-SET:
BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=SUPL
IOAREA=VAR1
BASE-DATA:
SUPLPART=9    PIC X(9)    CONTROL KEY TO PART RECORD
PARTLKSU=8                LINK FROM PART TO SUPL
SUPLNAME=25   PIC X(25)   SUPPLIER NAME
SUPLADDR=20   PIC X(20)   SUPPLIER ADDRESS
SUPLTELE=10   PIC X(10)   SUPPLIER TELEPHONE
SUPLDATA=16   PIC X(16)   SUPPLIER DATA
END-DATA:
DEVICE=3330
TOTAL-LOGICAL-RECORDS=100
CYLINDER-LOAD-LIMIT=70
END-VARIABLE-ENTRY-DATA-SET:
END-DATA-BASE-GENERATION:
//ASM   EXEC ASMGCL,COND=(4,LT,DBGEN)
//SYSIN DD DSN=&&DBSOURCE,DISP=(OLD,DELETE)
//LKED.SYSLMOD DD DSN=DS2L1.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
   NAME PIECES(R)
//FORMAT EXEC PGM=FORMAT
//STEPLIB DD DSN=DS2L1.LOADLIB,DISP=SHR
// DD DSN=DS2L1.TOTAL.LINKLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//PART DD DSN=DS2L1.PART,DISP=(,CATLG),UNIT=SYSDA,
//  SPACE=(TRK,1),VOL=REF=DS2L1.TOTAL.LINKLIB
//SUPL DD DSN=DS2L1.SUPL,DISP=(,CATLG),UNIT=SYSDA,
//  SPACE=(TRK,2),VOL=REF=DS2L1.TOTAL.LINKLIB
//SYSIN DD *
 FORMAT PIECES  PART,SUPL
```

APPENDIX E

```
$IFNULL @1;$COPY
*  CALLING FORMAT:  COB1EST INTERFACE NAME,ASMBLR INTERFACE NAME,
*  TOTAL,DATA-BASE NAME
$NO
$EXIT;$ENDC
$IFNX @1.OBJ;$COPY
*   FILE @1.OBJ DOES NOT EXIST
$NO
$CLEAR ;$EXIT ;$ENDC
$IFX SYS2:ESTAB.CMD  ;DEL SYS2:ESTAB.CMD  ;$ENDC
$IFNX @1.TCM
$BUILD SYS2:ESTAB,CMD
ES TA
IN @1.OBJ
EDIT SYS2:CB0200.LIB/S
IN @2.OBJ
IN @3.OBJ
IN @4.OBJ
BU TA,@1.TSK
AMAP
END
$ENDB
$ENDC
$IFX  @1.TSK;DE @1.TSK;$ENDC
$IFNX SYS2:ESTAB.SCR ;AL SYS2:ESTAB.SCR,IN,126/4 ;$ENDC
$IFX SYS2:ESTAB.LST  ;DEL SYS2:ESTAB.LST  ;$ENDC
AL SYS2:ESTAB.LST,IN,126/4
LO TET32
AS 1,@1.OBJ
AS 3,PR:
TE 4,IN,126
$IFX @1.TCM;AS 5,@1,TCM;$ENDC
$IFNX @1.TCM;AS 5,SYS2:ESTAB.CMD;$ENDC
AS 7,CON:
```

```
ST
$EXIT
```

APPENDIX F

```
*       TITLE     MESSAGE SYSTEM INTERFACE FROM INTERDATA COBOL
*
*       POSSIBLE TASK CANCELLATION ERRORS
*           BECAUSE OF BAD COBOL CALL
*
*       CODE      MEANING
*
*        54         WRONG NUMBER OF ARGUMENTS
*        55         ILLEGAL REMOTE-ID LENGTH
*        56         ILLEGAL LOCAL-ID LENGTH
*        57         ILLEGAL STATUS LENGTH
*
*
*       POSSIBLE BAD STATUS RETURNS
*           NOT DUE TO MESSAGE SYSTEM
*
*       CODE      MEANING
*
*        58         UNABLE TO DETERMINE LOCAL-ID
*
*
*       COBOL CALLING SEQUENCES AND DATA TYPES
*
*
*       1)    CALL 'SEND' USING REMOTE-ID, MESSAGE, STATUS
*
*       2)    CALL 'RECV' USING REMOTE-ID, MESSAGE, STATUS
*
*       3)    CALL 'CONN' USING REMOTE-ID, LOCAL-ID, STATUS
*
*       4)    CALL 'DISC' USING REMOTE-ID, STATUS
*
*           WHERE THE FOLLOW DATA RULES APPLY
*
```

```
*         REMOTE-ID IS A 77 OR 01 LEVEL NAME OF LENGTH 4
*
*         LOCAL-ID IS A 77 OR 01 LEVEL NAME OF LENGTH 4
*
*         STATUS IS A 77 OR 01 LEVEL NAME OF LENGTH 2
*
*         MESSAGE IS A 77 OR 01 LEVEL NAME
*
          TITLE     INITIAL SET UP
*
*         REGISTER DEFINITIONS
*
ALL       EQU       0
TEMP2     EQU       0
LINK      EQU       1
TEMP      EQU       1
TOWHOM    EQU       2
STATUS    EQU       2
WHOAMI    EQU       2
ERROR     EQU       3
COMMD     EQU       4
ERLINK    EQU       5
BUF1ADD   EQU       6
LEN1ADD   EQU       7
BUF2ADD   EQU       8
LEN2ADD   EQU       9
BUF3ADD   EQU       10
LEN3ADD   EQU       11
ARG4      EQU       12
ARG1      EQU       13
ARG2      EQU       14
ARG3      EQU       15
 TITLE PROCEDURE SEND-MESSAGE (REMOTE-ID,MESSAGE,STATUS)
*
```

```
*         PROC SEND-MESSAGE (REMOTE-ID,MESSAGE,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'SEND'
*         DO P05-CHECK-CALLER-ARGUMENTS
*         RECV-SCB (REMOTE-ID) =
*             TO-WHOM
*         SEND-SCB (DATA-BUFFER-ADDRESS) =
*             BUFER-ADDRESS-2ND-ARGUMENT
*         SEND-SCB (DATA-BUFFER-LENGTH) =
*             BUFFER-LENGTH-2ND-ARGUMENT
*         SEND-THE-MESSAGE
*         CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) =
*             SEND-SCB (STATUS)
*         RETURN
*
          SPACE
          SPACE
          SPACE
          ENTRY     SEND
SEND      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
          LI        COMMD,C'SEND'
          BAL       LINK,P5CHKARG
          ST        TOWHOM,SENDRMID
          LIS   TOWHOM,0          R01- JIM R. - DEC. 3, 1977
          ST    TOWHOM,SENDTXBF   R01- JIM R. - DEC. 3, 1977
          ST    TOWHOM,SENDCMBF   R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,SENDLNTX   R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,SENDLNCM   R01- JIM R. - DEC. 3, 1977
          CHI   LEN2ADD,129       R01- JIM R. - DEC. 3, 1977
          BL    SEND.CMD          R01- JIM R. - DEC. 3, 1977
          ST        BUF2ADD,SENDTXBF
          STH       LEN2ADD,SENDLNTX
```

```
          B       SEND.GO             R01- JIM R. - DEC. 3, 1977
SEND.CMD  ST      BUF2ADD,SENDCMBF    R01- JIM R. - DEC. 3, 1977
          STH     LEN2ADD,SENDLNCM    R01- JIM R. - DEC. 3, 1977
SEND.GO   EQU     *                   R01- JIM R. - DEC. 3, 1977
          SVC         0,SENDSVC
          LH          STATUS,SENDCDOU
          STH         STATUS,0(BUF3ADD)
          LM          ALL,REGSAV
          A           15,0(15)
          BR          15
 TITLE PROCEDURE RECEIVE-MESSAGE (REMOTE-ID,MESSAGE,STATUS)
*
*         PROC        RECV (REMOTE-ID,MESSAGE,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'RECV'
*         DO P05-CHECK-CALLER-ARGUMENTS
*         RECV-SCB (REMOTE-ID) =
*             TO-WHOM
*         RECV-SCB (DATA-BUFFER-ADDRESS) =
*             BUFFER-ADDRESS-2ND-ARGUMENT
*         RECV-SCB (DATA-BUFFER-LENGTH) =
*             BUFFER-LENGTH-2ND-ARGUMENT
*         RECEIVE-THE-MESSAGE
*         CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) =
*             RECV-SCB (STATUS)
*         RETURN
*
          SPACE
          SPACE
          SPACE
          ENTRY     RECV
RECV      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
```

```
          LI        COMMD,C'RECV'
          BAL       LINK,P5CHKARG
          ST        TOWHOM,RECVRMID
          LIS   TOWHOM,0
          ST    TOWHOM,RECVTXBF R01- JIM R. - DEC. 3, 1977
          ST    TOWHOM,RECVCMBF R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,RECVLNCM R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,RECVLNTX R01- JIM R. - DEC. 3, 1977
          CHI   LEN2ADD,129     R01- JIM R. - DEC. 3, 1977
          BL    RECV.CMD        R01- JIM R. - DEC. 3, 1977
          ST        BUF2ADD,RECVTXBF
          STH       LEN2ADD,RECVLNTX
          B     RECV.GO         R01- JIM R. - DEC. 3, 1977
RECV.CMD  ST    BUF2ADD,RECVCMBF  R01- JIM R. - DEC. 3, 1977
          STH   LEN2ADD,RECVLNCM R01- JIM R. - DEC. 3, 1977
RECV.GO   EQU   *               R01- JIM R. - DEC. 3, 1977
          SVC       0,RECVSVC
          LH        STATUS,RECVCDOU
          STH       STATUS,0(BUF3ADD)
          LM        ALL,REGSAV
          A         15,0(15)
          BR        15
 TITLE PROCEDURE CONNECT (REMOTE-ID,LOCAL-ID,STATUS)
*
*         PROC      CONN (REMOTE-ID,LOCAL-ID,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'CONN'
*         DO P05-CHECK-CALLER-ARGUMENTS
*         DO P06-CHECK-LOCAL-ID
*         GET-LOCAL-ID
*         IF SUCCESSFUL-ID-RETRIEVE
*             CONN-SCB (REMOTE-ID) =
*                 TO-WHOM
```

```
*               WHO-AM-I =
*                   LOCAL-ID (BUFFER-ADDRESS-2ND-ARGUMENT)
*               CALLER (LOCAL-ID) =
*                   CALLER (WHO-AM-I)
*               BUILD-NEW-LOCAL-ID
*               SEND-SCB (LOCAL-ID) =
*                   REBUILT-LOCAL-ID
*               RECV-SCB (LOCAL-ID) =
*                   REBUILT-LOCAL-ID
*               CONN-SCB (LOCAL-ID) =
*                   REBUILT-LOCAL-ID
*               DISC-SCB (LOCAL-ID) =
*                   REBUILT-LOCAL-ID
*               CONNECT-TO-MESSAGE-SYSTEM
*               CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) =
*                   CONN-SCB (STATUS)
*              CALLER (WHO-AM-I) =
*                   REBUILT-LOCAL-ID
*          ELSE
*              CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) = 58
*          ENDIF
*          RETURN
*
           SPACE
           SPACE
           SPACE
           ENTRY     CONN
CONN       STM       ALL,REGSAV
           BAL       LINK,P10DUMB
           LI        COMMD,C'CONN'
           BAL       LINK,P5CHKARG
           ST        TOWHOM,CONNRMID ;CHANGED TO SAVE REG. 2
           L      LINK,LOCALID     R01- JIM R. - DEC. 3, 1977
           BNZ    MULTCONN         NOT THE FIRST CONNECTION,
```

```
*                                   ;  SO DON'T DO THE GETID SVC0.
          BAL       LINK,P6CKLCID
          SVC       0,GETIDSVC
          LH        STATUS,GETSTAT
          BNZ       P3E001
          L         WHOAMI,0(BUF2ADD)
          NI        WHOAMI,X'FFFF'
          LR        TEMP2,WHOAMI
MULTCONN  EQU   *                 R01- JIM R. - DEC. 3, 1977
          L         WHOAMI,LOCALID
          NI        WHOAMI,Y'FFFF0000'
          OR        WHOAMI,TEMP2
          ST        WHOAMI,SENDLCID
          ST        WHOAMI,RECVLCID
          ST        WHOAMI,CONNLCID
          ST        WHOAMI,DISCLCID
          ST        WHOAMI,0(BUF2ADD)
          SVC       0,CONNSVC
          LH        STATUS,CONNCDOU
          STH       STATUS,0(BUF3ADD)
          BS        P3X001
P3E001    EQU       *
          LHI       STATUS,58
          STH       STATUS,0(BUF3ADD)
P3X001    EQU       *
          LM        ALL,REGSAV
          A         15,0(15)
          BR        15
 TITLE PROCEDURE DISCONNECT (REMOTE-ID,STATUS)
*
*         PROC      DISC (REMOTE-ID,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'DISC'
```

```
*         DO P08-CHECK-FOR-TWO-ARGUMENTS
*         DO P07-CHECK-REMOTE-ID
*         DISCONNECT-FROM-MESSAGE-SYSTEM
*         CALLER (BUFFER-ADDRESS-2RD-ARGUMENT) =
*             DISC-SCB (STATUS)
*         RETURN
*
          SPACE
          SPACE
          SPACE
          ENTRY     DISC
DISC      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
          LI        COMMD,C'DISC'
          BAL       LINK,P8CK2ARG
          ST    TOWHOM,DISCRMID ;STORE REMOTE ID.
          BAL       LINK,P7CKRMID
          SVC       0,DISCSVC
          LH        STATUS,DISCCDOU
          STH       STATUS,0(BUF2ADD)
          LM        ALL,REGSAV
          A         15,0(15)
          BR        15
          TITLE     P05-CHECK-CALLER-ARGUMENTS
*
*         PROC      P05-CHECK-CALLER-ARGUMENTS
*
*         IF NUMBER-OF-CALLER-ARGUMENTS NOT = 3
*             CALLER-ERROR = 1
*             DO P09-CALLER-ERROR
*         ELSE
*             LENGTH-1ST-ARGUMENT =
*                 CALLER (LENGTH-1ST-ARGUMENT)
*             IF LENGTH-1ST-ARGUMENT NOT = 4
```

```
*                   CALLER-ERROR = 2
*                   DO P09-CALLER-ERROR
*               ELSE
*                   LENGTH-3RD-ARGUMENT =
*                       CALLER (LENGTH-3RD-ARGUMENT)
*                   IF LENGTH-3RD-ARGUMENT NOT = 2
*                       CALLER-ERROR = 4
*                       DO P09-CALLER-ERROR
*                   ELSE
*                       BUFFER-ADDRESS-1ST-ARGUMENT =
*                           CALLER (BUFFER-ADDRESS-1ST-ARGUMENT)
*                       TO-WHOM =
*                           REMOTE-ID (BUFFER-ADDRESS-1ST-ARGUMENT)
*                       BUFFER-ADDRESS-2ND-ARGUMENT =
*                           CALLER (BUFFER-ADDRESS-2ND-ARGUMENT)
*                       BUFFER-LENGTH-2ND-ARGUMENT =
*                           CALLER (BUFFER-LENGTH-2ND-ARGUMENT)
*                       BUFFER-ADDRESS-3RD-ARGUMENT =
*                           CALLER (BUFFER-ADDRESS-3RD-ARGUMENT)
*                   ENDIF
*               ENDIF
*           ENDIF
*           RETURN
*
            SPACE
            SPACE
            SPACE
P5CHKARG    LM          ARG4,0(15)
            CLHI        ARG4,16                 4(N+1)
            BES         P5E001
            LHI         ERROR,1
            BAL         ERLINK,P9ERROR
            B           P5X001
P5E001      EQU         *
```

```
          L           LEN1ADD,4(ARG1)
          CLHI        LEN1ADD,4
          BES         P5E002
          LHI         ERROR,2
          BAL         ERLINK,P9ERROR
          B           P5X002
P5E002    EQU         *
          L           LEN3ADD,4(ARG3)
          CLHI        LEN3ADD,2
          BES         P5E003
          LHI         ERROR,4
          BAL         ERLINK,P9ERROR
          B           P5X003
P5E003    EQU         *
 STM ALL,TEMPSAV
          L           BUF1ADD,0(ARG1)
          L           TOWHOM,0(BUF1ADD)
          L           BUF2ADD,0(ARG2)
          L           LEN2ADD,4(ARG2)
          L           BUF3ADD,0(ARG3)
P5X003    EQU         *
P5X002    EQU         *
P5X001    EQU         *
          BR          LINK
 TITLE P06-CHECK-LOCAL-ID
*
*         PROC        P06-CHECK-LOCAL-ID
*
*         IF BUFFER-LENGTH-2RD-ARGUMENT NOT = 4
*             CALLER-ERROR = 4
*             DO P09-CALLER-ERROR
*         ENDIF
*         RETURN
*
```

```
          SPACE
          SPACE
          SPACE
P6CKLCID  CLHI      LEN2ADD,4
          BES       P6E001
          LHI       ERROR,4
          BAL       ERLINK,P9ERROR
          BS        P6X001
P6E001    EQU       *
P6X001    EQU       *
          BR        LINK
 TITLE P07-CHECK-REMOTE-ID
*
*         PROC      P07-CHECK-REMOTE-ID
*
*         IF BUFFER-LENGTH-1ST-ARGUMENT NOT = 4
*             CALLER-ERROR = 2
*             DO P09-CALLER-ERROR
*         ENDIF
*         RETURN
*
          SPACE
          SPACE
          SPACE
P7CKRMID  CLHI      LEN1ADD,4
          BES       P7E001
          LHI       ERROR,2
          BAL       ERLINK,P9ERROR
          BS        P7X001
P7E001    EQU       *
P7X001    EQU       *
          BR        LINK
 TITLE P08-CHECK-FOR-TWO-ARGUMENTS
*
```

```
*          PROC        P08-CHECK-FOR-TWO-ARGUMENTS
*
*          IF NUMBER-OF-CALLER-ARGUMENTS NOT = 2
*               CALLER-ERROR = 1
*               DO P09-CALLER-ERROR
*          ELSE
*               BUFFER-ADDRESS-1ST-ARGUMENT =
*                    CALLER (BUFFER-ADDRESS-1ST-ARGUMENT)
*               TO-WHOM =
*                    REMOTE-ID (BUFFER-ADDRESS-1ST-ARGUMENT)
*               BUFFER-LENGTH-1ST-ARGUMENT =
*                    CALLER (BUFFER-LENGTH-1ST-ARGUMENT)
*               BUFFER-ADDRESS-2ND-ARGUMENT =
*                    CALLER (BUFFER-ADDRESS-2ND-ARGUMENT)
*               BUFFER-LENGTH-2ND-ARGUMENT =
*                    CALLER (BUFFER-LENGTH-2ND-ARGUMENT)
*          RETURN
*
          SPACE
          SPACE
          SPACE
P08CK2ARG L#          ARG1,0(15)
          CLM1        ARG1,12                4(N+1)
          BL          P8E001
          LM1         ERROR,1
          BAL         ERLINK,P-ERROR
          B           P8X001
P8E001    EQU         *
          L           BUF1ADD,0(ARG2)
          L           TOWHOM,0(BUF1ADD)
          L           LEN1ADD,4(ARG2)
          L           BUF2ADD,0(ARG3)
          L           LEN2ADD,4(ARG3)
P8X001    EQU         *
```

```
          DR          LINK
 TITLE P09-CALLER-ERROR
*
*         PROC        P09-CALLER-ERROR
*
*         CALLER-ERROR = (CALLER-ERROR - 1) * 8
*         MESSAGE-LENGTH =
*             ERROR-TABLE (CALLER-ERROR)
*         CONSOLE-MESSAGE-LENGTH =
*             MESSAGE-LENGTH
*         MESSAGE-ADDRESS =
*             ERROR-TABLE + 4 (CALLER-ERROR)
*         CONSOLE-MESSAGE-LENGTH =
*             MESSAGE-ADDRESS
*         ERROR-MESSAGE-PREFACE (MESSAGE-ADDRESS) =
*             COMMAN
*         ISSUE-MESSAGE-TO-CONSOLE
*         CANCEL-TASK (100)
*
          SPACE
          SPACE
          SPACE
P09ERROR  STM         ERROR,1
          SLLS        ERROR,3
          L           TEMP,ERRTAB(ERROR)
          STH         TEMP,ERRLEN
          L           TEMP,ERRTAB+4(ERROR)
          ST          TEMP,ERRADD
          ST          COMREG,0(TEMP)
          SVC         2,ERRSVC
          SVC         3,100
 TITLE P10-CHECK-DUMB-INTERDATA-ADDRESS
*
*         PROC        P10-CHECK-DUMB-INTERDATA-ADDRESS
```

```
*
*         IF DUMB-INTERDATA-ADDRESS-ON-HALFWORD
*             PUT-DUMB-INTERDATA-ADDRESS-ON-FULLWORD
*         ENDIF
*         RETURN
*
          SPACE
          SPACE
          SPACE
P10DUMB   NI        15,2
          BZS       P10E001
          L         15,LASTREG
          AIS       15,2
          ST        15,LASTREG
          BS        P10X001
P10E001   EQU       *
          L         15,LASTREG
P10X001   EQU       *
          BR        LINK
          TITLE     LOCAL DATA AREAS
*
*         REGISTER SAVE AREA
*
REGSAV    DAS       15
LASTREG   DAS       1
TEMPSAV   DAS 16
*
*         ERROR MESSAGE TABLE
*
          ALIGN     4
ERRTAB    DAC       MES1LEN,MESS1,MES2LEN,MESS2
          DAC       MES3LEN,MESS3,MES4LEN,MESS4
*
*         ERROR MESSAGES
```

```
*
          ALIGN     4
MESS1     DC        C'XXXX-54 WRONG NUMBER OF ARGUMENTS'
MES1LEN   EQU       *-MESS1
          ALIGN     4
MESS2     DC        C'XXXX-55 ILLEGAL REMOTE-ID LENGTH'
MES2LEN   EQU       *-MESS2
          ALIGN     4
MESS3     DC        C'XXXX-56 ILLEGAL LOCAL-ID LENGTH'
MES3LEN   EQU       *-MESS3
          ALIGN     4
MESS4     DC        C'XXXX-57 ILLEGAL STATUS LENGTH'
MES4LEN   EQU       *-MESS4
          TITLE     SVC BLOCKS
*
*         GET ID SVC
*
          ALIGN     4
GETIDSVC  DB        0                   COMMAND
          DB        1                   WHAT EVER
GETSTAT   DC        H'0'                STATUS OF GET ID
LOCALID   DAC       0                   ID IN THE FORM CMTP
          DAC       0                   - NUMBER INDICATES PASCAL
*
*         SEND SVC
*
          ALIGN     4
SENDSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(SENDSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         RECEIVE SVC
```

```
*
          ALIGN     4
RECVSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(RECVSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         CONNECT SVC
*
          ALIGN     4
CONNSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(CONNSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         DISCONNECT SVC
*
          ALIGN     4
DISCSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(DISCSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         CONSOLE MESSAGE SVC
*
          ALIGN     4
ERRSVC    DB        X'40',7
ERRLEN    DC        H'0'
ERRADD    DAC       0
          TITLE     SCB BLOCKS
*
```

```
*         SEND SCB
*
          ALIGN     4
SENDSCB   EQU   *
SENDRMID  DS        4                   REMOTE ID
SENDLCID  DS        4                   LOCAL ID
SENDCMBF  DAC       0                   COMMAND TEXT BUFFER
SENDTXBF  DAC       0                   DATA TEXT BUFFER
SENDLNCM  DC        H'0'                LENGTH OF COMMAND TEXT
SENDLNTX  DC        H'0'                LENGTH OF DATA TEXT
SENDCDIN  DC        H'60'               CODE,IN (MESSAGE COMMAND)
SENDCDOU  DS        2                   CODE,OUT MESSAGE RETURN STATUS
SENDCMD   DC        H'2'                MESSAGE COMMAND CLASS
*
          SPACE 3
*
*         RECEIVE SCB
*
          ALIGN     4
RECVSCB   EQU   *
RECVRMID  DS        4                   REMOTE ID
RECVLCID  DS        4                   LOCAL ID
RECVCMBF  DAC       0                   COMMAND TEXT BUFFER
RECVTXBF  DAC       0                   DATA TEXT BUFFER
RECVLNCM  DC        H'0'                LENGTH OF COMMAND TEXT
RECVLNTX  DC        H'0'                LENGTH OF DATA TEXT
RECVCDIN  DC        H'63'               CODE,IN (MESSAGE COMMAND)
RECVCDOU  DC        H'0'                CODE,OUT (MESSAGE RETURN STATUS)
RECVCMD   DC        H'3'                MESSAGE COMMAND CLASS
*
          EJECT
*
*         CONNECT SCB
*
```

```
         ALIGN     4
CONNSCB  EQU   *
CONNRMID DS        4                    REMOTE ID
CONNLCID DS        4                    LOCAL ID
CONNCMBF DAC       0                    COMMAND TEXT BUFFER
CONNTXBF DAC       0                    DATA TEXT BUFFER
CONNLNCM DC        H'0'                 LENGTH OF COMMAND TEXT
CONNLNTX DC        H'0'                 LENGTH OF DATA TEXT
CONNCDIN DC        H'3'                 CODE.IN (MESSAGE COMMAND)
CONNCDOU DS        2                    CODE.OUT (MESSAGE RETURN STATUS)
CONNCMD  DC        H'0'                 MESSAGE COMMAND CLASS
*
         SPACE
         SPACE
         SPACE
*
*        DISCONNECT SCB
*
         ALIGN     4
DISCSCB  EQU   *
DISCRMID DS        4
DISCLCID DS        4
DISCCMBF DAC       0                    COMMAND TEXT BUFFER
DISCTXBF DAC       0                    DATA TEXT BUFFER
DISCLNCM DC        H'0'                 LENGTH OF COMMAND TEXT
DISCLNTC DC        H'0'                 LENGTH OF DATA TEXT
DISCCDIN DC        H'5'                 CODE.IN (MESSAGE COMMAND)
DISCCDOU DS        2                    CODE.OUT (MESSAGE RETURN STATUS)
DISCCML  DC        H'1'                 MESSAGE COMMAND CLASS
*        TITLE SYMBOL DICTIONARY
         END
```

```
          TITLE     MESSAGE SYSTEM INTERFACE FROM INTERDATA COBOL
*
* REVISION 02: REMOTE-ID LENGTH RESTRICTIONS RELAXED TO
*   ALLOW FIVE BYTE PARAMETER (OF WHICH ONLY FIRST FOUR ARE USED)
*
*
*         POSSIBLE TASK CANCELLATION ERRORS
*             BECAUSE OF BAD COBOL CALL
*
*         CODE        MEANING
*
*          54           WRONG NUMBER OF ARGUMENTS
*          55           ILLEGAL REMOTE-ID LENGTH
*          56           ILLEGAL LOCAL-ID LENGTH
*          57           ILLEGAL STATUS LENGTH
*
*
*         POSSIBLE BAD STATUS RETURNS
*             NOT DUE TO MESSAGE SYSTEM
*
*         CODE        MEANING
*
*          58           UNABLE TO DETERMINE LOCAL-ID
*
*
*         COBOL CALLING SEQUENCES AND DATA TYPES
*
*
*         1)     CALL 'SEND' USING REMOTE-ID, MESSAGE, STATUS
*
*         2)     CALL 'RECV' USING REMOTE-ID, MESSAGE, STATUS
*
*         3)     CALL 'CONN' USING REMOTE-ID, LOCAL-ID, STATUS
*
```

```
*          4)    CALL 'DISC' USING REMOTE-ID, STATUS
*
*              WHERE THE FOLLOW DATA RULES APPLY
*
*          REMOTE-ID IS A 77 OR 01 LEVEL NAME OF LENGTH 4
*             (OR 5 WHERE FIRST 4 BYTES ARE USED)
*
*          LOCAL-ID IS A 77 OR 01 LEVEL NAME OF LENGTH 4
*
*          STATUS IS A 77 OR 01 LEVEL NAME OF LENGTH 2
*
*          MESSAGE IS A 77 OR 01 LEVEL NAME
*
           TITLE     INITIAL SET UP
*
*          REGISTER DEFINITIONS
*
ALL        EQU       0
TEMP2      EQU       0
LINK       EQU       1
TEMP       EQU       1
TOWHOM     EQU       2
STATUS     EQU       2
WHOAMI     EQU       2
ERROR      EQU       3
COMMD      EQU       4
ERLINK     EQU       5
BUF1ADD    EQU       6
LEN1ADD    EQU       7
BUF2ADD    EQU       8
LEN2ADD    EQU       9
BUF3ADD    EQU       10
LEN3ADD    EQU       11
ARG4       EQU       12
```

```
ARG1      EQU       13
ARG2      EQU       14
ARG3      EQU       15
 TITLE PROCEDURE SEND-MESSAGE (REMOTE-ID,MESSAGE,STATUS)
*
*         PROC SEND-MESSAGE (REMOTE-ID,MESSAGE,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'SEND'
*         DO P05-CHECK-CALLER-ARGUMENTS
*         RECV-SCB (REMOTE-ID) =
*             TO-WHOM
*         SEND-SCB (DATA-BUFFER-ADDRESS) =
*             BUFER-ADDRESS-2ND-ARGUMENT
*         SEND-SCB (DATA-BUFFER-LENGTH) =
*             BUFFER-LENGTH-2ND-ARGUMENT
*         SEND-THE-MESSAGE
*         CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) =
*             SEND-SCB (STATUS)
*         RETURN
*
          SPACE
          SPACE
          SPACE
          ENTRY     SEND
SEND      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
          LI        COMMD,C'SEND'
          BAL       LINK,P5CHKARG
          ST        TOWHOM,SENDRMID
          LIS   TOWHOM,0          R01- JIM R. - DEC. 3, 1977
          ST    TOWHOM,SENDTXBF   R01- JIM R. - DEC. 3, 1977
          ST    TOWHOM,SENDCMBF   R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,SENDLNTX   R01- JIM R. - DEC. 3, 1977
```

```
          STH   TOWHOM,SENDLNCM R01- JIM R. - DEC. 3, 1977
          CHI   LEN2ADD,129     R01- JIM R. - DEC. 3, 1977
          BL    SEND.CMD        R01- JIM R. - DEC. 3, 1977
          ST        BUF2ADD,SENDTXBF
          STH       LEN2ADD,SENDLNTX
          B     SEND.GO         R01- JIM R. - DEC. 3, 1977
SEND.CMD  ST    BUF2ADD,SENDCMBF R01- JIM R. - DEC. 3, 1977
          STH   LEN2ADD,SENDLNCM R01- JIM R. - DEC. 3, 1977
SEND.GO   EQU   *               R01- JIM R. - DEC. 3, 1977
          SVC       0,SENDSVC
          LH        STATUS,SENDCDOU
          STH       STATUS,0(BUF3ADD)
          LM        ALL,REGSAV
          A         15,0(15)
          BR        15
 TITLE PROCEDURE RECEIVE-MESSAGE (REMOTE-ID,MESSAGE,STATUS)
*
*         PROC      RECV (REMOTE-ID,MESSAGE,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'RECV'
*         DO P05-CHECK-CALLER-ARGUMENTS
*         RECV-SCB (REMOTE-ID) =
*             TO-WHOM
*         RECV-SCB (DATA-BUFFER-ADDRESS) =
*             BUFFER-ADDRESS-2ND-ARGUMENT
*         RECV-SCB (DATA-BUFFER-LENGTH) =
*             BUFFER-LENGTH-2ND-ARGUMENT
*         RECEIVE-THE-MESSAGE
*         CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) =
*             RECV-SCB (STATUS)
*         RETURN
*
          SPACE
```

```
          SPACE
          SPACE
          ENTRY     RECV
RECV      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
          LI        COMMD,C'RECV'
          BAL       LINK,P5CHKARG
          ST        TOWHOM,RECVRMID
          LIS   TOWHOM,0
          ST    TOWHOM,RECVTXBF R01- JIM R. - DEC. 3, 1977
          ST    TOWHOM,RECVCMBF R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,RECVLNCM R01- JIM R. - DEC. 3, 1977
          STH   TOWHOM,RECVLNTX R01- JIM R. - DEC. 3, 1977
          CHI   LEN2ADD,129     R01- JIM R. - DEC. 3, 1977
          BL    RECV.CMD        R01- JIM R. - DEC. 3, 1977
          ST        BUF2ADD,RECVTXBF
          STH       LEN2ADD,RECVLNTX
          B     RECV.GO         R01- JIM R. - DEC. 3, 1977
RECV.CMD  ST    BUF2ADD,RECVCMBF  R01- JIM R. - DEC. 3, 1977
          STH   LEN2ADD,RECVLNCM R01- JIM R. - DEC. 3, 1977
RECV.GO   EQU   *               R01- JIM R. - DEC. 3, 1977
          SVC       0,RECVSVC
          LH        STATUS,RECVCDOU
          STH       STATUS,0(BUF3ADD)
          LM        ALL,REGSAV
          A         15,0(15)
          BR        15
.TITLE PROCEDURE CONNECT (REMOTE-ID,LOCAL-ID,STATUS)
*
*         PROC      CONN (REMOTE-ID,LOCAL-ID,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'CONN'
*         DO P05-CHECK-CALLER-ARGUMENTS
```

```
*         DO P06-CHECK-LOCAL-ID
*         GET-LOCAL-ID
*         IF SUCCESSFUL-ID-RETRIEVE
*              CONN-SCB (REMOTE-ID) =
*                  TO-WHOM
*              WHO-AM-I =
*                  LOCAL-ID (BUFFER-ADDRESS-2ND-ARGUMENT)
*              CALLER (LOCAL-ID) =
*                  CALLER (WHO-AM-I)
*              BUILD-NEW-LOCAL-ID
*              SEND-SCB (LOCAL-ID) =
*                  REBUILT-LOCAL-ID
*              RECV-SCB (LOCAL-ID) =
*                  REBUILT-LOCAL-ID
*              CONN-SCB (LOCAL-ID) =
*                  REBUILT-LOCAL-ID
*              DISC-SCB (LOCAL-ID) =
*                  REBUILT-LOCAL-ID
*              CONNECT-TO-MESSAGE-SYSTEM
*              CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) =
*                  CONN-SCB (STATUS)
*              CALLER (WHO-AM-I) =
*                  REBUILT-LOCAL-ID
*         ELSE
*              CALLER (BUFFER-ADDRESS-3RD-ARGUMENT) = 58
*         ENDIF
*         RETURN
*
          SPACE
          SPACE
          SPACE
          ENTRY     CONN
CONN      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
```

```
         LI        COMMD,C'CONN'
         BAL       LINK,P5CHKARG
         ST        TOWHOM,CONNRMID ;CHANGED TO SAVE REG. 2
         L     LINK,LOCALID      R01- JIM R. - DEC. 3, 1977
         BNZ   MULTCONN          NOT THE FIRST CONNECTION,
*                                ;  SO DON'T DO THE GETID SVC0.
         BAL       LINK,P6CKLCID
         SVC       0,GETIDSVC
         LH        STATUS,GETSTAT
         BNZ       P3E001
         L         WHOAMI,0(BUF2ADD)
         NI        WHOAMI,X'FFFF'
         LR        TEMP2,WHOAMI
MULTCONN EQU   *                 R01- JIM R. - DEC. 3, 1977
         L         WHOAMI,LOCALID
         NI        WHOAMI,Y'FFFF0000'
         OR        WHOAMI,TEMP2
         ST        WHOAMI,SENDLCID
         ST        WHOAMI,RECVLCID
         ST        WHOAMI,CONNLCID
         ST        WHOAMI,DISCLCID
         ST        WHOAMI,0(BUF2ADD)
         SVC       0,CONNSVC
         LH        STATUS,CONNCDOU
         STH       STATUS,0(BUF3ADD)
         BS        P3X001
P3E001   EQU       *
         LHI       STATUS,58
         STH       STATUS,0(BUF3ADD)
P3X001   EQU       *
         LM        ALL,REGSAV
         A         15,0(15)
         BR        15
 TITLE PROCEDURE DISCONNECT (REMOTE-ID,STATUS)
```

```
*
*         PROC      DISC (REMOTE-ID,STATUS)
*
*         DO P10-CHECK-DUMB-INTERDATA-ADDRESS
*         COMMAND = 'DISC'
*         DO P08-CHECK-FOR-TWO-ARGUMENTS
*         DO P07-CHECK-REMOTE-ID
*         DISCONNECT-FROM-MESSAGE-SYSTEM
*         CALLER (BUFFER-ADDRESS-2ND-ARGUMENT) =
*             DISC-SCB (STATUS)
*         RETURN
*
          SPACE
          SPACE
          SPACE
          ENTRY     DISC
DISC      STM       ALL,REGSAV
          BAL       LINK,P10DUMB
          LI        COMMD,C'DISC'
          BAL       LINK,P8CK2ARG
          ST    TOWHOM,DISCRMID ;STORE REMOTE ID.
          BAL       LINK,P7CKRMID
          SVC       0,DISCSVC
          LH        STATUS,DISCCDOU
          STH       STATUS,0(BUF2ADD)
          LM        ALL,REGSAV
          A         15,0(15)
          BR        15
          TITLE     P05-CHECK-CALLER-ARGUMENTS
*
*         PROC      P05-CHECK-CALLER-ARGUMENTS
*
*         IF NUMBER-OF-CALLER-ARGUMENTS NOT = 3
*             CALLER-ERROR = 1
```

```
*            DO P09-CALLER-ERROR
*        ELSE
*            LENGTH-1ST-ARGUMENT =
*                CALLER (LENGTH-1ST-ARGUMENT)
*            IF LENGTH-1ST-ARGUMENT NOT = (4 OR 5)     R02-DAVE S-FEB 14, 1978
*                CALLER-ERROR = 2
*                DO P09-CALLER-ERROR
*            ELSE
*                LENGTH-3RD-ARGUMENT =
*                    CALLER (LENGTH-3RD-ARGUMENT)
*                IF LENGTH-3RD-ARGUMENT NOT = 2
*                    CALLER_ERROR = 4
*                    DO P09-CALLER-ERROR
*                ELSE
*                    BUFFER-ADDRESS-1ST-ARGUMENT =
*                        CALLER (BUFFER-ADDRESS-1ST-ARGUMENT)
*                    TO-WHOM =
*                        REMOTE-ID (BUFFER-ADDRESS-1ST-ARGUMENT)
*                    BUFFER-ADDRESS-2ND-ARGUMENT =
*                        CALLER (BUFFER-ADDRESS-2ND-ARGUMENT)
*                    BUFFER-LENGTH-2ND-ARGUMENT =
*                        CALLER (BUFFER-LENGTH-2ND-ARGUMENT)
*                    BUFFER-ADDRESS-3RD-ARGUMENT =
*                        CALLER (BUFFER-ADDRESS-3RD-ARGUMENT)
*                ENDIF
*            ENDIF
*        ENDIF
*        RETURN
*
         SPACE
         SPACE
         SPACE
PSCHKARG LM        ARG4,0(15)
         CLHI      ARG4,16              4(N+1)
```

```
         BES       P5E001
         LHI       ERROR,1
         BAL       ERLINK,P9ERROR
         B         P5X001
P5E001   EQU       *
         L         LEN1ADD,4(ARG1)
         CLHI      LEN1ADD,4
         BES       P5E002
         CLHI      LEN1ADD,5   R02-DAVE S-FEB 14, 1978
         BES       P5E002   R02-DAVE S-FEB 14,1978
         LHI       ERROR,2
         BAL       ERLINK,P9ERROR
         B         P5X002
P5E002   EQU       *
         L         LEN3ADD,4(ARG3)
         CLHI      LEN3ADD,2
         BES       P5E003
         LHI       ERROR,4
         BAL       ERLINK,P9ERROR
         B         P5X003
P5E003   EQU       *
 STM ALL,TEMPSAV
         L         BUF1ADD,0(ARG1)
         L         TOWHOM,0(BUF1ADD)
         L         BUF2ADD,0(ARG2)
         L         LEN2ADD,4(ARG2)
         L         BUF3ADD,0(ARG3)
P5X003   EQU       *
P5X002   EQU       *
P5X001   EQU       *
         BR        LINK
 TITLE P06-CHECK-LOCAL-ID
*
*        PROC      P06-CHECK-LOCAL-ID
```

```
*
*         IF BUFFER-LENGTH-2RD-ARGUMENT NOT = 4
*             CALLER-ERROR = 4
*             DO P09-CALLER-ERROR
*         ENDIF
*         RETURN
*
          SPACE
          SPACE
          SPACE
P6CKLCID  CLHI      LEN2ADD,4
          BES       P6E001
          LHI       ERROR,4
          BAL       ERLINK,P9ERROR
          BS        P6X001
P6E001    EQU       *
P6X001    EQU       *
          BR        LINK
 TITLE P07-CHECK-REMOTE-ID
*
*         PROC      P07-CHECK-REMOTE-ID
*
*         IF BUFFER-LENGTH-1ST-ARGUMENT NOT = (4 OR 5) R02-DAVE S-FEB 14,1978
*             CALLER-ERROR = 2
*             DO P09-CALLER-ERROR
*         ENDIF
*         RETURN
*
          SPACE
          SPACE
          SPACE
P7CKRMID  CLHI      LEN1ADD,4
          BES       P7E001
          CLHI      LEN1ADD,5   R02-DAVE S-FEB 14,1978
```

```
          BLS       P7E001  R02-DAVE S-FEB 14,1978
          LHI       ERROR,2
          BAL       ERLINK,P9ERROR
          BS        P7X001
P7E001    EQU       *
P7X001    EQU       *
          BR        LINK
 TITLE P08-CHECK-FOR-TWO-ARGUMENTS
*
*         PROC      P08-CHECK-FOR-TWO-ARGUMENTS
*
*         IF NUMBER-OF-CALLER-ARGUMENTS NOT = 2
*             CALLER-ERROR = 1
*             DO P09-CALLER-ERROR
*         ELSE
*             BUFFER-ADDRESS-1ST-ARGUMENT =
*                 CALLER (BUFFER-ADDRESS-1ST-ARGUMENT)
*             TO-WHOM =
*                 REMOTE-ID (BUFFER-ADDRESS-1ST-ARGUMENT)
*             BUFFER-LENGTH-1ST-ARGUEMENT =
*                 CALLER (BUFFER-LENGTH-1ST-ARGUMENT)
*             BUFFER-ADDRESS-2ND-ARGUMENT =
*                 CALLER (BUFFER-ADDRESS-2ND-ARGUMENT)
*             BUFFER-LENGTH-2ND-ARGUMENT =
*                 CALLER (BUFFER-LENGTH-2ND-ARGUMENT)
*         RETURN
*
          SPACE
          SPACE
          SPACE
P3CK2ARG  LM        ARG1,0(15)
          CLHI      ARG1,12             4(N+1)
          BE        P8E001
          LHI       ERROR,1
```

```
         BAL       ERLINK,P9ERROR
         B         P8X001
P8E001   EQU       *
         L         BUF1ADD,0(ARG2)
         L         TOWHOM,0(BUF1ADD)
         L         LEN1ADD,4(ARG2)
         L         BUF2ADD,0(ARG3)
         L         LEN2ADD,4(ARG3)
P8X001   EQU       *
         BR        LINK
 TITLE P09-CALLER-ERROR
*
*        PROC      P09-CALLER-ERROR
*
*        CALLER-ERROR = (CALLER-ERROR - 1) * 8
*        MESSAGE-LENGTH =
*            ERROR-TABLE (CALLER-ERROR)
*        CONSOLE-MESSAGE-LENGTH =
*            MESSAGE-LENGTH
*        MESSAGE-ADDRESS =
*            ERROR-TABLE + 4 (CALLER-ERROR)
*        CONSOLE-MESSAGE-LENGTH =
*            MESSAGE-ADDRESS
*        ERROR-MESSAGE-PREFACE (MESSAGE-ADDRESS) =
*            COMMAND
*        ISSUE-MESSAGE-TO-CONSOLE
*        CANCEL-TASK (100)
*
         SPACE
         SPACE
         SPACE
P9ERROR  SIS       ERROR,1
         SLLS      ERROR,3
         L         TEMP,ERRTAB(ERROR)
```

```
         STH       TEMP,ERRLEN
         L         TEMP,ERRTAB+4(ERROR)
         ST        TEMP,ERRADD
         ST        COMMD,0(TEMP)
         SVC       2,ERRSVC
         SVC       3,100
 TITLE P10-CHECK-DUMB-INTERDATA-ADDRESS
*
*        PROC      P10-CHECK-DUMB-INTERDATA-ADDRESS
*
*        IF DUMB-INTERDATA-ADDRESS-ON-HALFWORD
*            PUT-DUMB-INTERDATA-ADDRESS-ON-FULLWORD
*        ENDIF
*        RETURN
*
         SPACE
         SPACE
         SPACE
P10DUMB  NI        15,2
         BZS       P10E001
         L         15,LASTREG
         AIS       15,2
         ST        15,LASTREG
         BS        P10X001
P10E001  EQU       *
         L         15,LASTREG
P10X001  EQU       *
         BR        LINK
         TITLE     LOCAL DATA AREAS
*
*        REGISTER SAVE AREA
*
REGSAV   DAS       15
LASTREG  DAS       1
```

```
TEMPSAV DAS 16
*
*         ERROR MESSAGE TABLE
*
          ALIGN     4
ERRTAB    DAC       MES1LEN,MESS1,MES2LEN,MESS2
          DAC       MES3LEN,MESS3,MES4LEN,MESS4
*
*         ERROR MESSAGES
*
          ALIGN     4
MESS1     DC        C'XXXX-54 WRONG NUMBER OF ARGUMENTS'
MES1LEN   EQU       *-MESS1
          ALIGN     4
MESS2     DC        C'XXXX-55 ILLEGAL REMOTE-ID LENGTH'
MES2LEN   EQU       *-MESS2
          ALIGN     4
MESS3     DC        C'XXXX-56 ILLEGAL LOCAL-ID LENGTH'
MES3LEN   EQU       *-MESS3
          ALIGN     4
MESS4     DC        C'XXXX-57 ILLEGAL STATUS LENGTH'
MES4LEN   EQU       *-MESS4
          TITLE     SVC BLOCKS
*
*         GET ID SVC
*
          ALIGN     4
GETIDSVC  DB        0                   COMMAND
          DB        1                   WHAT EVER
GETSTAT   DC        H'0'                STATUS OF GET ID
LOCALID   DAC       0                   ID IN THE FORM CMTP
          DAC       0                   - NUMBER INDICATES PASCAL
*
*         SEND SVC
```

```
*
          ALIGN     4
SENDSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(SENDSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         RECEIVE SVC
*
          ALIGN     4
RECVSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(RECVSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         CONNECT SVC
*
          ALIGN     4
CONNSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(CONNSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
*
*         DISCONNECT SVC
*
          ALIGN     4
DISCSVC   DB        1                   COMMAND
          DB        0                   WHAT EVER
          DS        2                   STATUS
          DAC       A(DISCSCB)          ADDRESS OF SCB
          DS        4                   NOT USED
```

```
*
*         CONSOLE MESSAGE SVC
*
          ALIGN     4
ERRSVC    DB        X'40',7
ERRLEN    DC        H'0'
ERRADD    DAC       0
          TITLE     SCB BLOCKS
*
*         SEND SCB
*
          ALIGN     4
SENDSCB   EQU   *
SENDRMID  DS        4                   REMOTE ID
SENDLCID  DS        4                   LOCAL ID
SENDCMBF  DAC       0                   COMMAND TEXT BUFFER
SENDTXBF  DAC       0                   DATA TEXT BUFFER
SENDLNCM  DC        H'0'                LENGTH OF COMMAND TEXT
SENDLNTX  DC        H'0'                LENGTH OF DATA TEXT
SENDCDIN  DC        H'60'               CODE,IN (MESSAGE COMMAND)
SENDCDOU  DS        2                   CODE,OUT MESSAGE RETURN STATUS
SENDCMD   DC        H'2'                MESSAGE COMMAND CLASS
*
          SPACE 3
*
*         RECEIVE SCB
*
          ALIGN     4
RECVSCB   EQU   *
RECVRMID  DS        4                   REMOTE ID
RECVLCID  DS        4                   LOCAL ID
RECVCMBF  DAC       0                   COMMAND TEXT BUFFER
RECVTXBF  DAC       0                   DATA TEXT BUFFER
RECVLNCM  DC        H'0'                LENGTH OF COMMAND TEXT
```

```
RECVLNTX DC        H'0'          LENGTH OF DATA TEXT
RECVCDIN DC        H'63'         CODE,IN (MESSAGE COMMAND)
RECVCDOU DC        H'0'          CODE,OUT (MESSAGE RETURN STATUS)
RECVCMD  DC        H'3'          MESSAGE COMMAND CLASS
*
         EJECT
*
*        CONNECT SCB
*
         ALIGN     4
CONNSCB  EQU   *
CONNRMID DS        4             REMOTE ID
CONNLCID DS        4             LOCAL ID
CONNCMBF DAC       0             COMMAND TEXT BUFFER
CONNTXBF DAC       0             DATA TEXT BUFFER
CONNLNCM DC        H'0'          LENGTH OF COMMAND TEXT
CONNLNTX DC        H'0'          LENGTH OF DATA TEXT
CONNCDIN DC        H'3'          CODE,IN (MESSAGE COMMAND)
CONNCDOU DS        2             CODE,OUT (MESSAGE RETURN STATUS)
CONNCMD  DC        H'0'          MESSAGE COMMAND CLASS
*
         SPACE
         SPACE
         SPACE
*
*        DISCONNECT SCB
*
         ALIGN     4
DISCSCB  EQU   *
DISCRMID DS        4             REMOTE ID
DISCLCID DS        4             LOCAL IS
DISCCMBF DAC       0             COMMAND TEXT BUFFER
DISCTXBF DAC       0             DATA TEXT BUFFER
DISCLNCM DC        H'0'          LENGTH OF COMMAND TEXT
```

```
DISCLNTC DC        H'0'              LENGTH OF DATA TEXT
DISCCDIN DC        H'5'              CODE.IN (MESSAGE COMMAND)
DISCCDOU DS        2                 CODE.OUT (MESSAGE RETRUN STATUS)
DISCCMD  DC        H'1'              MESSAGE COMMAND CLASS
*
         TITLE SYMBOL DICTIONARY
         END
```

APPENDIX G

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  BINT.
 REMARKS.  'BINT' IS THE INTERFACE PROGRAM THAT
      DIRECTLY ACCESSES THE BACKEND DBMS.  IT RECEIVES
      THE CALLING PARAMETERS VIA THE PACKED PARAMETER
      BUFFERS BUF1 , DETERMINES THE CALL
      DESIRED, AND EXECUTES THAT CALL.  THE RESULTS
      ARE REPACKED AND THE PARAMETERS RETURNED.
*
* THIS VERSION DESIGNED FOR 7/32 TO 8/32 INTERTASK COMMUNICATION
*

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. INTERDATA MODEL-8-32.
 OBJECT-COMPUTER.  INTERDATA MODEL-8-32.
 INPUT-OUTPUT SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

*  DATBAS PARAMETERS:

 77  FUNCTION      PIC X(5).
 77  STAT          PIC X(4).
 77  FILE-NAME     PIC X(4).
 77  REFER         PIC X(4).
 77  LINKPATH      PIC X(8).
 77  CTRL          PIC X(9).
 77  VELEM         PIC X(60).
 77  VAREA         PIC X(82).
 77  ENDP          PIC X(4)  VALUE 'END.'.

*  MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
```

```
 77  FROMID          PIC X(4) VALUE 'AAUQ'.
 77  TOID            PIC X(4) VALUE 'ABRH'.
 77  MSTAT           PIC 9(5) COMP.

* SYSTEM USER COUNTER:

* INITTED FOR ONE USER TASK ONLY AT THIS TIME:
 77  ACTIVE-USERS    PIC 99    VALUE 1.


*  INPUT PARAMETERS BUF1 AND BUF2:

 01  BUF1.
     03  BCALLER         PIC 9.
     03  BFUNCTION       PIC X(5).
     03  BSTAT           PIC X(4).
     03  BFILE-NAME      PIC X(4).
     03  BREFER          PIC X(4).
     03  BLINKPATH       PIC X(8).
     03  BCTRL           PIC X(9).
     03  BVELEM          PIC X(60).
     03  BVAREA          PIC X(82).


 PROCEDURE DIVISION.


 INIT-ROUTINE.
* INITIALIZE MESSAGE SYSTEM COMMUNICATION:
 ESTAB-TASK-ID.
 CONNECT-INTO-SYSTEM.
     CALL 'CONN' USING
         TOID FROMID MSTAT.
```

```
     IF MSTAT NOT EQUAL ZERO
         DISPLAY 'BINT ABORT-- BAD CONNECT' UPON CRT
         STOP RUN.

*
******REPEAT
*



 MAIN-LOOP.
* RECEIVE REQUEST FROM USER:
 HELLO-THERE.
     CALL 'RECV' USING
         TOID  BUF1  MSTAT.
     IF MSTAT = 32 THEN DISPLAY
         'BINT: TIMED OUT-- WILL RECV AGAIN'
         UPON CRT
         GO TO HELLO-THERE
     ELSE
     IF MSTAT NOT EQUAL ZERO
         DISPLAY 'BINT ABORT-- BAD RECV' UPON CRT
         GO TO SYS-SINOF.


 BINT-UNPACK.
     MOVE BFUNCTION TO FUNCTION.
     MOVE BFILE-NAME TO FILE-NAME.
     MOVE BVAREA TO VAREA.

 BINT-EXECUTE-CALL.
     GO TO BV1, BV2, BV3, BV4, BV5, BV6
         DEPENDING ON BCALLER.
```

```
BINT-ERROR.
    MOVE 'XXXX' TO STAT.
    GO TO BINT-REPACK.

* DATBAS CALL ROUTINES:

BV1.
    MOVE BREFER TO REFER.
    MOVE BLINKPATH TO LINKPATH.
    MOVE BCTRL TO CTRL.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION  STAT  FILE-NAME  REFER
        LINKPATH  CTRL  VELEM  VAREA
        ENDP.
    MOVE REFER TO BREFER.
    GO TO BINT-REPACK.


BV2.
    MOVE BCTRL TO CTRL.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION  STAT  FILE-NAME  CTRL
        VELEM  VAREA  ENDP.
    GO TO BINT-REPACK.


BV3.
    MOVE BREFER TO REFER.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION  STAT  FILE-NAME  REFER
```

```
           VELEM  VAREA  ENDP.
       MOVE REFER TO BREFER.
       GO TO BINT-REPACK.


   BV4.
       MOVE BREFER TO REFER.
       CALL 'DATBAS' USING
           FUNCTION  STAT  REFER
           VAREA  ENDP.
       MOVE REFER TO BREFER.
       GO TO BINT-REPACK.


   BV5.
       MOVE BCTRL TO CTRL.
       CALL 'DATBAS'  USING
           FUNCTION  STAT  FILE-NAME
           CTRL  VAREA  ENDP.
       GO TO BINT-REPACK.


   BV6.
       CALL 'DATBAS'  USING
           FUNCTION  STAT  VAREA  ENDP.
   *   IF  FUNCTION = 'SINON'  AND  STAT = '****'
   *      * THEN ADD 1 TO ACTIVE-USERS.
       IF  FUNCTION = 'SINOF'  AND  STAT = '****'
           THEN SUBTRACT 1 FROM ACTIVE-USERS.
       GO TO BINT-REPACK.


   BINT-REPACK.
       MOVE STAT TO BSTAT.
```

```
     MOVE VAREA TO BVAREA.


* RETURN INFORMATION TO USER:

 GOOD-BYE.
     CALL 'SEND' USING
          TOID  BUF1  MSTAT.
     IF MSTAT = 29 THEN DISPLAY
          'BINT: TIMED OUT-- WILL SEND AGAIN'
          UPON CRT
          GO TO GOOD-BYE
     ELSE
     IF MSTAT NOT EQUAL ZERO
          DISPLAY 'BINT ABORT-- BAD SEND' UPON CRT
          GO TO SYS-SINOF.


* SEE IF ANY USERS STILL ON THE LINE:

     IF ACTIVE-USERS GREATER THAN ZERO
          THEN GO TO MAIN-LOOP.

*
******UNTIL  ACTIVE-USERS = 0.
*


* TERMINATION ROUTINE:

 SYS-SINOF.
     CALL 'DISC' USING    TOID    MSTAT.
     IF MSTAT NOT EQUAL ZERO
          DISPLAY 'BINT ABORT-- BAD DISCONNECT' UPON CRT.
```

```
STOP RUN.
```

APPENDIX H

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  BINT.
 REMARKS.  'BINT' IS THE INTERFACE PROGRAM THAT
     DIRECTLY ACCESSES THE BACKEND DBMS.  IT RECEIVES
     THE CALLING PARAMETERS VIA THE PACKED PARAMETER
     BUFFERS BUF1 , DETERMINES THE CALL
     DESIRED, AND EXECUTES THAT CALL.  THE RESULTS
     ARE REPACKED AND THE PARAMETERS RETURNED.
*
* THIS VERSION DESIGNED FOR 8/32 TO 8/32 INTERTASK COMMUNICATION
*

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. INTERDATA MODEL-8-32.
 OBJECT-COMPUTER.  INTERDATA MODEL-8-32.
 INPUT-OUTPUT SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

*  DATBAS PARAMETERS:

 77  FUNCTION      PIC X(5).
 77  STAT          PIC X(4).
 77  FILE-NAME     PIC X(4).
 77  REFER         PIC X(4).
 77  LINKPATH      PIC X(8).
 77  CTRL          PIC X(9).
 77  VELEM         PIC X(60).
 77  VAREA         PIC X(82).
 77  ENDP          PIC X(4)  VALUE 'END.'.

*  MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
```

```
 77  FROMID          PIC X(4) VALUE 'ABUQ'.
 77  TOID            PIC X(4) VALUE 'ABRD'.
 77  MSTAT           PIC 9(4) COMP.

* SYSTEM USER COUNTER:

* INITIED FOR ONE USER TASK ONLY AT THIS TIME:
 77  ACTIVE-USERS    PIC 99    VALUE 1.


*  INPUT PARAMETERS BUF1 AND BUF2:

 01  BUF1.
     03  BCALLER         PIC 9.
     03  BFUNCTION       PIC X(5).
     03  BSTAT           PIC X(4).
     03  BFILE-NAME      PIC X(4).
     03  BREFER          PIC X(4).
     03  BLINKPATH       PIC X(8).
     03  BCTRL           PIC X(9).
     03  BVELEM          PIC X(60).
     03  BVAREA          PIC X(82).


 PROCEDURE DIVISION.


 INIT-ROUTINE.
* INITIALIZE MESSAGE SYSTEM COMMUNICATION:
 ESTAB-TASK-ID.
 CONNECT-INTO-SYSTEM.
     CALL 'CONN' USING
          TOID FROMID MSTAT.
```

```
     IF MSTAT NOT EQUAL ZERO
         DISPLAY 'BINT ABORT-- BAD CONNECT' UPON CRT
         STOP RUN.
           \
*
******REPEAT
*



 MAIN-LOOP.
* RECEIVE REQUEST FROM USER:
 HELLO-THERE.
     CALL 'RECV' USING
         TOID  BUF1  MSTAT.
     IF MSTAT = 32 THEN DISPLAY
         'BINT: TIMED OUT-- WILL RECV AGAIN'
         UPON CRT
         GO TO HELLO-THERE
     ELSE
     IF MSTAT NOT EQUAL ZERO
         DISPLAY 'BINT ABORT-- BAD RECV' UPON CRT
         GO TO SYS-SINOF.


 BINT-UNPACK.
     MOVE BFUNCTION TO FUNCTION.
     MOVE BFILE-NAME TO FILE-NAME.
     MOVE BVAREA TO VAREA.

 BINT-EXECUTE-CALL.
     GO TO BV1, BV2, BV3, BV4, BV5, BV6
         DEPENDING ON BCALLER.
```

```
BINT-ERROR.
    MOVE 'XXXX' TO STAT.
    GO TO BINT-REPACK.


* DATBAS CALL ROUTINES:

BV1.
    MOVE BREFER TO REFER.
    MOVE BLINKPATH TO LINKPATH.
    MOVE BCTRL TO CTRL.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION  STAT  FILE-NAME  REFER
        LINKPATH  CTRL  VELEM  VAREA
        ENDP.
    MOVE REFER TO BREFER.
    GO TO BINT-REPACK.


BV2.
    MOVE BCTRL TO CTRL.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION  STAT  FILE-NAME  CTRL
        VELEM  VAREA  ENDP.
    GO TO BINT-REPACK.


BV3.
    MOVE BREFER TO REFER.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION  STAT  FILE-NAME  REFER
```

```
           VELEM  VAREA  ENDP.
       MOVE REFER TO BREFER.
       GO TO BINT-REPACK.


   BV4.
       MOVE BREFER TO REFER.
       CALL 'DATBAS' USING
           FUNCTION  STAT  REFER
           VAREA  ENDP.
       MOVE REFER TO BREFER.
       GO TO BINT-REPACK.


   BV5.
       MOVE BCTRL TO CTRL.
       CALL 'DATBAS'  USING
           FUNCTION  STAT  FILE-NAME
           CTRL  VAREA  ENDP.
       GO TO BINT-REPACK.


   BV6.
       CALL 'DATBAS'  USING
           FUNCTION  STAT  VAREA  ENDP.
   *   IF  FUNCTION = 'SINON'  AND  STAT = '****'
   *      * THEN ADD 1 TO ACTIVE-USERS.
       IF  FUNCTION = 'SINOF'  AND  STAT = '****'
           THEN SUBTRACT 1 FROM ACTIVE-USERS.
       GO TO BINT-REPACK.


   BINT-REPACK.
       MOVE STAT TO BSTAT.
```

```
     MOVE VAREA TO BVAREA.

* RETURN INFORMATION TO USER:

 GOOD-BYE.
     CALL 'SEND' USING
          TOID  BUF1  MSTAT.
     IF MSTAT = 29 THEN DISPLAY
         'BINT: TIMED OUT-- WILL SEND AGAIN'
         UPON CRT
         GO TO GOOD-BYE
     ELSE
     IF MSTAT NOT EQUAL ZERO
         DISPLAY 'BINT ABORT-- BAD SEND' UPON CRT
         GO TO SYS-SINOF.

* SEE IF ANY USERS STILL ON THE LINE:

     IF ACTIVE-USERS GREATER THAN ZERO
         THEN GO TO MAIN-LOOP.

*
******UNTIL  ACTIVE-USERS = 0.
*

* TERMINATION ROUTINE:

 SYS-SINOF.
     CALL 'DISC' USING    TOID    MSTAT.
     IF MSTAT NOT EQUAL ZERO
         DISPLAY 'BINT ABORT-- BAD DISCONNECT' UPON CRT.
```

```
STOP RUN.
```

APPENDIX I

```
//COB EXEC COB3CL
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  BINT.
       REMARKS.  'BINT' IS THE INTERFACE PROGRAM THAT
           DIRECTLY ACCESSES THE BACKEND DBMS.  IT RECEIVES
           THE CALLING PARAMETERS VIA THE PACKED PARAMETER
           BUFFERS BUF1 AND BUF2, DETERRMINES THE CALL
           DESIRED, AND EXECUTES THAT CALL.  THE RESULTS
           ARE THEN REPACKED AND THE PARAMETERS RETURNED.


       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  IBM-370.
       OBJECT-COMPUTER.  IBM-370.
       INPUT-OUTPUT SECTION.
       DATA DIVISION.
        WORKING-STORAGE SECTION.
      *
      *  DATBAS PARAMETERS:
      *
        77  FUNCTION              PIC X(5).
        77  STAT                  PIC X(4).
        77  FILE-NAME             PIC X(4).
        77  REFER                 PIC X(4).
        77  LINKPATH              PIC X(8).
        77  CTRL                  PIC X(9).
        77  VELEM                 PIC X(60).
        77  VAREA                 PIC X(82).
        77  ENDP                  PIC X(4) VALUE 'END,'.
       *
       *  MESSAGE SYSTEM PARAMETERS:
       *
```

```
 77  TOID                    PIC 9(9) COMP-3 VALUE 414254500.
 77  FROMID                  PIC 9(9) COMP-3 VALUE 424104010.
 77  MSMODE                  PIC 9(8) COMP SYNC VALUE 0.
 77  OUTCMD                  PIC X(128).
 77  OUTCMDLTH               PIC 9(8) COMP SYNC VALUE 0.
 77  INCMD                   PIC X(128).
 77  INCMDLTH                PIC 9(8) COMP SYNC.
 77  MSTAT                   PIC 9(8) COMP SYNC.
 77  ECB                     PIC 9(8) COMP SYNC.
 77  BUF1LTH                 PIC 9(8) COMP SYNC.
 77  BUF2LTH                 PIC 9(8) COMP SYNC VALUE 0.
*
*  SYSTEM USER COUNTER:
*  INITTED FOR ONE USER TASK ONLY AT THIS TIME.
*
 77  ACTIVE-USERS            PIC 99 VALUE 1.
*
*  INPUT PARAMETERS BUF1 AND BUF2:
*
 01  BUF1.
     03  BCALLER             PIC 9.
     03  BFUNCTION           PIC X(5).
     03  BSTAT               PIC X(4).
     03  BFILE-NAME          PIC X(4).
     03  BREFER              PIC X(4).
     03  BLINKPATH           PIC X(8).
     03  BCTRL               PIC X(9).
     03  BVELEM              PIC X(60).
     03  BAREA               PIC X(82).
 01  BUF2.
     02  DUMMY-ELEMENT       PIC X(1).

 PROCEDURE DIVISION.
```

```
     INIT-ROUTINE.
*
*    INITIALIZE MESSAGE SYSTEM COMMUNICATION.
*
 ESTAB-TASK-ID.
      CALL 'MSWNTSK' USING FROMID MSTAT.
      IF MSTAT NOT EQUAL ZERO
          DISPLAY 'BINT ABORT-- BAD TASK ID' UPON CONSOLE
          STOP RUN.

  CONNECT-INTO-SYSTEM.
      CALL 'MSCONN' USING
          FROMID TOID MSMODE OUTCMD OUTCMDLTH
          INCMD INCMDLTH MSTAT ECB.
          CALL 'MSCHECK' USING ECB.
      IF MSTAT NOT EQUAL ZERO
          DISPLAY 'BINT ABORT-- BAD CONNECT' UPON CONSOLE
          CALL 'MSPURGE' USING FROMID FROMID MSMODE MSTAT ECB
          STOP RUN.

  MAIN-LOOP.
*
*  RECEIVE REQUEST FROM USER;
*
  HELLO-THERE.
      MOVE 177 TO BUF1LTH.
      CALL 'MSRECV' USING
          MSMODE FROMID TOID BUF2 BUF2LTH
          BUF1 BUF1LTH MSTAT ECB.
          CALL 'MSCHECK' USING ECB.
      IF MSTAT = 32 THEN
          DISPLAY 'BINT:  TIMED OUT-- WILL RECV AGAIN'
          UPON CONSOLE
          GO TO HELLO-THERE
```

```
    ELSE
    IF MSTAT NOT EQUAL ZERO
        DISPLAY 'BINT ABORT--BAD RECEIVE' UPON CONSOLE
        GO TO SYS-SINOF.

BINT-UNPACK.
    CALL 'TRANSE' USING BUF1 BUF1LTH.
    MOVE BFUNCTION TO FUNCTION.
    MOVE BFILE-NAME TO FILE-NAME.
    MOVE BVAREA TO VAREA.

BINT-EXECUTE-CALL.
    GO TO BV1, BV2, BV3, BV4, BV5, BV6, BV7, BV8
    DEPENDING ON BCALLER.

BINT-ERROR.
    MOVE 'XXXX' TO STAT.
    GO TO BINT-REPACK.
*
*    DATBAS CALL ROUTINES.
*
BV1.
    MOVE BREFER TO REFER.
    MOVE BLINKPATH TO LINKPATH.
    MOVE BCTRL TO CTRL.
    MOVE BVELEM TO VELEM.
    CALL 'DATBAS' USING
        FUNCTION STAT FILE-NAME REFER
        LINKPATH CTRL VELEM VAREA
        ENDP.
    MOVE REFER TO BREFER.
    GO TO BINT-REPACK.

BV2.
```

```
           MOVE BCTRL TO CTRL.
           MOVE BVELEM TO VELEM.
           CALL 'DATBAS' USING
                FUNCTION STAT FILE-NAME CTRL
                VELEM VAREA ENDP.
           GO TO BINT-REPACK.

       BV3.
           MOVE BREFER TO REFER.
           MOVE BVELEM TO VELEM.
           CALL 'DATBAS' USING
                FUNCTION STAT FILE-NAME REFER
                VELEM VAREA ENDP.
           MOVE REFER TO BREFER.
           GO TO BINT-REPACK.

       BV4.
           MOVE BREFER TO REFER.
           CALL 'DATBAS' USING
                FUNCTION STAT REFER
                VAREA ENDP.
           MOVE REFER TO BREFER.
           GO TO BINT-REPACK.

       BV5.
           MOVE BCTRL TO CTRL.
           CALL 'DATBAS' USING
                FUNCTION STAT FILE-NAME
                CTRL VAREA ENDP.
           GO TO BINT-REPACK.

       BV6.
           MOVE BLINKPATH TO LINKPATH.
           MOVE BCTRL TO CTRL.
```

```
           CALL 'DATBAS' USING
               FUNCTION STAT CTRL LINKPATH ENDP.
      *
      *    ADD 1 TO ACTIVE-USERS.
      *
           GO TO BINT-REPACK.

    BV7.
        MOVE BLINKPATH TO LINKPATH.
        CALL 'DATBAS' USING FUNCTION STAT LINKPATH ENDP.
        SUBTRACT 1 FROM ACTIVE-USERS.
        GO TO BINT-REPACK.

    BV8.
        CALL 'DATBAS' USING FUNCTION STAT
             FILE-NAME ENDP.
        GO TO BINT-REPACK.

       BINT-REPACK.
           MOVE STAT TO BSTAT.
           MOVE VAREA TO BVAREA.
      *
      *    RETURN INFORMATION TO USER.
      *
       GOOD-BYE.
      *
      *    ESTABLISH LENGTH OF BUFFERS.
      *
           MOVE 177 TO BUF1LGTH.
           CALL 'TRANSA' USING BUF1 BUF1LTH.
           CALL 'MSSEND' USING
               MSMODE FROMID TOID BUF2 BUF2LTH
               BUF1 BUF1LTH MSTAT ECB.
           CALL 'MSCHECK' USING ECB.
```

```
            IF MSTAT = 29 THEN DISPLAY
                'BINT:  TIMED OUT-- WILL SEND AGAIN,'
                UPON CONSOLE
                GO TO GOOD-BYE
            ELSE
            IF MSTAT NOT EQUAL ZERO
                DISPLAY 'BINT ABORT-- BAD SEND' UPON CONSOLE
                GO TO SYS-SINOF.
      *
      *     SEE IF ANY USERS ARE STILL ON THE LINE.
      *
            IF ACTIVE-USERS GREATER THAN ZERO
            THEN GO TO MAIN-LOOP.
      *
      *     TERMINATION ROUTINE:
      *
       SYS-SINOF.
            CALL 'MSDISC' USING FROMID TOID MSMODE MSTAT ECB.
            CALL 'MSCHECK' USING ECB.
            IF MSTAT NOT EQUAL ZERO
                DISPLAY 'BINT ABORT-- BAD DISCONNECT' UPON CONSOLE.
            CALL 'MSPURGE' USING FROMID FROMID MSTAT ECB.
            STOP RUN.
//LKED.SYSLMOD DD DSN=DS2L1,LOADLIB,DISP=OLD
//LKED.TOTLIB DD DSN=DS2L1.TOTAL,LINKLIB,DISP=SHR
//LKED.MSLIB DD DSN=DSQ76.TXTLIB,DISP=SHR
//LKED.SYSIN DD *
   INCLUDE TOTLIB(DATBAS4)
   INCLUDE TOTLIB(TRANS)
   INCLUDE MSLIB(MSCALL,SYSCALL)
   LIBRARY *(DATBASXT)
   NAME BINT(R)
```

****
*   *
* APPENDIX J *
*   *
****

```
$IFNULL @1;$COPY
*  CALLING FORMAT:  COBESTAB INTERFACE NAME,APPLICATION NAME,
*  ASSEMBLER INTERFACE NAME,          *
$NO
$EXIT;$ENDC
$IFNX @1.OBJ;$COPY
*   FILE @1.OBJ DOES NOT EXIST
$NO
$CLEAR ;$EXIT ;$ENDC
$IFX SYS2:ESTAB.CMD  ;DEL SYS2:ESTAB.CMD  ;$ENDC
$IFNX @1.TCM
$BUILD SYS2:ESTAB.CMD
ES TA
IN @1.OBJ
IN @2.OBJ
EDIT SYS2:CBU200.LIB/S
IN @3.OBJ
BU TA,@1.TSK
AMAP
END
$ENDB
$ENDC
$IFX  @1.TSK;DE @1.TSK;$ENDC
$IFNX SYS2:ESTAB.SCR ;AL SYS2:ESTAB.SCR,IN,126/4 ;$ENDC
$IFX SYS2:ESTAB.LST  ;DEL SYS2:ESTAB.LST  ;$ENDC
AL SYS2:ESTAB.LST,IN,126/4
LO TET32
AS 1,@1.OBJ
AS 3,PR:
TE 4,IN,126
$IFX @1.TCM;AS 5,@1.TCM;$ENDC
$IFNX @1.TCM;AS 5,SYS2:ESTAB.CMD;$ENDC
AS 7,CON:
ST
```

```
$EXIT
```

****
*   *
* APPENDIX K *
*   *
****

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HINI.
REMARKS.  'HINT' IS THE HOST INTERFACE DBMS PROGRAM.
     THE DATBAS PARAMETERS ARE PACKED INTO  DATA
     BUFFERS: BUF1 , WHICH ARE THEN PASSED
     TO 'BINI', THE BACKEND DBMS INTERFACE.
     'BINT' EXECUTES THE PHYSICAL CALL AND RETURNS THE RESULTS.

*
* THIS VERSION DESIGNED FOR 8/32 TO 7/32 INTERTASK COMMUNICATION
*

     ACTIVE DATBAS PARAMETERS  USED BY THE
        PROGRAM ARE:

          FUNCTION
          STAT
          FILE-NAME
          REFER
          LINKPATH
          CTRL-FIELD
          VELEMENTS
          VAREA.


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  INTERDATA MODEL-8-32.
OBJECT-COMPUTER.  INTERDATA MODEL-8-32.
INPUT-OUTPUT SECTION.



DATA DIVISION.
```

```
 WORKING-STORAGE SECTION.

* MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
 77  FROMID              PIC X(4) VALUE 'ABRH'.
 77  TOID                PIC X(4) VALUE 'AAUQ'.
 77  MSTAT               PIC 9(4) USAGE COMP.


*  PARAMETER BUFFER FOR INTERFACE CALL

 01  BUF1.
     03  CALLER      PIC 9.
     03  FUNCTION    PIC X(5).
     03  STAT        PIC X(4).
     03  FILE-NAME   PIC X(4).
     03  REFER       PIC X(4).
     03  LINKPATH    PIC X(8).
     03  CTRL-FIELD  PIC X(9).
     03  VELEMENTS   PIC X(60).

     03  VAREA        PIC X(82).


 LINKAGE SECTION.

* INPUT PARAMETERS

 77  BFUNCTION           PIC X(5).
 77  BSTAT               PIC X(4).
 77  BFILE-NAME          PIC X(4).
 77  BREFER              PIC X(4).
 77  BLINKPATH           PIC X(8).
 77  BCTRL               PIC X(9).
 77  BVELEM              PIC X(60).
```

```
 77  BVAREA              PIC X(82).
 77  BEND                PIC X(4).

 PROCEDURE DIVISION USING
     BFUNCTION  BSTAT  BFILE-NAME  BREFER
     BLINKPATH  BCTRL  BVELEM  BVAREA  BEND.

*  MOVE PARAMETER ELEMENTS INTO BUFFER
     MOVE BFUNCTION TO FUNCTION.
     MOVE BFILE-NAME TO FILE-NAME.
     MOVE BVAREA TO VAREA.

 HINT-CALL-SELECT.
*  DETERMINE TYPE OF CALL

     IF FUNCTION = 'READV' OR
         FUNCTION = 'READR' OR
         FUNCTION = 'READD' OR
         FUNCTION = 'WRITV' OR
         FUNCTION = 'DELVD' OR
         FUNCTION = 'ADDVC' OR
         FUNCTION = 'ADDVA' OR
         FUNCTION = 'ADDVB'
         THEN GO TO CALL-HINT-V001

     ELSE IF FUNCTION = 'READM' OR
         FUNCTION = 'WRITM' OR
         FUNCTION = 'ADD-M' OR
         FUNCTION = 'DEL-M'
         THEN GO TO CALL-HINT-V002

     ELSE IF FUNCTION = 'RDNXT'
         THEN GO TO CALL-HINT-V003
```

```
      ELSE IF FUNCTION = 'MARKL' OR
          FUNCTION = 'QUIET'
          THEN GO TO CALL-HINT-V004

      ELSE IF FUNCTION = 'RQLOC'
          THEN GO TO CALL-HINT-V005

      ELSE IF FUNCTION = 'SINON' OR
          FUNCTION = 'SINOF'
          THEN GO TO CALL-HINT-V006

* ELSE AN INVALID CALL:
      ELSE MOVE 'XXXX' TO STAT
          GO TO HINT-EXIT.


**********************
*
*  CALL-HINT-V001:
*     READV
*     READR
*     READD
*     WRITV
*     DELVD
*     ADDVC
*     ADDVA
*     ADDVB
*
**********************

 CALL-HINT-V001.
      MOVE 1 TO CALLER.
      MOVE BREFER TO REFER.
```

```
      MOVE BLINKPATH TO LINKPATH.
      MOVE BCTRL TO CTRL-FIELD.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.


**********************
*
*   CALL-HINT-V002:
*      READM
*      WRITM
*      ADD-M
*      DEL-M
*
**********************

 CALL-HINT-V002.
      MOVE 2 TO CALLER.
      MOVE BCTRL TO CTRL-FIELD.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V003:
*      RUNXT
*
**********************
```

```
 CALL-HINT-V003.
      MOVE 3 TO CALLER.
      MOVE BREFER TO REFER.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V004:
*      MARKL
*      QUIET
*
**********************

 CALL-HINT-V004.
      MOVE 4 TO CALLER.
      MOVE BREFER TO REFER.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V005:
*      RGLOC
*
**********************
```

```
 CALL-HINT-V005.
      MOVE 5 TO CALLER.
      MOVE BCTRL TO CTRL-FIELD.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V006:
*     SINON
*     SINOF
*
**********************

 CALL-HINT-V006.
      MOVE 6 TO CALLER.
      IF FUNCTION = 'SINON'
          PERFORM SYS-SINON THRU SYS-SINON-EXIT.
      PERFORM CALL-BINT THRU CALL-EXIT.
      IF FUNCTION = 'SINOF'
          PERFORM SYS-SINOF THRU SYS-SINOF-EXIT.
      GO TO HINT-RETURN.

 HINT-RETURN.
* RETRANSFER PARAMETER VALUES
      MOVE STAT TO BSTAT.
      MOVE VAREA TO BVAREA.
 HINT-EXIT.
      EXIT PROGRAM.


* SUBROUTINES FOR MESSAGE SYSTEM COMMUNICATION:
```

```
 SYS-SINON.
* CONNECT INTO MESSAGE SYSTEM:
 CONNECT-INTO-SYSTEM.
     CALL 'CONN' USING
         TOID  FROMID  MSTAT.
     IF MSTAT NOT EQUAL ZERO
         PERFORM MSYSERROR
         GO TO HINT-EXIT.
 SYS-SINON-EXIT.  EXIT.


 SYS-SINOF.
* DISCONNECT FROM SYSTEM:
 SYS-DISCONNECT.
     CALL 'DISC' USING     TOID     MSTAT.
 SYS-SINOF-EXIT.  EXIT.


 CALL-BINT.
* CALL THE MESSAGE SYSTEM FOR BUFFER IO:
 CALL-BINT1.
     CALL 'SEND' USING    TOID  BUF1
          MSTAT.
     IF MSTAT = 29 THEN DISPLAY
         'HINT: TIMED OUT-- WILL SEND AGAIN'
         UPON CRT
         GO TO CALL-BINT1
     ELSE
     IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR  GO TO HINT-EXIT.
 CALL-BINT2.
* RECEIVE REPLY FROM BACK END:
     CALL 'RECV' USING    TOID  BUF1
```

```
          MSTAT.
      IF MSTAT = 32 THEN DISPLAY
          'HINT: TIMED OUT-- WILL RECV AGAIN'
          UPON CRT
          GO TO CALL-BINT2
      ELSE
      IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR  GO TO HINT-EXIT.
 CALL-EXIT.  EXIT.

 MSYSERROR.
*  FATAL MESSAGE SYSTEM ERROR:
      DISPLAY 'HINT: MESSAGE SYS ERROR-- REQUEST ABORTED' UPON CRT.
      MOVE 'XXXX' TO STAT.
```

****
*   *
* APPENDIX L *
*   *
****

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HINT.
REMARKS.  'HINT' IS THE HOST INTERFACE DBMS PROGRAM.
     THE DATBAS PARAMETERS ARE PACKED INTO  DATA
     BUFFERS: BUF1 , WHICH ARE THEN PASSED
     TO 'BINT', THE BACKEND DBMS INTERFACE.
     'BINT' EXECUTES THE PHYSICAL CALL AND RETURNS THE RESULTS.

*
* THIS VERSION DESIGNED FOR 8/32 TO 8/32 INTERTASK COMMUNICATION
*

     ACTIVE DATBAS PARAMETERS  USED BY THE
        PROGRAM ARE:

         FUNCTION
         STAT
         FILE-NAME
         REFER
         LINKPATH
         CTRL-FIELD
         VELEMENTS
         VAREA.


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  INTERDATA MODEL-8-32.
OBJECT-COMPUTER.  INTERDATA MODEL-8-32.
INPUT-OUTPUT SECTION.



DATA DIVISION.
```

```
 WORKING-STORAGE SECTION.

* MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
 77  FROMID              PIC X(4) VALUE 'ABRD'.
 77  TOID                PIC X(4) VALUE 'ABUQ'.
 77  MSTAT               PIC 9(4) USAGE COMP.


*  PARAMETER BUFFER FOR INTERFACE CALL

 01  BUF1.
     05  CALLER      PIC 9.
     05  FUNCTION    PIC X(5).
     05  STAT        PIC X(4).
     05  FILE-NAME   PIC X(4).
     05  REFER       PIC X(4).
     05  LINKPATH    PIC X(8).
     05  CTRL-FIELD  PIC X(9).
     05  VELEMENTS   PIC X(60).

     05  VAREA       PIC X(82).


 LINKAGE SECTION.

* INPUT PARAMETERS

 77  BFUNCTION           PIC X(5).
 77  BSTAT               PIC X(4).
 77  BFILE-NAME          PIC X(4).
 77  BREFER              PIC X(4).
 77  BLINKPATH           PIC X(8).
 77  BCTRL               PIC X(9).
 77  BVELEM              PIC X(60).
```

```
 77  BVAREA              PIC X(82).
 77  BEND                PIC X(4).


 PROCEDURE DIVISION USING
     BFUNCTION  BSTAT  BFILE-NAME  BREFER
     BLINKPATH  BCTRL  BVELEM  BVAREA  BEND.

*  MOVE PARAMETER ELEMENTS INTO BUFFER
     MOVE BFUNCTION TO FUNCTION.
     MOVE BFILE-NAME TO FILE-NAME.
     MOVE BVAREA TO VAREA.

 HINT-CALL-SELECT.
*  DETERMINE TYPE OF CALL

      IF FUNCTION = 'READV' OR
          FUNCTION = 'READR' OR
          FUNCTION = 'READD' OR
          FUNCTION = 'WRITV' OR
          FUNCTION = 'DELVD' OR
          FUNCTION = 'ADDVC' OR
          FUNCTION = 'ADDVA' OR
          FUNCTION = 'ADDVB'
          THEN GO TO CALL-HINT-V001

      ELSE IF FUNCTION = 'READM' OR
          FUNCTION = 'WRITM' OR
          FUNCTION = 'ADD-M' OR
          FUNCTION = 'DEL-M'
          THEN GO TO CALL-HINT-V002

      ELSE IF FUNCTION = 'RDNXT'
          THEN GO TO CALL-HINT-V003
```

```
           ELSE IF FUNCTION = 'MARKL' OR
                FUNCTION = 'QUIET'
                THEN GO TO CALL-HINT-V004

           ELSE IF FUNCTION = 'RQLOC'
                THEN GO TO CALL-HINT-V005

           ELSE IF FUNCTION = 'SINON' OR
                FUNCTION = 'SINOF'
                THEN GO TO CALL-HINT-V006

      * ELSE AN INVALID CALL:
           ELSE MOVE 'XXXX' TO STAT
                GO TO HINT-EXIT.


      **********************
      *
      *  CALL-HINT-V001:
      *     READV
      *     RLADR
      *     READD
      *     WRITV
      *     DELVD
      *     ADDVC
      *     ADDVA
      *     ADDVB
      *
      **********************

       CALL-HINT-V001.
           MOVE 1 TO CALLER.
           MOVE BREFER TO REFER.
```

```
     MOVE BLINKPATH TO LINKPATH.
     MOVE BCTRL TO CTRL-FIELD.
     MOVE BVELEM TO VELEMENTS.
     PERFORM CALL-BINT THRU CALL-EXIT.
     MOVE REFER TO BREFER.
     GO TO HINT-RETURN.


**********************
*
*  CALL-HINT-V002:
*     READM
*     WRITM
*     ADD-M
*     DEL-M
*
**********************

 CALL-HINT-V002.
     MOVE 2 TO CALLER.
     MOVE BCTRL TO CTRL-FIELD.
     MOVE BVELEM TO VELEMENTS.
     PERFORM CALL-BINT THRU CALL-EXIT.
     GO TO HINT-RETURN.



**********************
*
*  CALL-HINT-V003:
*     RDNXT
*
**********************
```

```
 CALL-HINT-V003.
      MOVE 3 TO CALLER.
      MOVE BREFER TO REFER.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V004:
*      MARKL
*      QUICI
*
**********************

 CALL-HINT-V004.
      MOVE 4 TO CALLER.
      MOVE BREFER TO REFER.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V005:
*      RGLOC
*
**********************
```

```
 CALL-HINT-V005.
      MOVE 5 TO CALLER.
      MOVE BCTRL TO CTRL-FIELD.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V006:
*      SINON
*      SINOF
*
**********************

 CALL-HINT-V006.
      MOVE 6 TO CALLER.
      IF FUNCTION = 'SINON'
          PERFORM SYS-SINON THRU SYS-SINON-EXIT.
      PERFORM CALL-BINT THRU CALL-EXIT.
      IF FUNCTION = 'SINOF'
          PERFORM SYS-SINOF THRU SYS-SINOF-EXIT.
      GO TO HINT-RETURN.

 HINT-RETURN.
* RETRANSFER PARAMETER VALUES
      MOVE STAT TO BSTAT.
      MOVE VAREA TO BVAREA.
 HINT-EXIT.
      EXIT PROGRAM.


* SUBROUTINES FOR MESSAGE SYSTEM COMMUNICATION:
```

```
 SYS-SINON.
* CONNECT INTO MESSAGE SYSTEM:
 CONNECT-INTO-SYSTEM.
     CALL 'CONN' USING
         TOID  FROMID  MSTAT.
     IF MSTAT NOT EQUAL ZERO
         PERFORM MSYSERROR
         GO TO HINT-EXIT.
 SYS-SINON-EXIT.  EXIT.


 SYS-SINOF.
* DISCONNECT FROM SYSTEM:
 SYS-DISCONNECT.
     CALL 'DISC' USING    TOID    MSTAT.
     IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR.
 SYS-SINOF-EXIT.  EXIT.


 CALL-BINT.
* CALL THE MESSAGE SYSTEM FOR BUFFER 10:
 CALL-BINT1.
     CALL 'SEND' USING   TOID  BUF1
          MSTAT.
     IF MSTAT = 29 THEN DISPLAY
         'HINT: TIMED OUT-- WILL SEND AGAIN'
         UPON CRT
         GO TO CALL-BINT1
     ELSE
     IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR  GO TO HINT-EXIT.
 CALL-BINT2.
* RECEIVE REPLY FROM BACK END:
```

```
     CALL 'RECV' USING    TOID  BUF1
          MSTAT.
     IF MSTAT = 32 THEN DISPLAY
          'HINT: TIMED OUT-- WILL RECV AGAIN'
          UPON CRT
          GO TO CALL-BINT2
     ELSE
     IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR  GO TO HINT-EXIT.
CALL-EXIT.  EXIT.

MSYSERROR.
* FATAL MESSAGE SYSTEM ERROR:
     DISPLAY 'HINT: MESSAGE SYS ERROR-- REQUEST ABORTED' UPON CRT.
     MOVE 'XXXX' TO STAT.
```

APPENDIX M

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HINT.
REMARKS.  'HINT' IS THE HOST INTERFACE DBMS PROGRAM.
     THE DATBAS PARAMETERS ARE PACKED INTO  DATA
     BUFFERS: BUF1 , WHICH ARE THEN PASSED
     TO 'BINT', THE BACKEND DBMS INTERFACE.
     'BINT' EXECUTES THE PHYSICAL CALL AND RETURNS THE RESULTS.

*
* THIS VERSION DESIGNED FOR 8/32 TO 370 INTERTASK COMMUNICATION
*

     ACTIVE DATBAS PARAMETERS  USED BY THE
        PROGRAM ARE:

          FUNCTION
          STAT
          FILE-NAME
          REFER
          LINKPATH
          CTRL-FIELD
          VELEMENTS
          VAREA.


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  INTERDATA MODEL-8-32.
OBJECT-COMPUTER.  INTERDATA MODEL-8-32.
INPUT-OUTPUT SECTION.



DATA DIVISION.
```

```
 WORKING-STORAGE SECTION.

* MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
 77  FROMID              PIC X(4) VALUE 'ABTP'.
 77  MSTAT               PIC 9(4) USAGE COMP.                    PW1APR05
 77  TOID                PIC 9(9) COMP-3 VALUE 424104010.


*  PARAMETER BUFFER FOR INTERFACE CALL

 01  BUF1.
     05  CALLER      PIC 9.
     05  FUNCTION    PIC X(5).
     05  STAT        PIC X(4).
     05  FILE-NAME   PIC X(4).
     05  REFER       PIC X(4).
     05  LINKPATH    PIC X(8).
     05  CTRL-FIELD  PIC X(9).
     05  VELEMENTS   PIC X(60).

     05  VAREA        PIC X(82).


 LINKAGE SECTION.

* INPUT PARAMETERS

 77  BFUNCTION           PIC X(5).
 77  BSTAT               PIC X(4).
 77  BFILE-NAME          PIC X(4).
 77  BREFER              PIC X(4).
 77  BLINKPATH           PIC X(8).
 77  BCTRL               PIC X(9).
 77  BVELEM              PIC X(60).
```

```
 77  BVAREA              PIC X(82).
 77  BEND                PIC X(4).


 PROCEDURE DIVISION USING
     BFUNCTION  BSTAT  BFILE-NAME  BREFER
     BLINKPATH  BCTRL  BVELEM  BVAREA  BEND.

*  MOVE PARAMETER ELEMENTS INTO BUFFER
     MOVE BFUNCTION TO FUNCTION.
     MOVE BFILE-NAME TO FILE-NAME.
     MOVE BVAREA TO VAREA.

 HINT-CALL-SELECT.
*  DETERMINE TYPE OF CALL

     IF FUNCTION = 'READV' OR
         FUNCTION = 'READR' OR
         FUNCTION = 'READD' OR
         FUNCTION = 'WRITV' OR
         FUNCTION = 'DELVD' OR
         FUNCTION = 'ADDVC' OR
         FUNCTION = 'ADDVA' OR
         FUNCTION = 'ADDVB'
         THEN GO TO CALL-HINT-V001

     ELSE IF FUNCTION = 'READM' OR
         FUNCTION = 'WRITM' OR
         FUNCTION = 'ADD-M' OR
         FUNCTION = 'DEL-M'
         THEN GO TO CALL-HINT-V002

     ELSE IF FUNCTION = 'RDNXT'
         THEN GO TO CALL-HINT-V003
```

```
      ELSE IF FUNCTION = 'MARKL' OR
          FUNCTION = 'QUIET'
          THEN GO TO CALL-HINT-V004

      ELSE IF FUNCTION = 'RQLOC'
          THEN GO TO CALL-HINT-V005

      ELSE IF FUNCTION = 'TOTAL'
          THEN GO TO CALL-HINT-V006

      ELSE IF FUNCTION = 'DEQUE'
          THEN GO TO CALL-HINT-V007

      ELSE IF FUNCTION = 'OPENM' OR
          FUNCTION = 'CLOSM'
          THEN GO TO CALL-HINT-V008

* ELSE AN INVALID CALL:
      ELSE MOVE 'XXXX' TO STAT
          GO TO HINT-EXIT.


**********************
*
*  CALL-HINT-V001:
*     READV
*     READR
*     READD
*     WRITV
*     DELVD
*     ADDVC
*     ADDVA
*     ADDVB
```

```
*
**********************

 CALL-HINT-V001.
      MOVE 1 TO CALLER.
      MOVE BREFER TO REFER.
      MOVE BLINKPATH TO LINKPATH.
      MOVE BCTRL TO CTRL-FIELD.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.


**********************
*
*  CALL-HINT-V002:
*     READM
*     WRITM
*     ADD-M
*     DEL-M
*
**********************

 CALL-HINT-V002.
      MOVE 2 TO CALLER.
      MOVE BCTRL TO CTRL-FIELD.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.



**********************
```

```
*
*   CALL-HINT-V003:
*      RDNXT
*
**********************

 CALL-HINT-V003.
      MOVE 3 TO CALLER.
      MOVE BREFER TO REFER.
      MOVE BVELEM TO VELEMENTS.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V004:
*      MARKL
*      QUIET
*
**********************

 CALL-HINT-V004.
      MOVE 4 TO CALLER.
      MOVE BREFER TO REFER.
      PERFORM CALL-BINT THRU CALL-EXIT.
      MOVE REFER TO BREFER.
      GO TO HINT-RETURN.



**********************
```

```
*
*  CALL-HINT-V005:
*     KGLOC
*
**********************

 CALL-HINT-V005.
      MOVE 5 TO CALLER.
      MOVE BCTRL TO CTRL-FIELD.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.



**********************
*
*  CALL-HINT-V006:
*     TOTAL
*
**********************

 CALL-HINT-V006.
      MOVE 6 TO CALLER.
      MOVE BLINKPATH TO LINKPATH.
      MOVE BCTRL TO CTRL-FIELD.
      PERFORM SYS-SINON THRU SYS-SINON-EXIT.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.



**********************
*
*  CALL-HINT-V007:
```

```
*     DEQUE
*
**********************

 CALL-HINT-V007.
      MOVE 7 TO CALLER.
      MOVE BLINKPATH TO LINKPATH.
      PERFORM CALL-BINT THRU CALL-EXIT.
      PERFORM SYS-SINOF THRU SYS-SINOF-EXIT.
      GO TO HINT-RETURN.



**********************
*
*   CALL-HINT-V008:
*     OPENM
*     CLOSM
*
**********************

 CALL-HINT-V008.
      MOVE 8 TO CALLER.
      PERFORM CALL-BINT THRU CALL-EXIT.
      GO TO HINT-RETURN.

 HINT-RETURN.
* RETRANSFER PARAMETER VALUES
      MOVE STAT TO BSTAT.
      MOVE VAREA TO BVAREA.
 HINT-EXIT.
      EXIT PROGRAM.
```

```
* SUBROUTINES FOR MESSAGE SYSTEM COMMUNICATION:


 SYS-SINON.
* CONNECT INTO MESSAGE SYSTEM:
 CONNECT-INTO-SYSTEM.
     CALL 'CONN' USING
          TOID  FROMID  MSTAT.
     IF MSTAT NOT EQUAL ZERO
          PERFORM MSYSERROR
          GO TO HINT-EXIT.
 SYS-SINON-EXIT.  EXIT.


 SYS-SINOF.
* DISCONNECT FROM SYSTEM:
 SYS-DISCONNECT.
     CALL 'DISC' USING    TOID    MSTAT.
 SYS-SINOF-EXIT.  EXIT.


 CALL-BINT.
* CALL THE MESSAGE SYSTEM FOR BUFFER IO:
 CALL-BINT1.
     CALL 'SEND' USING   TOID  BUF1
          MSTAT.
     IF MSTAT = 29 THEN DISPLAY
          'HINT: TIMED OUT-- WILL SEND AGAIN'
          UPON CRT
          GO TO CALL-BINT1
     ELSE
     IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR  GO TO HINT-EXIT.
 CALL-BINT2.
* RECEIVE REPLY FROM BACK END:
```

```
     CALL 'RECV' USING    TOID   BUF1
          MSTAT.
     IF MSTAT = 52 THEN DISPLAY
          'HINT: TIMED OUT-- WILL RECV AGAIN'
          UPON CRT
          GO TO CALL-BINT2
     ELSE
     IF MSTAT NOT EQUAL ZERO PERFORM MSYSERROR  GO TO HINT-EXIT.
 CALL-EXIT.  EXIT.

 MSYSERROR.
* FATAL MESSAGE SYSTEM ERROR:
     DISPLAY 'HINT: MESSAGE SYS ERROR-- REQUEST ABORTED' UPON CRT.
     MOVE 'XXXX' TO STAT.
```

APPENDIX N

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     LOADPEOP.
 REMARKS.  'LOADPEOP' IS THE APPLICATION PROGRAM WHICH ACCESSES
     THE DATA BASE PERSON ON THE INTERDATA 7/32 THROUGH THE
     DISTRIBUTED DATA BASE MANAGEMENT SYSTEM.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. INTERDATA MODEL-8-32.
 OBJECT-COMPUTER. INTERDATA MODEL-8-32.
 SPECIAL-NAMES.
     C01 IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT PRINT-OUT  ASSIGN TO LU02-PRINTER.
 DATA DIVISION.
 FILE SECTION.
 FD  PRINT-OUT
     DATA RECORD IS PEOP-PERS-OUT.
 01  PEOP-PERS-OUT.
     05 CARRIAGECON          PIC X.
     05 INDATA               PIC X(132).
 WORKING-STORAGE SECTION.
*
*    TOTAL PARAMETERS AND I/O RECORD STRUCTURES.
*
 77 FUNCTION                 PIC X(5).
 77 STAT                     PIC X(4) VALUE '****'.
 77 PEOP                     PIC X(4) VALUE 'PEOP'.
 77 PERS                     PIC X(4) VALUE 'PERS'.
 77 SINON                    PIC X(5) VALUE 'SINON'.
 77 SINOF                    PIC X(5) VALUE 'SINOF'.
 77 ENDP                     PIC X(4) VALUE 'END.'.
 77 PEOP-DATA                PIC X(60) VALUE 'PEOPCTRLPEOPNAMEPEOP
-       'ADDRPEOPTELEPEOPHIREPEOPBIRTPEOPSOSCEND.'.
```

```
 77 PERS-DATA                    PIC X(60) VALUE 'PERSPEOPPERSSTRTPERS
-       'EXPRPERSDATAEND.                        '.
 77 ENUMBER                     PIC X(9).
 77 DUMMY1                      PIC X(4) VALUE SPACES.
 77 DUMMY2                      PIC X(8) VALUE SPACES.
 77 DUMMY3                      PIC X(9) VALUE SPACES.
 77 DUMMY4                      PIC X(60) VALUE SPACES.
 77 REFERENCE                   PIC X(4).
 77 LK-PATH                     PIC X(8) VALUE 'PEOPLKPE'.
 77 FIRST-READV                 PIC X(3).
 01  PEOP-IN.
     05 PEOP-NUM                PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PEOP-NAM                PIC X(25).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PEOP-ADR                PIC X(20).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PEOP-TEL                PIC X(10).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PEOP-HIR                PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PEOP-BRT                PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PEOP-SOC                PIC X(9).
 01 PERS-IN.
     05 PERS-EMP-NO             PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PERS-DATE               PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PERS-CODE               PIC X(2).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 PERS-PERSONAL           PIC X(21).
 01  PEOP-RECORD.
     05 XNUMBER                 PIC X(6).
```

```
    05 NAME                  PIC X(25).
    05 XADDRESS              PIC X(20).
    05 TELEPHONE             PIC X(10).
    05 HIRE-DATE             PIC X(6).
    05 BIRTH-DATE            PIC X(6).
    05 SOC-NUMB              PIC X(9).
01 PERS-RECORD.
    05 EMP-NO                PIC X(6).
    05 DATE-STARTED          PIC X(6).
    05 EXPERIENCE-CODE       PIC X(2).
    05 PERSONAL-DATA         PIC X(21).
    05 FILLER                PIC X(47) VALUE SPACES.
01  PRINT-LINE.
    05 FILLER                PIC X(5) VALUE SPACES.
    05 DATA1                 PIC X(5).
    05 FILLER                PIC X(5) VALUE SPACES.
    05 DATA2                 PIC X(4).
 01 PRINT-COMMAND.
    05 FILLER                PIC X(5) VALUE SPACES.
    05 COMM                  PIC X(5).
    05 FILLER                PIC X(5) VALUE SPACES.
    05 NUMB                  PIC 9(3).
 01 INPUT-COMMAND.
    05 COMMAND               PIC X(5).
    05 NO-TIMES              PIC 9(3).
01  SCHEMA.
    03 TASKID                PIC X(8) VALUE 'LOADPEOP'.
    03 DBMODD                PIC X(6) VALUE 'PERSON'.
    03 AMODED                PIC X(6) VALUE 'UPDATE'.
    03 LOGOPT                PIC XX   VALUE 'NL'.
    03 REALM-1.
       05 FILE-1             PIC X(4) VALUE 'PEOP'.
       05 MODE-1             PIC X(4) VALUE 'PRIV'.
       05 STAT-1             PIC X(4) VALUE SPACES.
```

```
           05 REALM-2.
              05 FILE-2                 PIC X(4) VALUE 'PERS'.
              05 MODE-2                 PIC X(4) VALUE 'PRIV'.
              05 STAT-2                 PIC X(4) VALUE SPACES.
           03 FILLER                    PIC X(4) VALUE 'END.'.
           03 FILLER                    PIC X(52) VALUE SPACES.
       PROCEDURE DIVISION.
       SIGN-ON.
      *
      *    SIGN ON TO THE DATA BASE IMMEDIATELY.
      *
           OPEN  OUTPUT PRINT-OUT.
           CALL 'HINT' USING SINON STAT DUMMY1 DUMMY1 DUMMY2
               DUMMY3 DUMMY4 SCHEMA ENDP.
           IF STAT IS EQUAL TO '****' THEN GO TO GET-COMMAND.
           MOVE SINON TO DATA1.
           MOVE STAT TO DATA2.
           MOVE ' ' TO CARRIAGECON.
           MOVE PRINT-LINE TO INDATA.
           DISPLAY PEOP-PERS-OUT UPON CRT.
           WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
           GO TO CLOSE-EM.
       GET-COMMAND.
      *
      *    MAIN LOOP - ACCEPTS USER DATA BASE FUNCTION COMMANDS.
      *
           DISPLAY 'ENTER 5 CHARACTER COMMAND.' UPON CRT.
           ACCEPT COMMAND FROM CRT.
           DISPLAY 'ENTER 3 DIGIT NUMBER OF TRANSACTIONS.' UPON CRT.
           ACCEPT NO-TIMES FROM CRT.
           MOVE COMMAND TO COMM.
           MOVE NO-TIMES TO NUMB.
           MOVE ' ' TO CARRIAGECON.
           MOVE PRINT-COMMAND TO INDATA.
```

```
      DISPLAY PEOP-PERS-OUT UPON CRT.
      WRITE PEOP-PERS-OUT AFTER ADVANCING TOP-OF-PAGE.
      IF COMMAND IS EQUAL TO 'CEASE' THEN GO TO CLOSE-EM.
      IF COMMAND IS  EQUAL TO 'ADD-M' OR COMMAND IS  EQUAL TO
          'READM' OR COMMAND  IS  EQUAL TO 'DEL-M' OR COMMAND
          IS EQUAL TO 'WRITM' THEN PERFORM MASTER-TRANS
          THRU  MASTER-EXIT NO-TIMES TIMES.
      IF COMMAND IS EQUAL TO 'READV' OR COMMAND IS EQUAL
          TO 'DELVD' OR COMMAND IS EQUAL TO 'ADDVC' OR
          COMMAND IS EQUAL TO 'WRITV' THEN PERFORM
          VARIABLE-TRANS THRU VARIABLE-EXIT NO-TIMES TIMES.
      GO TO GET-COMMAND.
 MASTER-TRANS.
*
*     ALL TRANSACTIONS INVOLVING THE MASTER DATA BASE FILE
*     TAKE PLACE IN THIS LOOP.  ENTER DATA AS NEEDED.
*
      DISPLAY 'ENTER 6 DIGIT EMPLOYEE NUMBER,' UPON CRT.
      ACCEPT XNUMBER FROM CRT.
      DISPLAY 'ENTER 25 CHARACTER NAME,' UPON CRT.
      ACCEPT NAME FROM CRT.
      DISPLAY 'ENTER 20 CHARACTER ADDRESS,' UPON CRT.
      ACCEPT XADDRESS FROM CRT.
      DISPLAY 'ENTER 10 DIGIT TELEPHONE NUMBER,' UPON CRT.
      ACCEPT TELEPHONE FROM CRT.
      DISPLAY 'ENTER 6 DIGIT HIRE DATE,' UPON CRT.
      ACCEPT HIRE-DATE FROM CRT.
      DISPLAY 'ENTER 6 DIGIT BIRTH DATE,' UPON CRT.
      ACCEPT BIRTH-DATE FROM CRT.
      DISPLAY 'ENTER 9 DIGIT SSN,' UPON CRT.
      ACCEPT SOC-NUMB FROM CRT.
      MOVE XNUMBER TO PEOP-NUM.
      MOVE NAME TO PEOP-NAM.
      MOVE XADDRESS TO PEOP-ADR.
```

```
      MOVE TELEPHONE TO PEOP-TEL.
      MOVE HIRE-DATE TO PEOP-HIR.
      MOVE BIRTH-DATE TO PEOP-BRT.
      MOVE SOC-NUMB TO PEOP-SOC.
      MOVE ' ' TO CARRIAGECON.
      MOVE PEOP-IN TO INDATA.
      DISPLAY PEOP-PERS-OUT UPON CRT.
      WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
      MOVE XNUMBER TO ENUMBER.
      MOVE COMMAND TO FUNCTION.
      CALL 'HINT' USING FUNCTION STAT PEOP DUMMY1 DUMMY2 ENUMBER
          PEOP-DATA PEOP-RECORD ENDP.
      IF STAT IS NOT EQUAL TO '****' THEN PERFORM STOP1
          GO TO MASTER-EXIT.
      IF COMMAND IS EQUAL TO 'READM' THEN MOVE XNUMBER TO PEOP-NUM
          MOVE NAME TO PEOP-NAM
          MOVE XADDRESS TO PEOP-ADR
          MOVE TELEPHONE TO PEOP-TEL
          MOVE HIRE-DATE TO PEOP-HIR
          MOVE BIRTH-DATE TO PEOP-BRT
          MOVE SOC-NUMB TO PEOP-SOC
          MOVE ' ' TO CARRIAGECON
          MOVE PEOP-IN TO INDATA
          DISPLAY PEOP-PERS-OUT UPON CRT
          WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
 MASTER-EXIT.  EXIT.
 VARIABLE-TRANS.
*
*     ALL TRANSACTIONS INVOLVING THE VARIABLE DATA BASE FILE
*     TAKE PLACE IN THIS LOOP.  ENTER DATA AS NEEDED.
*
      DISPLAY 'ENTER 6 DIGIT EMPLOYEE NUMBER.' UPON CRT.
      ACCEPT EMP-NO FROM CRT.
      DISPLAY 'ENTER 6 CHARACTER DATE STARTED.' UPON CRT.
```

```
ACCEPT DATE-STARTED FROM CRT.
DISPLAY 'ENTER 2 CHARACTER EXPERIENCE CODE.' UPON CRT.
ACCEPT EXPERIENCE-CODE FROM CRT.
DISPLAY 'ENTER 21 CHARACTER PERSONAL DATA.' UPON CRT.
ACCEPT PERSONAL-DATA FROM CRT.
MOVE EMP-NO TO PERS-EMP-NO.
MOVE DATE-STARTED TO PERS-DATE.
MOVE EXPERIENCE-CODE TO PERS-CODE.
MOVE PERSONAL-DATA TO PERS-PERSONAL.
MOVE ' ' TO CARRIAGECON.
MOVE PERS-IN TO INDATA.
DISPLAY PEOP-PERS-OUT UPON CRT.
WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
DISPLAY 'READING 1ST RECORD IN CHAIN ?' UPON CRT.
ACCEPT FIRST-READV FROM CRT.
MOVE ' ' TO CARRIAGECON.
MOVE FIRST-READV TO INDATA.
DISPLAY PEOP-PERS-OUT UPON CRT.
WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
IF FIRST-READV IS EQUAL TO 'YES' THEN MOVE 'LKPE'
    TO REFERENCE.
MOVE COMMAND TO FUNCTION.
MOVE EMP-NO TO ENUMBER.
CALL 'HINT' USING FUNCTION STAT PERS REFERENCE LK-PATH
    ENUMBER PERS-DATA PERS-RECORD ENDP.
IF STAT IS NOT EQUAL TO '****' THEN PERFORM STOP1
    GO TO VARIABLE-EXIT.
IF REFERENCE IS EQUAL TO 'END.' THEN DISPLAY 'END OF CHAIN.'
    UPON CRT
    MOVE 'END OF CHAIN.' TO INDATA
    WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
IF COMMAND IS EQUAL TO 'READV' AND REFERENCE IS NOT
    EQUAL TO 'END.' THEN MOVE EMP-NO TO PERS-EMP-NO
    MOVE DATE-STARTED TO PERS-DATE
```

```
           MOVE EXPERIENCE-CODE TO PERS-CODE
           MOVE PERSONAL-DATA TO PERS-PERSONAL
           MOVE ' ' TO CARRIAGECON
           MOVE PERS-IN TO INDATA
           DISPLAY PEOP-PERS-OUT UPON CRT
           WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
 VARIABLE-EXIT.  EXIT.
 STOP1.
*
*     DISPLAY ERROR STATUS RETURNED BY DISTRIBUTED DATA BASE
*     MANAGEMENT SYSTEM.
*
      MOVE FUNCTION TO DATA1.
      MOVE STAT TO DATA2.
      MOVE PRINT-LINE TO INDATA.
      MOVE ' ' TO CARRIAGECON.
      DISPLAY PEOP-PERS-OUT UPON CRT.
      WRITE PEOP-PERS-OUT AFTER ADVANCING 2 LINES.
 CLOSE-EM.
*
*     SIGN OFF OF THE DATA BASE AND TERMINATE EXECUTION.
*
      CALL 'HINT' USING SINOF STAT DUMMY1 DUMMY1 DUMMY2
          DUMMY3 DUMMY4 SCHEMA ENDP.
      CLOSE PRINT-OUT.
      STOP RUN.
```

APPENDIX O

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     LOADSTUD.
   REMARKS.  'LOADSTUD' IS THE APPLICATION PROGRAM WHICH ACCESSES
        THE DATA BASE STUDNT ON THE INTERDATA 8/32 THROUGH
        THE DISTRIBUTED DATA BASE MANAGEMENT SYSTEM.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. INTERDATA MODEL-8-32.
 OBJECT-COMPUTER. INTERDATA MODEL-8-32.
 SPECIAL-NAMES.
     C01 IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT PRINT-OUT  ASSIGN TO LU02-PRINTER.
 DATA DIVISION.
 FILE SECTION.
 FD   PRINT-OUT
     DATA RECORD IS STUD-CRSE-OUT.
 01   STUD-CRSE-OUT.
     05 CARRIAGECON           PIC X.
     05 INDATA                PIC X(132).
 WORKING-STORAGE SECTION.
*
*    TOTAL PARAMETERS AND I/O RECORD STRUCTURES.
*
 77 FUNCTION                  PIC X(5).
 77 STAT                      PIC X(4) VALUE '****'.
 77 STUD                      PIC X(4) VALUE 'STUD'.
 77 CRSE                      PIC X(4) VALUE 'CRSE'.
 77 SINON                     PIC X(5) VALUE 'SINON'.
 77 SINOF                     PIC X(5) VALUE 'SINOF'.
 77 ENDP                      PIC X(4) VALUE 'END.'.
 77 STUD-DATA                 PIC X(60) VALUE 'STUDCTRLSTUDNAMESTUD
-       'STARSTUDGRADSTUDYEAREND.                  '.
```

```
 77 CRSE-DATA                    PIC X(60) VALUE 'CRSESTUDCRSENAMECRSE
-       'NMBRCRSELINOCRSETORECRSEHORSCRSEGRADEND.'.
 77 SNUMBER                     PIC X(9).
 77 DUMMY1                      PIC X(4) VALUE SPACES.
 77 DUMMY2                      PIC X(8) VALUE SPACES.
 77 DUMMY3                      PIC X(9) VALUE SPACES.
 77 DUMMY4                      PIC X(60) VALUE SPACES.
 77 REFERENCE                   PIC X(4).
 77 LK-PATH                     PIC X(8) VALUE 'STUDLKCR'.
 77 FIRST-READV                 PIC X(3).
 01  STUD-IN.
     05 STUD-NUM                PIC X(9).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 STUD-NAM                PIC X(25).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 STUD-STR                PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 STUD-GRD                PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 STUD-YER                PIC X(4).
 01 CRSE-IN.
     05 CRSE-STUD-NO            PIC X(9).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 CRSE-NAME               PIC X(30).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 CRSE-NUMBER             PIC X(6).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 CRSE-LINE-NO            PIC X(4).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 CRSE-TAKEN-OR-ENR       PIC X(1).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 CRSE-HOURS              PIC X(1).
     05 FILLER                  PIC X(5) VALUE SPACES.
     05 CRSE-GRADE              PIC X(1).
```

```
01  STUD-RECORD.
    05 STNUMBER              PIC X(9).
    05 NAME                  PIC X(25).
    05 DATE-STARTED          PIC X(6).
    05 DATE-TO-GRAD          PIC X(6).
    05 YEAR                  PIC X(4).
    05 FILLER                PIC X(32) VALUE SPACES.
01 CRSE-RECORD.
    05 STUD-NO               PIC X(9).
    05 COURSE-NAME           PIC X(30).
    05 COURSE-NUMBER         PIC X(6).
    05 LINE-NO               PIC X(4).
    05 TAKEN-OR-ENROLLED     PIC X(1) .
    05 HOURS                 PIC X(1).
    05 GRADE                 PIC X(1).
    05 FILLER                PIC X(30) VALUE SPACES.
01  PRINT-LINE.
    05 FILLER                PIC X(5) VALUE SPACES.
    05 DATA1                 PIC X(5).
    05 FILLER                PIC X(5) VALUE SPACES.
    05 DATA2                 PIC X(4).
 01 PRINT-COMMAND.
    05 FILLER                PIC X(5) VALUE SPACES.
    05 COMM                  PIC X(5).
    05 FILLER                PIC X(5) VALUE SPACES.
    05 NUMB                  PIC 9(3).
 01 INPUT-COMMAND.
    05 COMMAND               PIC X(5).
    05 NO-TIMES              PIC 9(3).
01  SCHEMA.
    03 TASKID                PIC X(8) VALUE 'LOADSTUD'.
    03 DBMODL                PIC X(6) VALUE 'STUDNT'.
    03 AMODED                PIC X(6) VALUE 'UPDATE'.
```

```
      03 LOGOPT                  PIC XX    VALUE 'NL'.
      03 REALM-1.
         05 FILE-1               PIC X(4)  VALUE 'STUD'.
         05 MODE-1               PIC X(4)  VALUE 'PRIV'.
         05 STAT-1               PIC X(4)  VALUE SPACES.
      03 REALM-2.
         05 FILE-2               PIC X(4)  VALUE 'CRSE'.
         05 MODE-2               PIC X(4)  VALUE 'PRIV'.
         05 STAT-2               PIC X(4)  VALUE SPACES.
      03 FILLER                  PIC X(4)  VALUE 'END.'.
      03 FILLER                  PIC X(32) VALUE SPACES.
 PROCEDURE DIVISION.
 SIGN-ON.
*
*     SIGN ON TO THE DATA BASE IMMEDIATELY.
*
      OPEN  OUTPUT PRINT-OUT.
      CALL 'HINT' USING SINON STAT DUMMY1 DUMMY1 DUMMY2
          DUMMY3 DUMMY4 SCHEMA ENDP.
      IF STAT IS EQUAL TO '****' THEN GO TO GET-COMMAND.
      MOVE SINON TO DATA1.
      MOVE STAT TO DATA2.
      MOVE ' ' TO CARRIAGECON.
      MOVE PRINT-LINE TO INDATA.
      DISPLAY STUD-CRSE-OUT UPON CRT.
      WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
      GO TO CLOSE-EM.
 GET-COMMAND.
*
*     MAIN LOOP - ACCEPTS USER DATA BASE FUNCTION COMMANDS.
*
      DISPLAY 'ENTER 5 CHARACTER COMMAND.' UPON CRT.
      ACCEPT COMMAND FROM CRT.
      DISPLAY 'ENTER 3 DIGIT NUMBER OF TRANSACTIONS.' UPON CRT.
```

```
           ACCEPT NO-TIMES FROM CRT.
           MOVE COMMAND TO COMM.
           MOVE NO-TIMES TO NUMB.
           MOVE ' ' TO CARRIAGECON.
           MOVE PRINT-COMMAND TO INDATA.
           DISPLAY STUD-CRSE-OUT UPON CRT.
           WRITE STUD-CRSE-OUT AFTER ADVANCING TOP-OF-PAGE.
           IF COMMAND IS EQUAL TO 'CEASE' THEN GO TO CLOSE-EM.
           IF COMMAND IS  EQUAL TO 'ADD-M' OR COMMAND IS  EQUAL TO
               'READM' OR COMMAND  IS  EQUAL TO 'DEL-M' OR COMMAND
               IS EQUAL TO 'WRITM' THEN PERFORM MASTER-TRANS
               THRU  MASTER-EXIT NO-TIMES TIMES.
           IF COMMAND IS EQUAL TO 'READV' OR COMMAND IS EQUAL
               TO 'DELVD' OR COMMAND IS EQUAL TO 'ADDVC' OR
               COMMAND IS EQUAL TO 'WRITV' THEN PERFORM
               VARIABLE-TRANS THRU VARIABLE-EXIT NO-TIMES TIMES.
           GO TO GET-COMMAND.
       MASTER-TRANS.
      *
      *    ALL TRANSACTIONS INVOLVING THE MASTER DATA BASE FILE
      *    TAKE PLACE IN THIS LOOP.  ENTER DATA AS NEEDED.
      *
           DISPLAY 'ENTER 9 DIGIT STUDENT NUMBER,' UPON CRT.
           ACCEPT STNUMBER FROM CRT.
           DISPLAY 'ENTER 25 CHARACTER NAME,' UPON CRT.
           ACCEPT NAME FROM CRT.
           DISPLAY 'ENTER 6 CHARACTER DATE STARTED,' UPON CRT.
           ACCEPT DATE-STARTED FROM CRT.
           DISPLAY 'ENTER 6 DIGIT DATE TO GRADUATE,' UPON CRT.
           ACCEPT DATE-TO-GRAD FROM CRT.
           DISPLAY 'ENTER 4 CHARACTER YEAR IN SCHOOL,' UPON CRT.
           ACCEPT YEAR FROM CRT.
           MOVE STNUMBER TO STUD-NUM.
           MOVE NAME TO STUD-NAM.
```

```
      MOVE DATE-STARTED TO STUD-STR.
      MOVE DATE-TO-GRAD TO STUD-GRD.
      MOVE YEAR TO STUD-YER.
      MOVE ' ' TO CARRIAGECON.
      MOVE STUD-IN TO INDATA.
      DISPLAY STUD-CRSE-OUT UPON CRT.
      WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
      MOVE STNUMBER TO SNUMBER.
      MOVE COMMAND TO FUNCTION.
      CALL 'HINT' USING FUNCTION STAT STUD DUMMY1 DUMMY2 SNUMBER
          STUD-DATA STUD-RECORD ENDP.
      IF STAT IS NOT EQUAL TO '****' THEN PERFORM STOP1
          GO TO MASTER-EXIT.
      IF COMMAND IS EQUAL TO 'READM' THEN MOVE STNUMBER TO STUD-NUM
          MOVE NAME TO STUD-NAM
          MOVE DATE-STARTED TO STUD-STR
          MOVE DATE-TO-GRAD TO STUD-GRD
          MOVE YEAR TO STUD-YER
          MOVE ' ' TO CARRIAGECON
          MOVE STUD-IN TO INDATA
          DISPLAY STUD-CRSE-OUT UPON CRT
          WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
 MASTER-EXIT.  EXIT.
 VARIABLE-TRANS.
*
*     ALL TRANSACTIONS INVOLVING THE VARIABLE DATA BASE FILE
*     TAKE PLACE IN THIS LOOP.  ENTER DATA AS NEEDED.
*
      DISPLAY 'ENTER 9 DIGIT STUDENT NUMBER.' UPON CRT.
      ACCEPT STUD-NO FROM CRT.
      DISPLAY 'ENTER 30 CHARACTER COURSE NAME.' UPON CRT.
      ACCEPT COURSE-NAME FROM CRT.
      DISPLAY 'ENTER 6 CHARACTER COURSE NUMBER.' UPON CRT.
      ACCEPT COURSE-NUMBER FROM CRT.
```

```
DISPLAY 'ENTER 4 CHARACTER LINE NUMBER.' UPON CRT.
ACCEPT LINE-NO FROM CRT.
DISPLAY 'ENTER 1 CHARACTER TAKEN OR ENROLLED CODE.' UPON CRT.
ACCEPT TAKEN-OR-ENROLLED FROM CRT.
DISPLAY 'ENTER 1 CHARACTER NUMBER OF HOURS.' UPON CRT.
ACCEPT HOURS FROM CRT.
DISPLAY 'ENTER 1 CHARACTER COURSE GRADE.' UPON CRT.
ACCEPT GRADE FROM CRT.
MOVE STUD-NO TO CRSE-STUD-NO.
MOVE COURSE-NAME TO CRSE-NAME.
MOVE COURSE-NUMBER TO CRSE-NUMBER.
MOVE LINE-NO TO CRSE-LINE-NO.
MOVE TAKEN-OR-ENROLLED TO CRSE-TAKEN-OR-ENR.
MOVE HOURS TO CRSE-HOURS.
MOVE GRADE TO CRSE-GRADE.
MOVE ' ' TO CARRIAGECON.
MOVE CRSE-IN TO INDATA.
DISPLAY STUD-CRSE-OUT UPON CRT.
WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
DISPLAY 'READING 1ST RECORD IN CHAIN ?' UPON CRT.
ACCEPT FIRST-READY FROM CRT.
MOVE ' ' TO CARRIAGECON.
MOVE FIRST-READY TO INDATA.
DISPLAY STUD-CRSE-OUT UPON CRT.
WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
IF FIRST-READY IS EQUAL TO 'YES' THEN MOVE 'LKCR'
    TO REFERENCE.
MOVE COMMAND TO FUNCTION.
MOVE STUD-NO TO SNUMBER.
CALL 'HINT' USING FUNCTION STAT CRSE REFERENCE LK-PATH
    SNUMBER CRSE-DATA CRSE-RECORD ENDP.
IF STAT IS NOT EQUAL TO '****' THEN PERFORM STOP1
    GO TO VARIABLE-EXIT.
IF REFERENCE IS EQUAL TO 'END.' THEN DISPLAY 'END OF CHAIN.'
```

```
           UPON CRT
           MOVE 'END OF CHAIN.' TO INDATA
           WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
       IF COMMAND IS EQUAL TO 'READV' AND REFERENCE IS NOT
           EQUAL TO 'END.' THEN MOVE STUD-NO TO CRSE-STUD-NO
           MOVE COURSE-NAME TO CRSE-NAME
           MOVE COURSE-NUMBER TO CRSE-NUMBER
           MOVE LINE-NO TO CRSE-LINE-NO
           MOVE TAKEN-OR-ENROLLED TO CRSE-TAKEN-OR-ENR
           MOVE HOURS TO CRSE-HOURS
           MOVE GRADE TO CRSE-GRADE
           MOVE ' ' TO CARRIAGECON
           MOVE CRSE-IN TO INDATA
           DISPLAY STUD-CRSE-OUT UPON CRT
           WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
 VARIABLE-EXIT.  EXIT.
 STOP1.
*
*      DISPLAY ERROR STATUS RETURNED BY DISTRIBUTED DATA BASE
*      MANAGEMENT SYSTEM.
*
       MOVE FUNCTION TO DATA1.
       MOVE STAT TO DATA2.
       MOVE PRINT-LINE TO INDATA.
       MOVE ' ' TO CARRIAGECON.
       DISPLAY STUD-CRSE-OUT UPON CRT.
       WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
 CLOSE-EM.
*
*      SIGN OFF OF DATA BASE AND TERMINATE.
*
       CALL 'HINT' USING SINOF STAT DUMMY1 DUMMY1 DUMMY2
           DUMMY3 DUMMY4 SCHEMA ENDP.
       CLOSE PRINT-OUT.
```

```
STOP RUN.
```

APPENDIX P

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     LOADPART.
  REMARKS.  'LOADPART' IS THE APPLICATION PROGRAM WHICH ACCESSES
      THE DATA BASE PIECES ON THE IBM/370 THROUGH THE DISTRIBUTED
      DATA BASE MANAGEMENT SYSTEM.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. INTERDATA MODEL-8-32.
OBJECT-COMPUTER. INTERDATA MODEL-8-32.
SPECIAL-NAMES.
    C01 IS TOP-OF-PAGE.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT PRINT-OUT  ASSIGN TO LU02-PRINTER.
DATA DIVISION.
FILE SECTION.
FD  PRINT-OUT
    DATA RECORD IS PART-SUPL-OUT.
01  PART-SUPL-OUT.
    05 CARRIAGECON          PIC X.
    05 INDATA               PIC X(132).
WORKING-STORAGE SECTION.
*
*   TOTAL PARAMETERS AND I/O STRUCTURES.
*
 77 FUNCTION                PIC X(5).
 77 STAT                    PIC X(4) VALUE '****'.
 77 PART                    PIC X(4) VALUE 'PART'.
 77 SUPL                    PIC X(4) VALUE 'SUPL'.
 77 TOTAL                   PIC X(5) VALUE 'TOTAL'.
 77 OPENM                   PIC X(5) VALUE 'OPENM'.
 77 CLOSM                   PIC X(5) VALUE 'CLOSM'.
 77 DEQUE                   PIC X(5) VALUE 'DEQUE'.
 77 ENDP                    PIC X(4) VALUE 'END.'.
```

```
77 DBMOD                    PIC X(9) VALUE 'PIECES'.
77 TASKNAME                 PIC X(8) VALUE 'LOADPART'.
77 PTNUMBER                 PIC X(9).
77 DUMMY1                   PIC X(4) VALUE SPACES.
77 DUMMY2                   PIC X(8) VALUE SPACES.
77 DUMMY3                   PIC X(9) VALUE SPACES.
77 DUMMY4                   PIC X(60) VALUE SPACES.
77 DUMMY5                   PIC X(82) VALUE SPACES.
77 REFERENCE                PIC X(4).
77 LK-PATH                  PIC X(8) VALUE 'PARTLKSU'.
77 FIRST-READV              PIC X(3).
01  PART-IN.
    05 PART-NUM             PIC X(9).
    05 FILLER               PIC X(5) VALUE SPACES.
    05 PART-NAM             PIC X(25).
    05 FILLER               PIC X(5) VALUE SPACES.
    05 PART-DES             PIC X(30).
01 SUPL-IN.
    05 PART-SUPL-NO         PIC X(9).
    05 FILLER               PIC X(5) VALUE SPACES.
    05 SUPL-NAME            PIC X(25).
    05 FILLER               PIC X(5) VALUE SPACES.
    05 SUPL-ADR             PIC X(20).
    05 FILLER               PIC X(5) VALUE SPACES.
    05 SUPL-TEL             PIC X(10).
01  PART-RECORD.
    05 PNUMBER              PIC X(9).
    05 PNAME                PIC X(25).
    05 PDESC                PIC X(30).
    05 FILLER               PIC X(18) VALUE SPACES.
01 SUPL-RECORD.
    05 SNUMBER              PIC X(9).
    05 SNAME                PIC X(25).
    05 SADDRESS             PIC X(20).
```

```
           05 STELEPHONE              PIC X(10).
           05 FILLER                  PIC X(18) VALUE SPACES.
       01  PRINT-LINE.
           05 FILLER                  PIC X(5) VALUE SPACES.
           05 DATA1                   PIC X(5).
           05 FILLER                  PIC X(5) VALUE SPACES.
           05 DATA2                   PIC X(4).
        01 PRINT-COMMAND.
           05 FILLER                  PIC X(5) VALUE SPACES.
           05 COMM                    PIC X(5).
           05 FILLER                  PIC X(5) VALUE SPACES.
           05 NUMB                    PIC 9(3).
        01 INPUT-COMMAND.
           05 COMMAND                 PIC X(5).
           05 NO-TIMES                PIC 9(3).
       01  PART-DATA.
           03 PARTCTRL                PIC X(8) VALUE 'PARTCTRL'.
           03 PARTNAME                PIC X(8) VALUE 'PARTNAME'.
           03 PARTDESC                PIC X(8) VALUE 'PARTDESC'.
           03 ENDS                    PIC X(4) VALUE 'END.'.
           03 FILLER                  PIC X(32) VALUE SPACES.
       01  SUPL-DATA.
           03 SUPLPART                PIC X(8) VALUE 'SUPLPART'.
           03 SUPLNAME                PIC X(8) VALUE 'SUPLNAME'.
           03 SUPLADDR                PIC X(8) VALUE 'SUPLADDR'.
           03 SUPLTELE                PIC X(8) VALUE 'SUPLTELE'.
           03 ENDS                    PIC X(4) VALUE 'END.'.
           03 FILLER                  PIC X(24) VALUE SPACES.
       PROCEDURE DIVISION.
       SIGN-ON.
      *
      *    SIGN ON AND OPEN UP THE DATA BASE IMMEDIATELY.
      *
           OPEN  OUTPUT PRINT-OUT.
```

```
     CALL 'HINT' USING TOTAL STAT DUMMY1 DUMMY1 TASKNAME
         DBMOD DUMMY4 DUMMY5 ENDP.
     IF STAT IS EQUAL TO '****' THEN GO TO OPEN-UP.
     MOVE TOTAL TO DATA1.
     MOVE STAT TO DATA2.
     MOVE ' ' TO CARRIAGECON.
     MOVE PRINT-LINE TO INDATA.
     DISPLAY PART-SUPL-OUT UPON CRT.
     WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
     GO TO THATS-IT.
OPEN-UP.
     CALL 'HINT' USING OPENM STAT PART DUMMY1 DUMMY2 DUMMY3
         DUMMY4 DUMMY5 ENDP.
     IF STAT IS NOT EQUAL TO '****' THEN MOVE OPENM TO DATA1
         MOVE STAT TO DATA2
         MOVE ' ' TO CARRIAGECON
         MOVE PRINT-LINE TO INDATA
         DISPLAY PART-SUPL-OUT UPON CRT
         WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES
         GO TO CLOSE-EM1.
     CALL 'HINT' USING OPENM STAT SUPL DUMMY1 DUMMY2 DUMMY3
         DUMMY4 DUMMY5 ENDP.
     IF STAT IS NOT EQUAL TO '****' THEN MOVE OPENM TO DATA1
         MOVE STAT TO DATA2
         MOVE ' ' TO CARRIAGECON
         MOVE PRINT-LINE TO INDATA
         DISPLAY PART-SUPL-OUT UPON CRT
         WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES
         GO TO CLOSE-EM2.
GET-COMMAND.
*
*    MAIN LOOP - ACCEPTS USER DATA BASE FUNCTION COMMANDS.
*
     DISPLAY 'ENTER 5 CHARACTER COMMAND.' UPON CRT.
```

```
      ACCEPT COMMAND FROM CRT.
      DISPLAY 'ENTER 3 DIGIT NUMBER OF TRANSACTIONS,' UPON CRT.
      ACCEPT NO-TIMES FROM CRT.
      MOVE COMMAND TO COMM.
      MOVE NO-TIMES TO NUMB.
      MOVE ' ' TO CARRIAGECON.
      MOVE PRINT-COMMAND TO INDATA.
      DISPLAY PART-SUPL-OUT UPON CRT.
      WRITE PART-SUPL-OUT AFTER ADVANCING TOP-OF-PAGE.
      IF COMMAND IS EQUAL TO 'CEASE' THEN GO TO CLOSE-EM2.
      IF COMMAND IS  EQUAL TO 'ADD-M' OR COMMAND IS  EQUAL TO
          'READM' OR COMMAND  IS  EQUAL TO 'DEL-M' OR COMMAND
          IS EQUAL TO 'WRITM' THEN PERFORM MASTER-TRANS
          THRU  MASTER-EXIT NO-TIMES TIMES.
      IF COMMAND IS EQUAL TO 'READV' OR COMMAND IS EQUAL
          TO 'DELVD' OR COMMAND IS EQUAL TO 'ADDVC' OR
          COMMAND IS EQUAL TO 'WRITV' THEN PERFORM
          VARIABLE-TRANS THRU VARIABLE-EXIT NO-TIMES TIMES.
      GO TO GET-COMMAND.
 MASTER-TRANS.
*
*     ALL TRANSACTIONS INVOLVING THE MASTER DATA BASE FILE
*     TAKE PLACE IN THIS LOOP.  ENTER DATA AS NEEDED.
*
      DISPLAY 'ENTER 9 DIGIT PART NUMBER,' UPON CRT.
      ACCEPT PNUMBER FROM CRT.
      DISPLAY 'ENTER 25 CHARACTER PART NAME,' UPON CRT.
      ACCEPT PNAME FROM CRT.
      DISPLAY 'ENTER 30 CHARACTER PART DESCRIPTION,' UPON CRT.
      ACCEPT PDESC FROM CRT.
      MOVE PNUMBER TO PART-NUM.
      MOVE PNAME TO PART-NAM.
      MOVE PDESC TO PART-DES.
      MOVE ' ' TO CARRIAGECON.
```

```
      MOVE PART-IN TO INDATA.
      DISPLAY PART-SUPL-OUT UPON CRT.
      WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
      MOVE PNUMBER TO PTNUMBER.
      MOVE COMMAND TO FUNCTION.
      CALL 'HINT' USING FUNCTION STAT PART DUMMY1 DUMMY2 PTNUMBER
          PART-DATA PART-RECORD ENDP.
      IF STAT IS NOT EQUAL TO '****' THEN PERFORM STOP1
          GO TO MASTER-EXIT.
      IF COMMAND IS EQUAL TO 'READM' THEN MOVE PNUMBER TO PART-NUM
          MOVE PNAME TO PART-NAM
          MOVE PDESC TO PART-DES
          MOVE ' ' TO CARRIAGECON
          MOVE PART-IN TO INDATA
          DISPLAY PART-SUPL-OUT UPON CRT
          WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
 MASTER-EXIT.  EXIT.
 VARIABLE-TRANS.
*
*     ALL TRANSACTIONS INVOLVING THE VARIABLE DATA BASE FILE
*     TAKE PLACE IN THIS LOOP.  ENTER DATA AS NEEDED.
*
      DISPLAY 'ENTER 9 DIGIT PART NUMBER,' UPON CRT.
      ACCEPT SNUMBER FROM CRT.
      DISPLAY 'ENTER 25 CHARACTER SUPPLIER NAME,' UPON CRT.
      ACCEPT SNAME FROM CRT.
      DISPLAY 'ENTER 20 CHARACTER SUPPLIER ADDRESS,' UPON CRT.
      ACCEPT SADDRESS FROM CRT.
      DISPLAY 'ENTER 10 CHARACTER SUPPLIER TELEPHONE NUMBER,' UPON
       CRT.
      ACCEPT STELEPHONE FROM CRT.
      MOVE SNUMBER TO PART-SUPL-NO.
      MOVE SNAME TO SUPL-NAME.
      MOVE SADDRESS TO SUPL-ADR.
```

```
           MOVE STELEPHONE TO SUPL-TEL.
           MOVE ' ' TO CARRIAGECON.
           MOVE SUPL-IN TO INDATA.
           DISPLAY PART-SUPL-OUT UPON CRT.
           WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
           DISPLAY 'READING 1ST RECORD IN CHAIN ?' UPON CRT.
           ACCEPT FIRST-READV FROM CRT.
           MOVE ' ' TO CARRIAGECON.
           MOVE FIRST-READV TO INDATA.
           DISPLAY PART-SUPL-OUT UPON CRT.
           WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
           IF FIRST-READV IS EQUAL TO 'YES' THEN MOVE 'LKSU'
               TO REFERENCE.
           MOVE COMMAND TO FUNCTION.
           MOVE SNUMBER TO PTNUMBER.
           CALL 'HINT' USING FUNCTION STAT SUPL REFERENCE LK-PATH
               PTNUMBER SUPL-DATA SUPL-RECORD ENDP.
           IF STAT IS NOT EQUAL TO '****' THEN PERFORM STOP1
               GO TO VARIABLE-EXIT.
           IF REFERENCE IS EQUAL TO 'END.' THEN DISPLAY 'END OF CHAIN.'
               UPON CRT
               MOVE 'END OF CHAIN.' TO INDATA
               WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
           IF COMMAND IS EQUAL TO 'READV' AND REFERENCE IS NOT
               EQUAL TO 'END.' THEN MOVE SNUMBER TO PART-SUPL-NO
               MOVE SNAME TO SUPL-NAME
               MOVE SADDRESS TO SUPL-ADR
               MOVE STELEPHONE TO SUPL-TEL
               MOVE ' ' TO CARRIAGECON
               MOVE SUPL-IN TO INDATA
               DISPLAY PART-SUPL-OUT UPON CRT
               WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
VARIABLE-EXIT.  EXIT.
STOP1.
```

```
*
*     DISPLAY ERROR STATUS RETURNED BY DISTRIBUTED DATA BASE
*     MANAGEMENT SYSTEM.
*
      MOVE FUNCTION TO DATA1.
      MOVE STAT TO DATA2.
      MOVE PRINT-LINE TO INDATA.
      MOVE ' ' TO CARRIAGECON.
      DISPLAY PART-SUPL-OUT UPON CRT.
      WRITE PART-SUPL-OUT AFTER ADVANCING 2 LINES.
 CLOSE-EM2.
*
*     CLOSE DATA BASE AND TERMINATE.
*
      CALL 'HINT' USING CLOSM STAT SUPL DUMMY1 DUMMY2
          DUMMY3 DUMMY4 DUMMY5 ENDP.
 CLOSE-EM1.
      CALL 'HINT' USING CLOSM STAT PART DUMMY1 DUMMY2
          DUMMY3 DUMMY4 DUMMY5 ENDP.
 THATS-IT.
      CALL 'HINT' USING DEQUE STAT DUMMY1 DUMMY1 TASKNAME
          DUMMY3 DUMMY4 DUMMY5 ENDP.
      CLOSE PRINT-OUT.
      STOP RUN.
```

A USER TRANSPARENT DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

by

Richard Dale Housh

B.S., Kansas State University, Manhattan Kansas, 1976

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1978

The purpose of this project is to design a distributed data base management system which runs on a variety of processors and is transparent to the user.

A prototype version of the distributed data base management system will be implemented. This implementation runs on the Interdata 8/32 and Interdata 7/32 processors, both utilizing the OS/32-MT3 operating system. The data base management system is CINCOM, INC.'s TOTAL, while the remainder of the software is written in an assortment of languages. Data bases may reside on either machine and few restrictions are placed on the user.