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:

[Signature]
Major Professor
# TABLE OF CONTENTS

I. Introduction ........................................ 4

II. Background ......................................... 6

III. System Organization ............................... 8

IV. Software Design .................................... 13

V. Results ............................................. 24

VI. Future Enhancements ............................... 26

VII. Conclusion ......................................... 29

REFERENCES ............................................. 30

APPENDIX

A. Data Base Generation (DBGEN1 and DBGEN2) .... A-1

B. DDL for Data Base on Interdata 7/32 (PERSON)  B-1

C. DDL for Data Base on Interdata 8/32 (STUDNT)  C-1

D. DDL for Data Base on 370 (PIECES) .............. D-1

E. Back-end Task Establish CSS (COBLEST) .......... E-1

F. Assembler Interface (MESSINFC and NEWINFC) .. F-1

G. I732BINT ........................................... G-1

H. I832BINT ........................................... H-1

I. IBMBINT ............................................. I-1

J. Front-end Task Establish CSS (COBESTAB) ....... J-1

K. I732HINT ........................................... K-1

L. I832HINT ........................................... L-1

M. IBMHINT ........................................... M-1

N. Application Program Accessing 7/32 Data Base  N-1

O. Application Program Accessing 8/32 Data Base  O-1

P. Application Program Accessing IBM Data Base  P-1
ILLEGIBLE DOCUMENT

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

THIS IS THE BEST COPY AVAILABLE
FIGURES

FIGURE 1 Interdata 8/32 and Interdata 7/32 Hardware . 9
FIGURE 2 Front-end and Back-end Task Components . . 12
FIGURE 3 Front-end and Back-end DBMS ............. 14
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 finally 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, ADBAS, 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 database 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
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 in FIGURE 2.
**FIGURE 2**

**FRONT-END TASK COMPONENTS**

<table>
<thead>
<tr>
<th>Application Program</th>
<th>HINT</th>
<th>MESSINFC</th>
</tr>
</thead>
</table>

**BACK-END TASK COMPONENTS**

| MESSINFC | DINT | 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 3/32, 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 DEGEN 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

INTERDATA 7/32 BACK-END

Message System Task
Other Tasks
Back-end Task Accessing PERSON
System Buffer
Operating System

STUDNT

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. BUFL 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 BUFL back to the front-end task. This paragraph will loop until it successfully sends BUFL 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 'AE' 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 BUFL. Referring to I732BINT again, the length of the first element is set by the fact that there are only six different types of calls to DATBAS that I732BINT can make. The next five elements in BUFL 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. -

COBOLST 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 in 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 BUFL 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 BUFL. The number moved to the element CALLER of BUFL 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 BUFL through the message system to the correct back-end. If BUFL 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 BUFL 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 forward 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 DATABASE. 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 I. 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. -

RENAME I732HINT.TSK,PERSEBACK.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 I732DINT 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, OPENN, CLOSN, 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 BUF1LT11. 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 DINT 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 DINT 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 preceding 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


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.


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


***************
*
* APPENDIX A *
*
***************
$COPY

******************************************************************************

* INTERNAL TOTAL DATA BASE GENERATION
*

******************************************************************************

$NO
$IFX $1,CAL1,D $1,CAL1$ENDC
$IFX DBGEN,WRK,D OBGEN,WRK$ENDC
AL $1,CAL,IN,90/3
AL OBGEN,WRK,IN,256/1
LO OBGEN,TSK
AS 1, $1
AS 2, $1
AS 3, $1,CAL
AS 4, OBGEN,WRK
ST
$EXIT
LO CAL.TSK
$IFX @1,OBJ1;DEL @1,OBJ1;ENDC
AL @1,OBJ1,IN,126/1
$IFNX CALS4,SCTIAL CALS4,SCT,IN,86/5;ENDC
$IFNX CALS5,SCTIAL CALS5,SCT,IN,1024/1;ENDC
$IFNX CALS6,SCTIAL CALS6,SCT,IN,126/1;ENDC
$IFNX CALS8,SCTIAL CALS8,SCT,IN,86/5;ENDC
$IFNX CALS9,SCTIAL CALS9,SCT,IN,86/5;ENDC
AS 1,31,CAL
AS 2,31,OBJ
AS 3,33
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,90RX
$EXIT
***************
*               *
* APPENDIX b   *
*               *
***************
BEGIN-DATA-BASE-GENERATION
DATA-BASE-NAME=PERSON
OPTIONS=OUTPUT=Y
SHARE=10
IOAREA=HAS1
IOAREA=VAR1
END=10
BEGIN-MASTER-DATA-SET
DATA-SET-NAME=PEOP
IOAREA=HAS1
MASTER-DATA
PEOPROOT=8
PEOPCTRL=6 PIC X(6) CONTROL KEY FOR PEOP
PEOPLKP=8 PIC X(6) LINK TO PERS RECORDS (ALL)
PEOPNAME=25 PIC X(25) NAME
PEOPSOC=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
PEOPSEX=1 PIC X(1) SEX
PEOPDIV=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
BEGIN-MASTER-DATA-SET
END-MASTER-DATA-SET
BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=PERS
IOAREA=VAR1
BASE-DATA
PEOPSPERS=6 PIC X(6) CONTROL KEY TO PEOP RECORD
PEOPLKP=8=PERSPEOP LINK FROM PEOP TO PERS
PERSSTRT=6 PIC X(6) DATE EXPERIENCE STARTED
B-1
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
BEGIN-DATABASE-GENERATION
DATA-BASE-NAME=STUDENT
OPTIONS=OUTPUT=Y
SHARE=IO
IOAREA=MAS1
IOAREA=VAR1
END-IO
BEGIN-MASTER-DATA-SET
DATA-SET-NAME=STUD
IOAREA=MAS1
MASTER-DATA
STUDKEY=8
STUDCTRL=9 PIC X(9) CONTROL KEY FOR STUD
STUDLKCR=8 PIC X(25) LINK TO CRSE RECORDS (ALL)
STUUDNAME=25 PIC X(25) NAME
STUDSTART=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 PIC X(25) LINK FROM STUD TO CRSE
CRSENAME=30 PIC X(30) COURSE NAME
CRSENUMBER=6 PIC X(6) COURSE NUMBER
CRSELINE=4 PIC X(4) COURSE LINE NUMBER
CRSESMESR=2 PIC X(2) SEMESTER
CRSETORE=1 PIC X(1) TAKEN OR ENROLLED
C-1
EXEC DUMMY

DD DSN=DS2L1.LOADLIB(LOADPART),DISP=(OLD,DELETE)
DD DSN=DS2L1.PART,DISP=(OLD,DELETE),UNIT=SYSDA,
   SPACE=(TRK,1),VOL=REF=DS2L1.TOTAL,LINKLIB
DD DSN=DS2L1.SUPL,DISP=(OLD,DELETE),UNIT=SYSDA,
   SPACE=(TRK,2),VOL=REF=DS2L1.TOTAL,LINKLIB

EXEC DUMMY

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
DD GEN EXEC PGM=DBGEND
STEPLIB DD DSN=DS2L1.TOTAL,LINKLIB,DISP=SHR
SYSPRINT DD SYSOUT=A
SYSPUNCH DD DSN=&&DBSOURCE,DISP=,*PASS*,UNIT=SYSDA,
   SPACE=(TRK,(5,5)),DCBO=BLKSIZE=3120
SYSSIN 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  PIC X(9)  LINK TO SUPL RECORDS (ALL)
PARTNAME=25  PIC X(25)  NAME
PARTDESC=30  PIC X(30)  PART DESCRIPTION
PARTFILL=30  PIC X(20)  FILLER
END-DATA:
DEVICE=3360
TOTAL-LOGICAL-RECORDS=25
END-MASTER-DATA-SET;
BEGIN-ENTRY-DATA-SET:
DATA-SET-NAME=SUPL
IOARLA=VAR1
BASE-DATA:
SUPLPART=9 PIC X(9) CONTROL KEY TO PART RECORD
PARTLKSU=8 PIC X(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-ENTRY-DATA:
DEVICE=3330
TOTAL-LOGICAL-RECORDS=100
CYLINDER-LOAD-LIMIT=70
END-ENTRY-DATA-SET:
END-ENTRY-DATA-SET-GENERATION:
//ASM EXEC ASMGCL,COND=(4,LT,DBGGEN)
//SYSPIN DD USN=&DDSOURCE,DISP=(OLD,DELETE)
//LKED,SYSIN DD USN=DS2L1,LOADLIB,DISP=SHR
//LKED,SYSIN DD *
   NAME PIECES(R)
//FORMAT EXEC PGM=FORMAT
//STEPLIB DD USN=DS2L1,LOADLIB,DISP=SHR
// DD USN=DS2L1,TOTAL,LINKLIB,DISP=SHR
//SYSPRINT DD SYSDA=R
//PART DD USN=DS2L1,PART,DISP=(,CATLG),UNIT=SYSDA,
// SPACE=(TRK,1),VOL=REF=DS2L1,TOTAL,LINKLIB
//SUPL DD USN=DS2L1,SUPL,DISP=(,CATLG),UNIT=SYSDA,
// SPACE=(TRK,2),VOL=REF=DS2L1,TOTAL,LINKLIB
//SYSPIN DD *
FORMAT PIECES PART,SUPL
>IFNULL @1;COPY
  * CALLING FORMAT: COBOL INTERFCE NAME, ASMLR INTERFACE NAME,
  * TOTAL, DATA-BASE NAME
  $NO
  $EXIT;SEND
  $IFNX @1.OBJ;COPY
  * FILE @1.OBJ DOES NOT EXIST
  $NO
  $CLEAR ;EXIT ;$ENDC
  $IFNX SYS2;ESTAB CMD ;DEL SYS2;ESTAB CMD ;$ENDC
  $IFNX @1.TCM
  $BUILD SYS2;ESTAB CMD
  ES TA
  IN @1.OBJ
  EDIT SYS2;CHU2O00L1B/S
  IN @2.OBJ
  IN @3.OBJ
  IN @4.OBJ
  BU TA, @1.TSK
  ARAP
  END
  $ENDB
  $ENDC
  $IFX @1.TSK;DE @1.TSK;$ENDC
  $IFNX SYS2;ESTAB SCR TAL SYS2;ESTAB SCR IN,126/4 $ENDC
  $IFX SYS2;ESTAB LST ;DEL SYS2;ESTAB LST ;$ENDC
  AL SYS2;ESTAB LST IN,126/4
  LU TET32
  AS 1 @1.OBJ
  AS 3,PR:
  TE 4 IN,126
  $IFX @1.TCM AS 5, @1.TCM;SEND
  $IFNX @1.TCM AS 5, SYS2;ESTAB CMD ;$ENDC
  AS 7,CON:

E-1
MESSAGE SYSTEM INTERFACE FROM INTERDATA Cobol

POSSIBLE TASK CANCELLATION ERRORS
BECAUSE OF BAD COBOL CALL

<table>
<thead>
<tr>
<th>CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>WRONG NUMBER OF ARGUMENTS</td>
</tr>
<tr>
<td>55</td>
<td>ILLEGAL REMOTE-ID LENGTH</td>
</tr>
<tr>
<td>56</td>
<td>ILLEGAL LOCAL-ID LENGTH</td>
</tr>
<tr>
<td>57</td>
<td>ILLEGAL STATUS LENGTH</td>
</tr>
</tbody>
</table>

POSSIBLE BAD STATUS RETURNS
NOT DUE TO MESSAGE SYSTEM

<table>
<thead>
<tr>
<th>CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>UNABLE TO DETERMINE LOCAL-ID</td>
</tr>
</tbody>
</table>

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
COHMD 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) =
    BUFFER-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
ENTRY SEND
STM ALL, REGSAV
BAL LINK, P10DUMB
LI COMMAND, C'SEND'
BAL LINK, P5CHKARG
ST TOWHOM, SENDWMID
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
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,RECVSCV
LH    STATUS,RECVCDOU
STH   STATUS,0(BUF3ADD)
LM    ALL,RECSAV
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

F-5
WHOA-M-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 COMMDC, C`CONN`
BAL LINK, P5CHKARG
ST TOWHON, CONNMID ; 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, GETID SVC
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
SVC  0, CONNSVC
LH   STATUS, CONNCDOU
STH  STATUS, 0 (BUF2ADD)
BS   P3X001

P3E001 EQU *
LHI  STATUS, 58
STH  STATUS, 0 (BUF2ADD)

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'

F-7
* 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 COMHD, C'DISC'
BAL LINK, P8CK2ARG
ST TOWHOM, DISCRMID ; STORE REMOTE ID.
BAL LINK, P7CKRMSB
SVC 0, DISCSVC
LH STATUS, DISCSOV
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
RETURN

SPACE
SPACE
SPACE
P5CHKARG

LM ARG4,0(15)
CLHI ARG4,16
BES P5E001
LHI ERROR,1
BAL ERLINK,P9ERROR
B P5X001

P5E001 EQU *

F-9
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 TOWHON,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
*

F-10
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
*        ENDF
*        RETURN
*
SPACE
SPACE
SPACE
P7CKRHMID  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
*
F-11
PROC PUB-CHECK-FOR-TWO-ARGUMENTS

IF BUFFER-OF-CALLER-ARGUMENTS NOT = 2
  CALLER-ERROR = 1
  DD PUB-CALLER-ERROR
ENDL

BUFFER-ADDRESS-1ST-ARGUMENT =
  CALLER (BUFFER-ADDRESS-1ST-ARGUMENT)
TO-WHO,
  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

F-12

PCL2001 EN  ARG1,0(15)
CLS1  ARG1,12  4(N+1)
BL    PCL001
LTH1  ERROR, L
BAT   LCBLK, PERROR
B    PDA001

PCL001 EN
  OF1AD0,0(ARG2)
  TOAH0,0(OF1AD0)
  LEN1AD0,0(ARG2)
  LUF2AD0,0(ARG3)
  LEN2AD0,1(ARG3)

F-12
TITLE PUY-CALLER-ERROR

PROC PUY-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) =
  CONJRH
ISSUE-MESSAGE-TO-CONSOLE
CANCEL-TASK (100)

SPACE
SPAC
SPACE
PYERROR
SIS ERROR,1
SLET ERROR+3
L TEMP,ERRORTAB(ERROR)
SIT TEMP,ERRORLEN
L TEMP,ERRORTAB+4 (ERROR)
S1 TEMP,ERRADD
S1 COMMON,0 (TEMP)
SVC +ERRSVC
SVC 3100

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,MES1,MES2LEN,MES2
         DAC MES3LEN,MES3,MES4LEN,MES4

ERROR MESSAGES

F-14
ALIGN 4
MESS1 DC C'xxxx-54 WRONG NUMBER OF ARGUMENTS'
MESS1LEN EQU *-MESS1
ALIGN 4
MESS2 DC C'xxxx-55 ILLEGAL REMOTE-ID LENGTH'
MESS2LEN EQU *-MESS2
ALIGN 4
MESS3 DC C'xxxx-56 ILLEGAL LOCAL-ID LENGTH'
MESS3LEN EQU *-MESS3
ALIGN 4
MESS4 DC C'xxxx-57 ILLEGAL STATUS LENGTH'
MESS4LEN EQU *-MESS4
TITLE SVC BLOCKS

* GET ID SVC *

ALIGN 4
GETIDSVC DB 0
DB 1
WHAT EVER
GETSTAT DC H'0'
STATUS OF GET ID
LOCALID DAC 0
DAC 0
ID IN THE FORM CMTP
- NUMBER INDICATES PASCAL

* SEND SVC *

ALIGN 4
SEND SVC DB 1
DB 0
WHAT EVER
DS 2
STATUS
DAC A(SENDSCB)
ADDRESS OF SCB
DS 4
NOT USED

* RECEIVE SVC

F-15
* 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

F-16
* SEND SCB
* ALIGN 4
SENDS CB EQU *
SENDRMID DS 4
SENDLCID DS 4
SENDMBF DAC 0
SENDTXB DAC 0
SENDLCM DC H'0'
SENDLTX DC H'0'
SENDCDIN DC H'60'
SENDCCOU DS 2
SENDCMD DC H'2'

REMOTE ID
LOCAL ID
COMMAND TEXT BUFFER
DATA TEXT BUFFER
LENGTH OF COMMAND TEXT
LENGTH OF DATA TEXT
CODE, IN (MESSAGE COMMAND)
CODE, OUT MESSAGE RETURN STATUS
MESSAGE COMMAND CLASS

* SPACE 3
* RECEIVE SCB
* ALIGN 4
RECVSCB EQU *
RECVRMID DS 4
RECVLCID DS 4
RECVMBF DAC 0
RECVTXB DAC 0
RECVLCM DC H'0'
RECVLTX DC H'0'
REVCDOIN DC H'60'
REVCDOU DC H'0'
REVCMD DC H'3'

REMOTE ID
LOCAL ID
COMMAND TEXT BUFFER
DATA TEXT BUFFER
LENGTH OF COMMAND TEXT
LENGTH OF DATA TEXT
CODE, IN (MESSAGE COMMAND)
CODE, OUT MESSAGE RETURN STATUS
MESSAGE COMMAND CLASS

* EJECT
*
* CONNECT SCB
*

F-17
ALIGN 4
CONNSCB EQU *
CONNRID DS 4
CONNLCID DS 4
CONNCBF DC 0
CONNTXBF DC 0
CONNLNCBH DC H'0'
CONNLNTX DC H'0'
CONNCBID DC H'3'
CONNCC0U DS 2
CONNC0D DC H'0'

* 
SPACE SPACE SPACE

* 
DISCONNECT SCB

ALIGN 4
DISCSCB EQU *
DISCRID DS 4
DISCLCID DS 4
DISCCBF DC 0
DISCTXBF DC 0
DISCLNCBH DC H'0'
DISCLNTX DC H'0'
DISCCBID DC H'5'
DISCC0U DS 2
DISCC0D DC H'1'

* TILL SYMBOL DICTIONARY
END

REMOTE ID
LOCAL ID
COMMAND TEXT BUFFER
DATA TEXT BUFFER
LENGTH OF COMMAND TEXT
LENGTH OF DATA TEXT
CODE.IN (MESSAGE COMMAND)
CODE.OUT (MESSAGE RETURN STATUS)
MESSAGE COMMAND CLASS

COMMAND TEXT BUFFER
DATA TEXT BUFFER
LENGTH OF COMMAND TEXT
LENGTH OF DATA TEXT
CODE.IN (MESSAGE COMMAND)
CODE.OUT (MESSAGE RETURN STATUS)
MESSAGE COMMAND CLASS
MSGSYS 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

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>Wrong number of arguments</td>
</tr>
<tr>
<td>55</td>
<td>Illegal remote-ID length</td>
</tr>
<tr>
<td>56</td>
<td>Illegal local-ID length</td>
</tr>
<tr>
<td>57</td>
<td>Illegal status length</td>
</tr>
</tbody>
</table>

Possible bad status returns
  * not due to message system

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>Unable to determine local-ID</td>
</tr>
</tbody>
</table>

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
TOJHOM    EQU      2
STATUS    EQU      2
WHOAMI    EQU      2
ERROR     EQU      3
COMMD     EQU      4
ERLINK    EQU      5
BUF1ADD   EQU      5
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 PI0-CHECK-DUMB-INTERDATA-ADDRESS
COMMAND = 'SEND'
DO PI5-CHECK-CALLER-ARGUMENTS
RECV-SCB (REMOTE-ID) =
TO-WHOM
SEND-SCB (DATA-BUFFER-ADDRESS) =
BUFFER-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
STM ALL-REGS
BAL LINK, PI0DUMB
LI COMM, C'SEND'
BAL LINK, PI5CHECKARG
ST TO-WHOM, SENDPRI
LIS TO-WHOM, RO1 - JIM R. = DEC. 3, 1977
ST TO-WHOM, SENDTXTBF RO1 - JIM R. = DEC. 3, 1977
ST TO-WHOM, SENDABF RO1 - JIM R. = DEC. 3, 1977
STH TO-WHOM, SENDTXTX RO1 - JIM R. = DEC. 3, 1977

F-21
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

F-22
DO POG-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
SPACL
SPACL
ENTRY

CONN
SIM ALL, REG3AV
BAL LINK, P10DUMB

F-24
LI COMMDC\$CONNN.
BAL LINK,P5CHKARG
ST TOWHOM,CONNNMID;CHANGED TO SAVE REG,2
L  LINK,LOCALID   K01- JIM R. - DEC. 3, 1977
BNZ MUTC\$ORN NOT THE FIRST CONNECTION;
     * SO DON'T DO THE GETID SVC0.
BAL   LINK,P6CKLACD
SVC   0,GETID\$VC
LH   STATUS\$GETSTAT
BNZ   33L001
L   WHOAMI\$,O(BUF2AUD)
NI   WHOAMI\$,X\$FFFF
LR   TEMP2,WHOAMI
MULTCONN EQU *                      K01- JIM R. - DEC. 3, 1977
L   WHOAMI\$LOCALID
NI   WHOAMI\$,Y\$FFFF0000
OK   WHOAMI\$,TEMP2
ST   WHOAMI\$,S\$NCID
ST   WHOAMI\$,RE\$CVID
ST   WHOAMI\$,CONN\$CID
ST   WHOAMI\$,DISC\$CID
ST   WHOAMI\$,O(BUF2AUD)
SVC   0,CONN\$VC
LH   STATUS\$,CONNCOU
STH  STATUS\$,O(BUF3AUD)
BS   33X001
33L001 EQU *
LHI  STATUS\$,58
STH  STATUS\$,O(BUF3AUD)
33X001 EQU *
LM   ALL\$REGS\$AV
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
ENTRY DISC
STM ALL, REGSAV
BAL LINK, P10DUMB
LI COMM, C 'DISC'
BAL LINK, P8CK2ARG
ST TOWHOM, DISCMID ; STORE REMOTE ID.
BAL LINK, PYCKRMID
SVC 0, DISCSVC
LH STATUS, DISCODOU
STH STATUS, O (BUF2AUD)
LM ALL, REGSAV
A 15, V (15)
BR 15
TITLE P05-CHECK-CALLER-ARGUMENTS

PROC P05-CHECK-CALLER-ARGUMENTS

IF NUMBER-OF-CALLER-ARGUMENTS NOT = 3
   CALLER-ERROR = 1
DU 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-WHOME =
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
RETURN

SPACE
SPACE
SPACE

POCHKARG
LM  ARG4, U(15)
CLHI ARG4, 16 4(N+1)
       F-27
* *  
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  
CLHI 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) K02-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 K02-DAVE S-FEB 14, 1978  
F-29
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-IU (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

P3CK2ARG LM ARG1,0(15)
CLE1 ARG1,12 4(N+1)
B E  PBE001
LHI ERROR,+1

T-30
BAL ERLINK, P9ERROR
B P8X001

P8E001 EQU *
L BUF1ADD,0(ARG2)
L TOWHON,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

F-31
STH  TEMP,ERRLEN
L    TEMP,ERRTAB+4(ERROR)
ST    TEMP,ERRADD
ST    COMMD,U(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 XI 15,2
BZS P10E001
L 15,LASTREG
AIS 15,2
ST 15,LASTREG
BS P10X001

P10E001 EQU *
L 15,LASTREG

P10X001 EQU *
BK LINK
TITLE LOCAL DATA AREAS
*
*
REGISTER SAVE AREA
*

REGSAVE DAS 15
LASTREG DAS 1
TEMPSAV DAS 16
*
ERROR MESSAGE TABLE
*
ALIGN 4
ERRTAB DAC MESS1LEN,MESS1,MES2LEN,MES3
DAC MESS3LEN,MES4LEN,MES4
*
ERROR MESSAGES
*
ALIGN 4
MESS1 DC C'XXXX-54 WRONG NUMBER OF ARGUMENTS'
MESS1LEN EQU *-MESS1
ALIGN 4
MESS2 DC C'XXXX-55 ILLEGAL REMOTE-ID LENGTH'
MESS2LEN EQU *-MESS2
ALIGN 4
MESS3 DC C'XXXX-56 ILLEGAL LOCAL-ID LENGTH'
MESS3LEN EQU *-MESS3
ALIGN 4
MESS4 DC C'XXXX-57 ILLEGAL STATUS LENGTH'
MESS4LEN EQU *-MESS4
TITLE SVC BLOCKS
*
GET ID SVC
*
ALIGN 4
GETIDSV HC 0
DB 1
COMMAND WHAT EVER
GETSTAT DC n'0'
STATUS OF GET ID
LOCALID DAC 0
ID IN THE FORM CNTP
DAC 0
- NUMBER INDICATES PASCAL
*
SEND SVC

F-33
<table>
<thead>
<tr>
<th>ALIGN</th>
<th>SENDSVC</th>
<th>DB</th>
<th>DB</th>
<th>DS</th>
<th>DAC</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>COMMAND</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>A(SENDSCB)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>WHAT EVER</td>
<td></td>
<td></td>
<td></td>
<td>ADDRESS OF SCB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td>NOT USED</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALIGN</th>
<th>RECEIVESVC</th>
<th>DB</th>
<th>DB</th>
<th>DS</th>
<th>DAC</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>COMMAND</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>A(RECVSCB)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>WHAT EVER</td>
<td></td>
<td></td>
<td></td>
<td>ADDRESS OF SCB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td>NOT USED</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALIGN</th>
<th>CONNSVC</th>
<th>DB</th>
<th>DB</th>
<th>DS</th>
<th>DAC</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>COMMAND</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>A(CONNSCB)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>WHAT EVER</td>
<td></td>
<td></td>
<td></td>
<td>ADDRESS OF SCB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td>NOT USED</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALIGN</th>
<th>DISCONNECTSVC</th>
<th>DB</th>
<th>DB</th>
<th>DS</th>
<th>DAC</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>COMMAND</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>A(DISCSSCB)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>WHAT EVER</td>
<td></td>
<td></td>
<td></td>
<td>ADDRESS OF SCB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td>NOT USED</td>
<td></td>
</tr>
<tr>
<td>F-34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* 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
SENDCALLID DS 4
SENDCMBF DAC 0
SENDTBDF DAC 0
SENDLCM DC H'0'
SENDLNTX DC H'0'
SENDCONDN DC H'60'
SENDSCOU DS 2
SENDCMD DC H'2'
*
SPACE 3
*
RECLIVE SCB
*
ALIGN 4
RECVSCB EQU *
RECVRMID DS 4
RECVLCID DS 4
RECVCMBF DAC 0
RECVTBDF DAC 0
RECVLCM DC H'0'
REMOTE ID
LOCAL ID
COMMAND TEXT BUFFER
DATA TEXT BUFFER
LENGTH OF COMMAND TEXT
LENGTH OF DATA TEXT
CODE, IN (MESSAGE COMMAND)
CODE, OUT MESSAGE RETURN STATUS
MESSAGE COMMAND CLASS

F-35
RECVLEN TX DC H'U'
RECVCLUD DC H'63'
RECVCLUDU DC H'U'
RECVCLUD DC H'3'
*  
   EJECT  
  
   *  
   CONNECT SCB  
   *  
   ALIGNSCB EQU *  
   CONNRMID DS 4  
   CONNLCID DS 4  
   CONNMBF DAC 0  
   CONNLNF DAC 0  
   CONNNLCM DC H'0'
   CONNLNTX DC H'0'
   CONNGCIDN DC H'3'
   CONNCGDOU DS 2  
   CONNCMD DC H'0'
   SPACESPACESPACE  
   *  
   DISCONNECT SCB  
   *  
   ALIGNSCB EQU *  
   DISCGCID DS 4  
   DISCGCID DS 4  
   DISCGMBF DAC 0  
   DISGCTBF DAC 0  
   DISGCLNCM DC H'0'
LEN OF DATA TEXT
CODE IN (MESSAGE COMMAND)
CODE OUT (MESSAGE RETURN STATUS)
MESSAGE COMMAND CLASS
REMT ID  
LOC ID  
CMND TEXT BUFFER  
DATA TEXT BUFFER  
LEN OF CMND TEXT  
LEN OF DATA TEXT  
CODE IN (MESSAGE COMMAND)
CODE OUT (MESSAGE RETURN STATUS)
MESSAGE COMMAND CLASS
REMOTE ID  
LOCAL ID  
CMND TEXT BUFFER  
DATA TEXT BUFFER  
LEN OF CMND TEXT  
LEN OF DATA TEXT
F-36
| DISCLNTC DC   | H'0'  | LENGTH OF DATA TEXT |
| DISCCODIN DC  | H'5'  | CODE.IN (MESSAGE COMMAND) |
| DISCCODOUT 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 KEFEN PIC X(4).
77 LINKPATH PIC X(8).
77 CTRL PIC X(9).
77 VELEM PIC X(60).
77 VARREA PIC X(82).
77 ENDP PIC X(4) VALUE 'END'.

* MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
FROMID PIC X(4) VALUE 'AAAUW'.
77 TOID PIC X(4) VALUE 'ABRH'.
77 MSTAT PIC 9(5) COMP.

* SYSTEM USER COUNTER:

* INITIATED 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 BFUCTION 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-Routines.
* INITIALIZE MESSAGE SYSTEM COMMUNICATION:
ESTAB-TASK-ID.
CONNECT-INTO-SYSTEM.
CALL 'CONN' USING
   TOID FROMID MSTAT.

G-2
IF MSTAT NOT EQUAL ZERO
DISPLAY *BINT ABORT-- BAD CONNC! Upon CRT
STOP RUN.

*****REPEAT

MAIN-LOOP.
* RECEIVE REQUEST FROM USER:
HELLO-THERE,
CALL *RECV* USING
TOID BUFI 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 BFUNCTI0N TO FUNCTION,
MOVE BFILNAM3 TO FILE-NAM3.
MOVE BVAREA TO VAREA.

BINT-EXEC-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 BCCTRL TO CTRL,
   MOVE BVELEM TO VELEM,
   CALL 'DATBAS' USING
       FUNCTION STAT FILE-NAME REFER
       LINKPATH CTRL VELEM VARLA
       ENDP,
   MOVE REFER TO BREFER,
   GO TO BINT-REPACK.

Bv2.
   MOVE BCCTRL TO CTRL,
   MOVE BVELEM TO VELEM,
   CALL 'DATBAS' USING
       FUNCTION STAT FILE-NAME CTRL
       VELEM VARLA ENDP,
   GO TO BINT-REPACK.

Bv3.
   MOVE BREFER TO REFER,
   MOVE BVELEM TO VELEM,
   CALL 'DATBAS' USING
       FUNCTION STAT FILE-NAME REFER
VELEN VARLA ENDP.
MOVE REFER TO BREFER.
GO TO BINT-REPACK.

B4V.
MOVE BREFER TO REFER.
CALL 'DATBAS' USING
   FUNCTION STAT REFER
   VAREA ENDP.
MOVE REFER TO BREFER.
GO TO BINT-REPACK.

B5V.
MOVE DCTRL TO CTRL.
CALL 'DATBAS' USING
   FUNCTION STAT FILE-NAME
   CTRL VAREA ENDP.
GO TO BINT-REPACK.

B6V.
CALL 'DATBAS' USING
   FUNCTION STAT VAREA ENDP.
* IF FUNCTION = 'SINON' AND STAT = '****'
  * THEN ADD 1 TO ACTIVE-UPLOAD,
* IF FUNCTION = 'SINOF' AND STAT = '****'
  THEN SUBTRACT 1 FROM ACTIVE-UPLOAD,
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.
G-6
***************
*
* APPENDIX H *
*
***************
IDENTIFICATION DIVISION.
PROGRAM-ID. BINT.
REMARKS. 'BINT' IS THE INTERFACE PROGRAM THAT
DIRECTLY ACQUIRES 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.

* DATBASE 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:

* 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 CVELEM PIC X(60).
   03 BVAREA PIC X(82).

PROCEDURE DIVISION.

INIT-Routine.
* INITIALIZE MESSAGE SYSTEM COMMUNICATION:
  ESTABLISH TASK-ID,
  CONNECT-INTO-SYSTEM.
  CALL 'CONN' USING
    TOID FROMID MSTAT.

H=2
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 BFUNCTO 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.

* DATBASE CALL ROUTINES:

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

BV3.
    MOVE BREFER TO REFER.
    MOVE BVELEM TO VELEM.
    CALL 'DATBASE' USING
        FUNCTION STAT FILE-NAME REFER
VELEH VARLA ENDP.
MOVE REFER TO UREFER.
GO TO BINT-REPACK.

BV4.
MOVE UREFER TO REFER.
CALL 'DATBAS' USING
   FUNCTION STAT REFER
   VAREA ENDP.
MOVE REFER TO UREFER.
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-SINFO.

* 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-SINFO.
CALL 'DISC' USING TOID MSTAT,
IF MSTAT NOT EQUAL ZERO
DISPLAY 'BINT ABORT-- BAD DISCONNECT' UPON CRT.

H-6
IDENTIFICATION DIVISION.
PROGRAM-ID. BINT.
REMARKS. *BINT* IS THE INTERFACE PROGRAM THAT
DIRECTLY ACCSESSES THE BACKEND DBMS. IT RECEIVES
THE CALLING PARAMETERS VIA THE PACKED PARAMETER
BUFFERS BUF1 AND BUF2, DETERMINES 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 VELEN PIC X(64),
77 VARLA PIC X(82),
77 ENDP PIC X(4) VALUE 'END,'.

* MESSAGE SYSTEM PARAMETERS:

I-1
PIC 9(9) COMP-3 VALUE 414254500.
PIC 9(9) COMP-3 VALUE 424104010.
PIC 9(8) COMP SYNC VALUE 0.
PIC X(128).
PIC 9(8) COMP SYNC VALUE 0.
PIC X(128).
PIC 9(8) COMP SYNC.
PIC 9(8) COMP SYNC.
PIC 9(8) COMP SYNC.
PIC 9(8) COMP SYNC.
PIC 9(8) COMP SYNC.
PIC 9(8) COMP SYNC.
PIC 99 VALUE 1.
PIC X(5).
PIC X(4).
PIC X(4).
PIC X(4).
PIC X(8).
PIC X(9).
PIC X(60).
PIC X(62).
PIC X(1).
PROCEDURE DIVISION.
INIT-Routine.

* INITIALIZE MESSAGE SYSTEM COMMUNICATION.

ESTAB-TASK-Id.
CALL 'HI1SWNTS' USING FROMID MSTAT.
IF MSTAT NOT EQUAL ZERO
   DISPLAY 'BINT ABORT-- BAD TASK ID' UPON CONSOLE
   STOP RUN.

CONNECT-INTO-SYSTEM.
CALL 'Hi1SCONN' USING
   FROMID TOID MSMODE OUTCMD OUTCMDLTH
   INCMD INCMDLTH MSTAT ECB.
   CALL 'Hi1SCHCk' USING ECB.
   IF MSTAT NOT EQUAL ZERO
   DISPLAY 'BINT ABORT-- BAD CONNECT' UPON CONSOLE
   CALL 'Hi1SPURGE' USING FROMID FROMID MSMODE MSTAT ECB
   STOP RUN.

MAIN-LOOP.

* RECEIVE REQUEST FROM USER;

HELLO-THERE.
   MOVE 177 TO BUF1LTH.
   CALL 'Hi1SRECv' USING
      MSMODE FROMID TOID BUF2 BUF2LTH
      BUF1 BUF1LTH MSTAT ECB.
      CALL 'Hi1SCHCk' USING ECB.
      IF MSTAT = 32 THEN
         DISPLAY 'BINT: TIMED OUT-- WILL RECV AGAIN'
      UPON CONSOLE
      GO TO HELLO-THERE

I-3
ELSE
IF MSTAT NOT EQUAL ZERO
   DISPLAY 'BINT ABORT--BAD RECEIVE' UPON CONSOLE
   GO TO SYS-SINOF.

BINT-UNPACK.
   CALL 'TRANSE' USING BUF1 BUF1TH.
   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 BVELEN TO VELEN.
   CALL 'DATBAS' USING
   FUNCTION STAT FILE-NAME REFER
   LINKPATH CTRL VELEN VAREA
   ENUP.
   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.

BVT.
    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 HVAREA.

*   RETURN INFORMATION TO USER.
*   GOOD-BYE.
*   ESTABLISH LENGTH OF BUFFERS.
*   MOVE 177 TO BUF1LGTH.
    CALL 'TRANSA' USING BUF1 BUF1LGTH,
    CALL 'MSSEND' USING
        MSMOVE FROM10 TO10 BUF2 BUF2LGTH
        BUF1 BUF1LGTH MSTAT ECB.
    CALL 'MSCHECK' USING ECB.

I-6
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 FROMM TOID MMSMODE MSTAT ECB,
  CALL *MDSKETCH USING ECB,
  IF MSTAT NOT EQUAL ZERO
    DISPLAY 'BINT ABORT-- BAD DISCONNECT' UPON CONSOLE.
  CALL *MSPURGE USING FROMM FROMID MSTAT ECB.
STOP RUN.
//LKED,SYSLMOD DD DSN=US2L1,LOADLIB,DISP=OLD
//LKED,TOTLIB DD DSN=US2L1,TOTAL,LINELIB,DISP=SHR
//LKED,MSLIB DD DSN=DSQ76,TXTLIB,DISP=SHR
//LKED,SYSSIN DD *
  INCLUDE TOTLIB(DATBASE)
  INCLUDE TOTLIB(TRAN)
  INCLUDE MSLIB(MSCALL,SYSCALL)
  LIBRARY *(DATBASEXT)
  NAME BINT(K)
I-7
***************
*               *
*      APPENDIX J    *
*               *
***************
$IFDEF @1:SCOPY
* CALLING FORMAT: COBEBSTAB INTERFACE NAME,APPLICATION NAME,
* ASSEMBLER INTERFACE NAME,       *
$NO
$EXIT;ENDC
$IFNX @1.OBJ:SCOPY
* FILE @1.OBJ DOES NOT EXIST
$NO
$CLEAR ;$EXIT ;$ENDC
$IFX SYS2:ESTAB,CMD 1DEL SYS2:ESTAB,CMD 1$ENDC
$IFNX @1.TCM
$BUILD SYS2:ESTAB,CMD
ES TA
IN @1.OBJ
IN @2.OBJ
EDIT SYS2:CBO200,LIB/S
IN @3.OBJ
BU TA,@1.TSK
AMAP
END
$ENDB
$ENDC
$IFDEF @1.TSK;IDE @1.TSK;$ENDC
$IFDEF SYS2:ESTAB,SCR 1AL SYS2:ESTAB,SCR,IN,126/4 1$ENDC
$IFDEF SYS2:ESTAB,LST 1DEL SYS2:ESTAB,LST 1$ENDC
AL SYS2:ESTAB,LST,IN,126/4
LO TET32
AS 1,@1.OBJ
AS 3,PR:
TE 4,IN,126
$IFDEF @1.TCM;AS 5,@1.TCM;$ENDC
$IFDEF @1.TCM;AS 5,SYS2:ESTAB,CMD;$ENDC
AS 7,CON:
ST
J-1
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 7/32 INTEPTASK 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.

K-1
WORKING-STORAGE SECTION.

* MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
  77 FROMID PIC X(4) VALUE 'ABRH'.
  77 TOID PIC X(4) VALUE 'AAUG'.
  77 MSTAI 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 VELEMNTS PIC X(60).
  03 VAREA PIC X(82).

LINKAGE SECTION.

* INPUT PARAMETERS

  77 BFUNCTION PIC X(5).
  77 BSTAI PIC X(4).
  77 BFILL-NAME PIC X(4).
  77 BREFER PIC X(4).
  77 BLINKPATH PIC X(6).
  77 BCTRL PIC X(9).
  77 BVELEMNTS PIC X(60).

K-2
PROCEDURE DIVISION USING
  BFUNCTI ON BSTAT BFIL E-NAM E BREFER
  BL INKPA TH DCTRL BVELE M BVARKE BEND.

* MOVE PARAMETER ELEMENTS INTO BUFFER
  MOVE BFUNCTI ON TO FUNCTION.
  MOVE BFIL E-NAM E TO FILE-NAM E.
  MOVE BVARKE TO VARKE.

HINT-CALL-SELECT.
* DETERMINE TYPE OF CALL

IF FUNCTION = 'READV' OR
  FUNCTION = 'READK' OR
  FUNCTION = 'READD' OR
  FUNCTION = 'WRITV' OR
  FUNCTION = 'DELVD' OR
  FUNCTION = 'ADDVC' OR
  FUNCTION = 'ADDOA' OR
  FUNCTION = 'ADOVB' THEN GO TO CALL-HINT-V001

ELSE IF FUNCTION = 'READM' OR
  FUNCTION = 'WRITM' OR
  FUNCTION = 'ADDM' OR
  FUNCTION = 'DELM' THEN GO TO CALL-HINT-V002

ELSE IF FUNCTION = 'RDNUX1'
  THEN GO TO CALL-HINT-V003
ELSE IF FUNCTION = 'MARKL' OR
FUNCTION = 'QUIET'
THEN GO TO CALL-HINT-VOO4

ELSE IF FUNCTION = 'KULOC'
THEN GO TO CALL-HINT-VOO5

ELSE IF FUNCTION = 'SINON' OR
FUNCTION = 'SINOJ'
THEN GO TO CALL-HINT-VOO6

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

*******************************
*
* CALL-HINT-VOO1:
* READV
* READR
* READD
* WR1TV
* DEJVD
* ADDVC
* ADDVA
* ADDVB
*
*******************************

CALL-HINT-VOO1.
MOVE 1 TO CALLER,
MOVE BREFEK TO REFER.
MOVE BLINKPATH TO LINKPATH.
MOVE BCHRL TO CTRL-FIELD.
MOVE BVELEM TO VELEMENTS.
PERFORM CALL-BINT THRU CALL-EXIT.
MOVE REFER TO REREFER.
GO TO HINT-RETURN.

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

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

*******************************
*  CALL-HINT-V003:
  RUNX1
 *
*******************************

K-5
CALL-HINT-V003.
   MOVE 3 TO CALLER,
   MOVE REFER TO REFER,
   MOVE BVLEEM TO VELEMENTS,
   PERFORM CALL-BINT THRU CALL-EXIT.
   MOVE REFER TO REFER,
   GO TO HINT-RETURN.

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

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

***************
*   CALL-HINT-V005:
*       RWLOC
*   ***************
CALL-HINT-V005.
MOVE 5 TO CALLER,
MOVE BCTRL TO CTRL-FIELD,
PERFORM CALL-HINT 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-HINT THRU CALL-EXIT,
   IF FUNCTION = 'SINOF'
   PERFORM SYS-SINOF THRU SYS-SINOF-EXIT;
GO TO HINT-RETURN.

HINT-RETURN.
* RETRANFER PARAMETER VALUES
   MOVE STAT TO BSTAT,
   MOVE VARFA TO BVAREA,
HINT-EXIT,
   EXIT PROGRAM.

* SUBROUTINES FOR MESSAGE SYSTEM COMMUNICATION:

K-7
SYS-SINON.
* CONNECT INTO MESSAGE SYSTEM:
CONNECT-INTO-SYSTEM.
   CALL 'CONNECT' 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-BINT1.
* CALL THE MESSAGE SYSTEM FOR BUFFER IO:
CALL-BINT1.
   CALL 'SEND' USING TOID BUF1
      MSTAT.
   IF MSTAT = 29 THEN DISPLAY
      *HINT: TIMEOUT-- 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 'RCLV' USING TOID BUF1

K-8
MSTAT.
IF MSTAT = 32 THEN DISPLAY
'HINT: TIMED OUT-- WILL RCV AGAIN'
UPON CRT
GO TO CALL-BINT2
ELSE
IF MSTAT NOT EQUAL ZERO PERFORM MSYERROR GO TO HINT-EXIT.
CALL-EXIT, EXIT.

MSYERROR.
4 FATAL MESSAGE SYSTEM ERROR!
DISPLAY 'HINT: MESSAGE SYS ERROR-- REQUEST ABORTED UPON CRT.
MOVE 'XXXX' TO STAT.
IDENTIFICATION DIVISION.
PROGRAM-ID. HINT.

REMARKS. HINT* IS THE HOST INTERFACE DBMS PROGRAM.
THE UATBAS PARAMETERS ARE PACKED INTO DATA
BUFFERS: BUF1*, WHICH ARE THEN PASSED
TO 'DINT*, THE BACKEND DBMS INTERFACE.
'DINT* EXECUTES THE PHYSICAL CALL AND RETURNS THE RESULTS.

* THIS VERSION DESIGNED FOR 6/32 TO 8/32 INTERTASK COMMUNICATION *

ACTIVE DATBAS PARAMETERS USED BY THE PROGRAM ARE:

FUNCTION
STAT
FILE-NAME
REFER
LINAPATH
CTRL-FIELD
V lang EN V ELEMENTS
VAREA.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER, INTERDATA MODEL-6-32.
OBJECT-COMPUTER, INTERDATA MODEL-8-32.
INPUT-OUTPUT SECTION.

DATA DIVISION.
WORKING-STORAGE SECTION.

* MESSAGE SYSTEM PARAMETERS FOR IRELAND'S INTERFACE:
  77 FRUID PIC X(4) VALUE 'ABRU'.
  77 TUID PIC X(4) VALUE 'ABUQ'.
  77 HSTAT 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).
   05REFERRED PIC X(4).
   05 LINKPATH PIC X(8).
   05 CTRL-FIELD PIC X(9).
   05 VELEMNTS PIC X(60).
   05 VARIA PIC X(62).

LINKAGE SECTION.

* INPUT PARAMETERS

  77 BFUCTION 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 BVELLA PIC X(60).
77 BVAREA PIC X(32).
77 BEND PIC X(4).

PROCEDURE DIVISION USING
  BFUNCTION RSTAT BFIL-NAME BREFER
  BLINKPATH BCTL BKBEH BVAREA BEND.

* MOVE PARAMETER ELEMENTS INTO BUFFER
  MOVE BFUNCTION TO FUNCTION,
  MOVE BFIL-NAME TO FILN-NAME,
  MOVE BVAREA TO VARA.

HINT-CALL-SELECT,
* DETERMINE TYPE OF CALL

  IF FUNCTION = 'READV' OR
     FUNCTION = 'READR' OR
     FUNCTION = 'READA' OR
     FUNCTION = 'WRITV' OR
     FUNCTION = 'WRITR' OR
     FUNCTION = 'ADDVA' OR
     FUNCTION = 'ADDVB'
     THEN GO TO CALL-HINT-VOO1
  ELSE IF FUNCTION = 'READM' OR
     FUNCTION = 'WRITM' OR
     FUNCTION = 'ADD-M' OR
     FUNCTION = 'DELM'
     THEN GO TO CALL-HINT-VOO2
  ELSE IF FUNCTION = 'RDNXT'
     THEN GO TO CALL-HINT-VOO3
ELSE IF FUNCTION = 'MARKL' OR
   FUNCTION = 'QUIET'
   THEN GO TO CALL-HINT-V004

ELSE IF FUNCTION = 'KULOC'
   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                       *
* KLAUK                       *
* READS                       *
* WRTIV                       *
* DL-LVD                      *
* AUVVC                      *
* AUVVA                      *
* AUVyb                      *

******************************

CALL-HINT-V001.
   MOVE 1 TO CALLER.
   MOVE KREFE TO KREFR.

L-4
MOVE BLINKPATH TO LINKPATH.
MOVE BCTRL TO CTRL-FIELD.
MOVE BCELEM TO VLELEMENTS.
PERFORM CALL-BINT THRU CALL-EXIT.
MOVE REFER TO HREFER.
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 BCELEM TO VLELEMENTS.
PERFORM CALL-BINT THRU CALL-EXIT.
GO TO HINT-RETURN.

************************************************
*
* CALL-HINT-V003:
* RunX1
*
************************************************
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
* QUIC
*
**************

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:
* RULUC
*
**************

L-6
CALL-HINT-VOUS.
MOVE 5 TO CALLER.
MOVE BCRL TO CTRL-FIELD.
PERFORM CALL-BINT THRU CALL-EXIT.
GO TO HINT-RETURN.

**********************
* CALL-HINT-VOUS:
* SINON
* SINOF
*
**********************

CALL-HINT-VOUS.
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 VARLA TO BVAREA.
HINT-EXIT.
EXIT PROGRAM.

* SUBROUTINES FOR MESSAGE SYSTEM COMMUNICATION:

L-7
SYS-SINON.
  * CONNECT INTO MESSAGE SYSTEM:
  CONNECT-INTO-SYSTEM.
  CALL 'CONN' USING
   TOID FROM ID 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 TO:
  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:

L-0
CALL 'RECV' USING UID 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 ASYSERROR
    GO TO HINT-EXIT.
CALL-EXIT. EXIT.

ASYSERROR.
* FATAL MESSAGE SYSTEM ERROR:
    DISPLAY 'HINT: MESSAGE SYS ERROR-- REQUEST ABORTED' UPON CRT.
    MOVE 'XXXA' TO STAT.
**************
*
* APPENDIX A *
*
**************
IDENTIFICATION DIVISION.
PROGRAM-ID. HINT.
REMARKS. *HINT* IS THE HOST INTERFACE DBMS PROGRAM.
THE DATABASE PARAMETERS ARE PACKED INTO DATA
BUFFERS: BUFI, WHICH ARE THEN PASSED
TO *HINT*, THE BACKEND DBMS INTERFACE.
*HINT* EXECUTES THE PHYSICAL CALL AND RETURNS THE RESULTS.

* THIS VERSION DESIGNED FOR 0/32 TO 370 INTERTASK COMMUNICATION
* 

ACTIVE DATABASE PARAMETERS USED BY THE
PROGRAM ARE:

FUNCTION
STAT
FILE-NAME
KEEP
LINKPATH
CTRL-FIELD
VELEMENTS
VAR-A

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. INTERDATA MODEL-0-32.
OBJECT-COMPUTER. INTERDATA MODEL-0-32.
INPUT-OUTPUT SECTION.

DATA DIVISION.

M-1
WORKING-STORAGE SECTION.

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

* PARAMETER BUFFER FOR INTERFACE CALL

01 BUFI.
   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(32).

LINKAGE SECTION.

* INPUT PARAMETERS

77 BUFICTION  PIC X(5).
77 USTAT      PIC X(4).
77 BUFILENAME PIC X(4).
77 UREFER     PIC X(4).
77 ULINKPATH  PIC X(3).
77 UCRTL      PIC X(9).
77 BUFILENAME PIC X(60).
77 BVAREA  PIC X(82).
77 BEND    PIC X(4).

PROCEDURE DIVISION USING
  BFUNCTION BSTAT BFIL-E-NAME BREFER
  BLINKPATH BCTRL BLEVEL BVAREA BEND.

* MOVE PARAMETER ELEMENTS INTO BUFFER
  MOVE BFUNCTION TO FUNCTION,
  MOVE BFIL-E-NAME TO FILE-NAME,
  MOVE BVAREA TO VAREA.

HINT-CALL-SELECT.
* DETERMINE TYPE OF CALL

  IF FUNCTION = 'READV' OR
  FUNCTION = 'READR' OR
  FUNCTION = 'READU' OR
  FUNCTION = 'WRITV' OR
  FUNCTION = 'WRLTV' OR
  FUNCTION = 'DLLVD' OR
  FUNCTION = 'ADDVC' OR
  FUNCTION = 'ADDVA' OR
  FUNCTION = 'ADDVB'
  THEN GO TO CALL-HINT-V001

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

  ELSE IF FUNCTION = 'RDNX1'
  THEN GO TO CALL-HINT-V003
ELSE IF FUNCTION = 'MARKL' OR
    FUNCTION = 'QUIET'
    THEN GO TO CALL-HINT-V004

ELSE IF FUNCTION = 'KULOC'
    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 = 'CLOSH'
    THEN GO TO CALL-HINT-V008

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

******************************
*
* CALL-HINT-V001:
*
* READV
* READR
* READU
* WRITV
* DELVD
* ADJVC
* ADJVA
* ADJVB
CALL-HINT-V001.
    MOVE 1 TO CALLER.
    MOVE REFER TO REFER.
    MOVE BLINKPATH TO LINKPATH.
    MOVE BCtrl TO Ctrl-Field.
    MOVE VELEM TO VELEMENTS.
    PERFORM CALL-HINT THRU CALL-EXIT.
    MOVE REFER TO REFER.
    GO TO HINT-RETURN.

CALL-HINT-V002.
    READ
    WRITN
    ADD-N
    DEL-N

CALL-HINT-V002.
    MOVE 2 TO CALLER.
    MOVE BCtrl TO Ctrl-Field.
    MOVE VELEM TO VELEMENTS.
    PERFORM CALL-HINT THRU CALL-EXIT.
    GO TO HINT-RETURN.

**************
CALL-HINT-V003:
        RUNX1

CALL-HINT-V003:
        MOVE 3 TO CALLER.
        MOVE REFER TO REFER.
        MOVE BVELEM TO VELEMETS.
        PERFORM CALL-HINT THRU CALL-EXIT.
        MOVE REFER TO REFER.
        GO TO HINT-RETURN.

CALL-HINT-V004:
        MARKL
        QUIET

CALL-HINT-V004:
        MOVE 4 TO CALLER.
        MOVE REFER TO REFER.
        PERFORM CALL-HINT THRU CALL-EXIT.
        MOVE REFER TO REFER.
        GO TO HINT-RETURN.
CALL-HINT-V005:
  RGLOD

************

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-SINOM 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 LINKPATH TO LINKPATH.
PERFORM CALL-BINT THRU CALL-EXIT.
PERFORM SYS-SINOF THRU SYS-SINOF-EXIT.
GO TO HINT-RETURN.

***************

* CALL-HINT-V008:
  OPENN
  CLUSN

***************

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-SINOR.
* CONNECT INTO MESSAGE SYSTEM:
CONNECT-INTO-SYSTEM.
   CALL 'CONN' USING
       TOID FROM0 MSTAT,
   IF MSTAT NOT EQUAL ZERO
       PERFORM ASYSERRORK
       GO TO HINT-EXIT.
SYS-SINOR-EXIT. EXIT.

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

CALL-BINTI.
* CALL THE MESSAGE SYSTEM FOR BUFFER 10:
CALL-BINTI1,
   CALL 'SEND' USING TOID BUF1
       MSTAT.
   IF MSTAT = 29 THEN DISPLAY
       'HINT: TIMED OUT-- WILL SEND AGAIN'
       UPON CRT
       GO TO CALL-BINTI1
   ELSE
       IF MSTAT NOT EQUAL ZERO PERFORM ASYSERRORK
       GO TO HINT-EXIT.
CALL-BINT2,
* RECEIVE REPLY FROM BACK END.
CALL 'RECV' USING TOID BUF1
  HSTAT.
IF HSTAT = 32 THEN DISPLAY
  'HINT: TIMED OUT -- WILL RECV AGAIN'
  UPON CR1
  GO TO CALL-BINT2
ELSE
  IF HSTAT 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 CR1.
  MOVE 'XXXXX' 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.
FILL-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 CARRIAGECONTROL 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-
   *ADDEPPEOPTELEPHPEOPHIREPEOPBIRTPEOPSOSEND*.','. 

77 PLKS-DATA
   PIC X(60) VALUE 'PERSPEOPLEPERSSTRTLPERS
   EXPRPERSDATAEND.'
77 NUMBER
77 DUMMY1
77 DUMMY2
77 DUMMY3
77 DUMMY4
77 REFERENCE
77 LK-PATH
77 FIRST-READV
  01 PEOP-IN.
     05 PEOP-NUM
     05 FILLER
     05 PEOP-NAM
     05 FILLER
     05 PEOP-ADR
     05 FILLER
     05 PEOPL-TEL
     05 FILLER
     05 PEOPL-HIR
     05 FILLER
     05 PEOP-BRT
     05 FILLER
     05 PEOP-SOC
  01 PERS-IN.
     05 PLKS-EMP-NO
     05 FILLER
     05 PLKS-DATE
     05 FILLER
     05 PLKS-CODE
     05 FILLER
     05 PLKS-PERSONAL
  01 PEOPL-RECODO.
     05 XNUMBER
     PIC X(6).
     PIC X(5) VALUE SPACES,
     PIC X(6).
     PIC X(5) VALUE SPACES,
     PIC X(6).
     PIC X(5) VALUE SPACES,
     PIC X(2),
     PIC X(5) VALUE SPACES,
     PIC X(21),
     PIC X(6),
     PIC X(9),
     PIC X(4),
     PIC X(8) VALUE SPACES,
     PIC X(9),
     PIC X(8) VALUE SPACES,
     PIC X(60) VALUE SPACES,
     PIC X(4),
     PIC X(8) VALUE 'PLOPLKPE',
     PIC X(3),
     PIC X(6),
     PIC X(5) VALUE SPACES,
     PIC X(25),
     PIC X(5) VALUE SPACES,
     PIC X(20),
     PIC X(5) VALUE SPACES,
     PIC X(10),
     PIC X(5) VALUE SPACES,
     PIC X(6),
     PIC X(5) VALUE SPACES,
     PIC X(6),
     PIC X(5) VALUE SPACES,
05 NAME
05 XADDRESS
05 TELEPHONE
05 HIRE-DATE
05 BIRTH-DATE
05 SOC-NUMB
01 PERS-RECORD.
  05 EMP-NO
  05 DATE-STARTED
  05 EXPERIENCE-CODE
  05 PERSONAL-DATA
  05 FILLER
01 PRINT-LINE.
  05 FILLER
  05 DATA1
  05 FILLER
  05 DATA2
01 PRINT-COMMAND.
  05 FILLER
  05 COMM
  05 FILLER
  05 NUMB
01 INPUT-COMMAND.
  05 COMMAND
  05 NO-TIMES
01 SCHEMA.
  03 TASKID
  03 IDNUMID
  03 ENDED
  03 LOGOPT
  03 REALN-1.
    05 FILE-1
    05 NOUE-1
    05 STAT-1
  PIC X(25).
  PIC X(20).
  PIC X(10).
  PIC X(6).
  PIC X(6).
  PIC X(9).
  PIC X(6).
  PIC X(6).
  PIC X(2).
  PIC X(21).
  PIC X(47) VALUE SPACES.
  PIC X(5) VALUE SPACES.
  PIC X(5) VALUE SPACES.
  PIC X(4).
  PIC X(5) VALUE SPACES.
  PIC X(5) VALUE SPACES.
  PIC X(9).
  PIC X(5).
  PIC 9(3).
  PIC X(8) VALUE 'LOADPEOP'.
  PIC X(6) VALUE 'PERSON'.
  PIC X(6) VALUE 'UPDATE'.
  PIC 9(3) VALUE 'NL'.
  PIC X(4) VALUE 'PEOP'.
  PIC X(4) VALUE 'PRIV'.
  PIC X(4) VALUE SPACES.

N-3
03 R0M-2.
  05 F1LE-2.
  05 M0DE-2.
  05 STAT-2.
  03 F1LLER.
  03 F1LLER.

PROCEDURE DIVISION.
SIGN-ON.

* SIGN ON TO THE DATABASE 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 CLOSL-EM.

GET-COMMAND.
*
MAIN LOOP - ACCEPTS USER DATABASE 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.

N-4
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 'READ-M' OR COMMAND IS EQUAL TO 'DEL-M' OR COMMAND IS EQUAL TO 'WRIT-M' 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 23 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 SCC-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-READ FROM CRT.
MOVE * * TO CARRIAGECON.
MOVE FIRST-READ TO INDATA.
DISPLAY PLOP-PERS-OUT UPON CRT.
WRITE PLOP-PERS-OUT AFTER ADVANCING 2 LINES.
IF FIRST-READ 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 'READY' AND REFERENCE IS NOT
EQUAL TO 'END,' THEN MOVE EMP-NO TO PERS-EMP-NO
MOVE DATE-STARTED TO PERS-DATE

N=7
MOVE EXPERIENCE-CODE TO PERS-CODE
MOVE PERSONAL-DATA TO PERS-PERSONAL
MOVE * TO CARRIAGECON
MOVE PERS-IN TO INDATA
DISPLAY PEOPEP-PERS-OUT UPON CRT
WRITE PEOPEP-PERS-OUT AFTER ADVANCING 2 LINES.

VARIABLE-EXIT. EXIT,
STOP.

* 
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 PEOPEP-PERS-OUT UPON CRT.
WRITE PEOPEP-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 STUDENT 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-VALUES.
CU1 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 LNJP PIC X(4) VALUE 'END'.
77 STUD-DATA PIC X(60) VALUE 'STUDENTNAMESTUDGRADUESTUDYEAREND'.

0-1
77 CRSE-DATA PIC X(60) VALUE 'CRSESTUDCRSENAMECRSE
-NMBRCRSELINOCRSLTOKERCSEHORSCRSEGRADEND.'.
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-LNR 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).

0-2
PROCEDURE DIVISION.

SIGN-ON.

* SIGN ON TO THE DATABASE 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 DATABASE 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-CSRSE-OUT UPON CRT.
WRITE STUD-CSRSE-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 'DELM' 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 'DELV' 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-NUM 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-NUM TO CRSE-STUD-NUM.
MOVE COURSE-NAME TO CRSE-NAME.
MOVE COURSE-NUMBER TO CRSE-NUMBER.
MOVE LINE-NUM TO CRSE-LINE-NUM.
MOVE TAKEN-OR-ENROLLED TO CRSE-TAKEN-OR-ENR.
MOVE HOURS TO CRSE-HOURS.
MOVE GRADE TO CRSE-GRADE.
MOVE * * TO CARRIAGE-CONT.
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-READV FROM CRT.
MOVE * * TO CARRIAGE-CONT.
MOVE FIRST-READV TO INDATA.
DISPLAY STUD-CRSE-OUT UPON CRT.
WRITE STUD-CRSE-OUT AFTER ADVANCING 2 LINES.
IF FIRST-READV IS EQUAL TO 'YES' THEN MOVE 'LKCR' TO REFERENCE.
MOVE COMMAND TO FUNCTION.
MOVE STUD-NUM 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.
CLOS-EN.
*
SIGN OFF OF DATA BASE AND TERMINATE.
*
CALL 'HINT' USING SINOFS STAT DUMMY1 DUMMY1 DUMMY2
DUMMY3 DUMMY4 SCHEMA ENDP.
CLOSE PRINT-OUT.

0-8
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.
  CO1 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 STAI 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','

P-1
05 TELEPHONE
  05 FILLER
  01 PRINT-LINE.
    05 FILLER
    05 DATA1
    05 FILLER
    05 DATA2
  01 PRINT-COMMAND.
    05 FILLER
    05 COMM
    05 FILLER
    05 NUMB
  01 INPUT-COMMAND.
    05 COMMAND
    05 NO-TIMES
  01 PART-DATA.
    03 PARTCTRL
    03 PARTNAME
    03 PARTDESC
    03 ENDS
    03 FILLER
  01 SUPL-DATA.
    03 SUPLPART
    03 SUPLNAME
    03 SUBLADDR
    03 SUPLTELE
    03 ENDS
    03 FILLER

PROCEDURE DIVISION.
SIGN-ON.

*  SIGN ON AND OPEN UP THE DATABASE IMMEDIATELY.
*
OPEN   OUTPUT PRINT-OUT.

P-3
CALL 'HINT' USING TOTAL STAT DUMMY1 DUMMY1 TASKNAME
             UBMOD 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 'CEASL' 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 'DELV2' OR COMMAND IS EQUAL TO 'AD DV2' 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 PNUMBER.
MOVE COMMAND TO FUNCTION.
CALL 'HINT' USING FUNCTION STAT PART DUMMY1 DUMMY2 PNUMBER
           PART-DATA PART-RECORD ENDUP.
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 STELPHONE FROM CRT.
MOVE SNUMBER TO PART-SUPL-NO.
MOVE SNAME TO SUPL-NAME.
MOVE SADDRESS TO SUPL-ADR.

P-6
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 SNUMBLR TO PNUMBER.
CALL 'HINT' USING FUNCTION STAT SUPL REFERENCE LK-PATH
  PNUMBER 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 'READY' AND REFERENCE IS NOT
  EQUAL TO 'END,' THEN MOVE SNUMBER TO PART-SUPL-NO
  MOVE SNAME TO SUPL-NAME
  MOVE SADDRESS TO SUPL-ADK
  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.

P-7
* 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 ENOP.
  CLOSE-EM1.
  CALL 'HINT' USING CLOSM STAT PART DUMMY1 DUMMY2
           DUMMY3 DUMMY4 DUMMY5 ENOP.
  THATS-IT.
  CALL 'HINT' USING DEQUE STAT DUMMY1 DUMMY1 TASKNAME
           DUMMY3 DUMMY4 DUMMY5 ENOP.
  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.