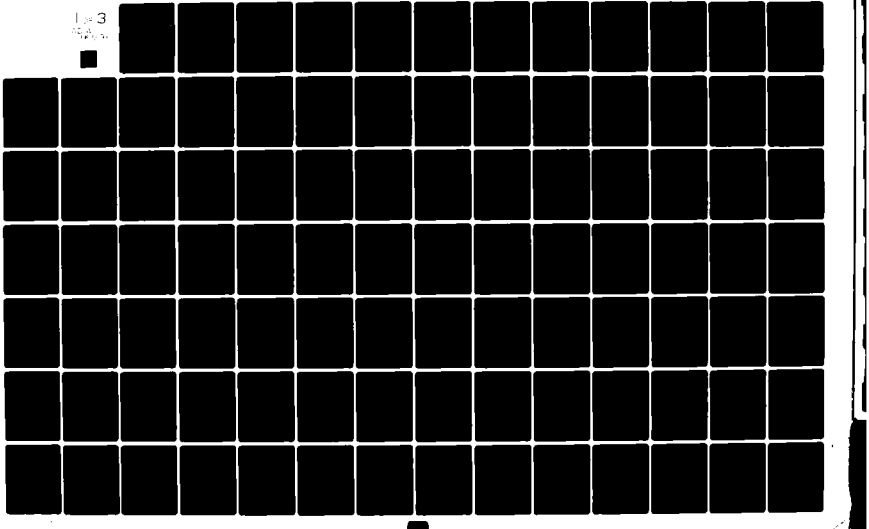
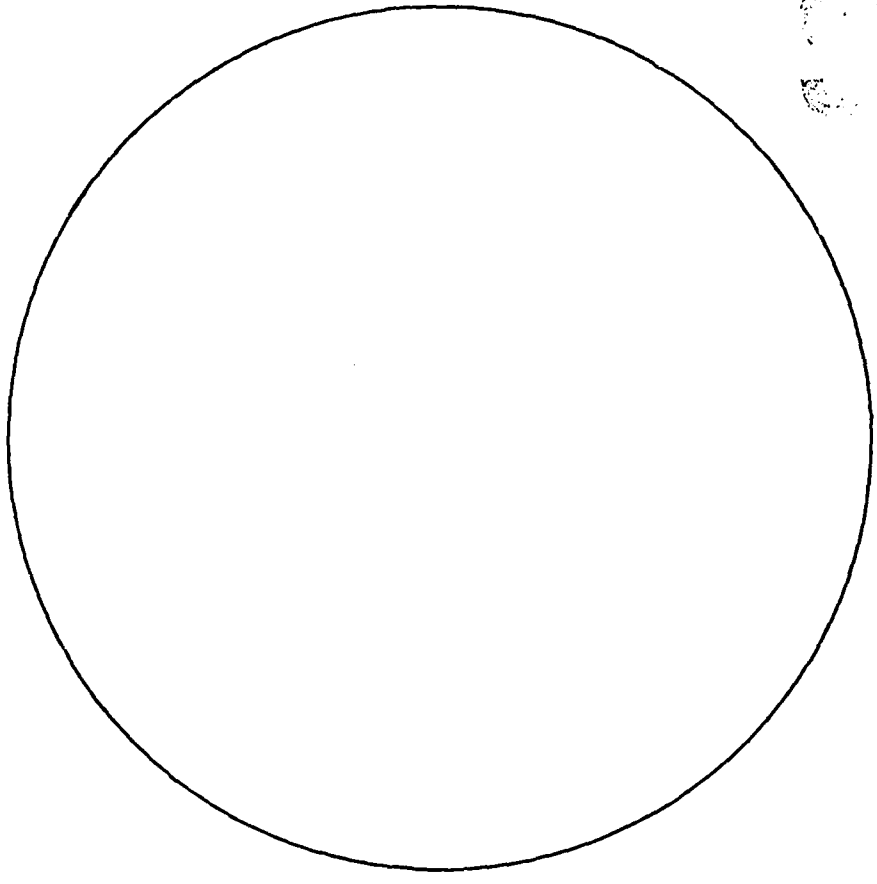
AD-A116 591 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA CEN--ETC F/G 9/2
FSTV: THE SOFTWARE TEST VEHICLE FOR THE FUNCTIONAL HIERARCHY OF--ETC(U)
JAN 82 M HSU N00039-81-C-0663
UNCLASSIFIED CISR-M010-8201-09 NL



(12)



Center for Information Systems Research

Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

(12)

Contract Number N00039-81-0663 (MIT # 91445)
Internal Report Number M010-8201-09
Deliverable Number 1

FSTV:
THE SOFTWARE TEST VEHICLE FOR THE FUNCTIONAL HIERARCHY OF THE INFOPLEX DATABASE COMPUTER

Technical Report # 9

By

Meichun Hsu

January, 1982

Principle Investigator:
Professor Stuart E. Madnick

Prepared for:
Naval Electronics Systems Command
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report #9	2. GOVT ACCESSION NO. AD-A116591	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FSTV - The Software Test Vehicle For The Functional Hierarchy of The Infoplex Database Computer		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER M010-8201-9
7. AUTHOR(s) Meichun Hsu		8. CONTRACT OR GRANT NUMBER(s) N0039-81-c-0663
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sloan School of Management, Massachusetts Institute of Technology Cambridge MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE January 1982 13. NUMBER OF PAGES 270
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

database computer, database management system, software test vehicle, hierarchical systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes in detail the current implementation of the Software Test Vehicle (STV) for the Functional Hierarchy of the INFOPLEX database computer. The purpose of STV is to provide a better understanding of the architecture and functionalities of the INFOPLEX design by emulating its architecture and simulating its functionalities in software. It aims at validating the communication protocols among its various components,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

tightening functional algorithms for database management and data movements, providing behavioral and preliminary performance information concerning the architecture, and serving as a test bed before realizing the design in the hardware prototype.

Implementation of FSTV is based on a preliminary design of the Functional Hierarchy presented in Hsu80. In that design, data base management functions are decomposed into hierarchical levels, each level to be implemented as a level of the Functional Hierarchy. The current version of FSTV is implemented with special attention paid to a richer set of data base capabilities and architectural compatibility.

ABSTRACT

This report describes in detail the current implementation of the software test vehicle (STV) for Functional Hierarchy of INFOPLEX data base computer.

The purpose of STV is to provide a better understanding of the architecture and funcitonalities of the INFOPLEX design by emulating its architecture and simulating its functionalities in software. It aims at validating the communication protocols among its various components, tightening functional algorithms for database management and data movements, providing behavioral and preliminary performance information concerning the architecture, and serving as a test bed before realizing the design in the hardware prototype.

Implementation of FSTV is based on a preliminary design of Functional Hierarchy presented in <Hsu80>. In that design, data base management functions are decomposed into hierarchical levels, each level to be implemented as a level of Functional Hierarchy. The current version of FSTV is implemented with special attention paid to a richer set of data base capabilities and architectural compatibility.

6.0 CONCLUDING REMARKS AND FUTURE DIRECTIONS 92
6.1 Memory Management Level Revisited 92
6.2 Internal Schema Level Revisited 93
6.3 The Entity Level Revisited 95
6.4 Location and Management of Catalogues and Working Buffers 95

REFERENCES 98

APPENDIX 100

Appendix I (Sample Terminal Session) 100
Appendix II (Macro Declarations) 116
Appendix 3 (Complete Program LIsting) 131

TABLE OF CONTENTS

Abstract 1
Table of Contents 2

1.0 OVERVIEW OF THE REPORT 4

2.0 OVERVIEW OF DATA BASE SYSTEM CAPABILITIES OF FSTV . . . 7
2.1 Structure of User Session 9
2.2 Data Definition Session 12
2.3 Data Definition Query Session 15
2.4 Data Manipulation Session 16
2.4.1 Databsase Query 16
2.4.2 Creating Entities 18
2.4.3 Modifying Entities 19
2.4.4 Deleting Entities 20
2.5 Data Base Internal Schema 21

3.0 IMPLEMENTATION CHARACTERISTICS OF FSTV 25
3.1 Emulation of a Multi-level Multi-processor Architecture 25
3.1.1 Scope and Storage Class of Variables 26
3.1.2 Inter-level Communication 27
3.2 Functional Decomposition 32

4.0 FSTV INTER-LEVEL COMMUNICATION SERVICE ROUTINES 37
4.1 Insulating From SEND Protocal -- Use of SVCS1 and SVCS2 37
4.2 Insulating from Control Structure -- Additional service routines 41

5.0 OVERVIEW OF FSTV FUNCTIONAL PROGRAM MODULES 44
5.1 Memory Management Level 47
5.1.1 Data Model at the Memory Management Level 47
5.1.2 System Databases and Data Definition Modules at the Memory Management Level 48
5.1.3 Data Manipulation Modules at the MM Level 49
5.1.4 Initialization at the Memory Management Level . . . 50
5.1.5 Subroutines at the Memory Management Level 51
5.2 Internal Schema Level 52
5.2.1 Data Structures at the Internal Schema Level 52
5.2.2 System Databases and Data Definition at Internal Schema Level 53
5.2.3 Data Manipulation at the Internal Schema Level . . . 58
5.2.4 Initialization at Internal Schema Level 69
5.2.5 Subroutines at Internal Schema Level 70
5.3 Entity Level 71
5.3.1 Data Structures at Entity Level 71
5.3.2 System Databases and Data Definition Modules at Entity Level 72
5.3.3 Data Manipulation at Entity Level 76
5.3.4 Initialization at Entity Level 79
5.3.5 Subroutines at Entity Level 81
5.4 The User Interface Level 82
5.4.1 User Interface Modules 82
5.4.2 Other Subroutines at User Interface Level 85
5.5 The driver module - FSTV 88

1.0 OVERVIEW OF THE REPORT

This report describes in detail the current implementation of the software test vehicle (STV) for Functional Hierarchy of INFOPLEX data base computer. The basic concepts of INFOPLEX are found in <Madnick 78> and the reader is assumed to have some understanding of the proposed architecture of this data base computer.

The purpose of STV is to provide a better understanding of the architecture and funcitonalities of the INFOPLEX design by emulating its architecture and simulating its functionalities in software. It aims at validating the communication protocols among its various components, tightening functional algorithms for database management and data movements, providing behavioral and preliminary performance information concerning the architecture, and serving as a test bed before realizing the design in the hardware prototype.

The STV project is divided into the Functional Hierarchy STV (FSTV) and the Storage Hierarchy STV (SSTV). In addition, there is a Control Structure Program which provides an emulation of the multi-level multi-processing hardware architecture and

communication protocols. FSTV and SSTV are to be developed separately, each integrated with a separate copy of the Control Structure Program. However, the eventual goal is to produce an integrated version, in which FSTV and SSTV are coupled together through a single copy of the Control Structure Program.

Implementation of FSTV is based on a preliminary design of Functional Hierarchy presented in <Hsu80>. In that design, data base management functions are decomposed into hierarchical levels, each level to be implemented as a level of Functional Hierarchy. A 3-level version of FSTV was implemented by Bruce Blumberg <Blumberg81>. His implementation followed the philosophy of hierarchical decomposition and incorporated many distictive features. However, his version did not take the hardware architecture of Functional Hierarchy into account, and could not be integrated with the Control Structure. Based on his experience and the availability of a Control Structure implemented by Tak To <To81>, the current version of FSTV is implemented with special attention paid to a richer set of data base capabilities and architectural compatibility.

This report is divided into 6 chapters. The next chapter provides an overview of the functionalities of FSTV as a data base management system: what it does, what data model it implements, and how it interacts with the user. Chapter three provides a description of the implementation characteristics of FSTV. In particular, it contains conventions used to preserve

architectural compatibility and to interface with the Control Structure. Chapter four and five are primarily program documentation. A description of a set of service routines devised to ease the task of interfacing with the Control Structre is presented in Chapter four, while Chapter five contains an explanation of functional program modules in the current implementation. Chapter six concludes this report by pointing out directions for future refinement. A sample terminal session is included in Appendix 1. Appendix 2 contains a list of Control Blocks (used as inter-level interface) and external/static variables. Finally, a complete program listing is found in Appendix 3.

2.0 OVERVIEW OF DATA BASE SYSTEM CAPABILITIES OF FSTV

FSTV currently implements a subset of functionalities proposed in the preliminary design <Hsu80>. The proposed architecture with respect to data abstraction is similar to that of the ANSI/SPARC design, basically a 3-level architecture which contains an external schema level, a conceptual schema level and an internal schema level. In our design, however, we have added a base data model schema level to capture data definition of the integrated data base and to support multiple conceptual schemas defined on top of, and mapped to, the base data model schema. This way, a conceptual schema may use a data model which is different from that of the base schema, effecting support of a multiple data model data base management system. External schemas can then be defined on top of, and mapped to, a conceptual schema. This 4-level architecture is depicted in the following figure.

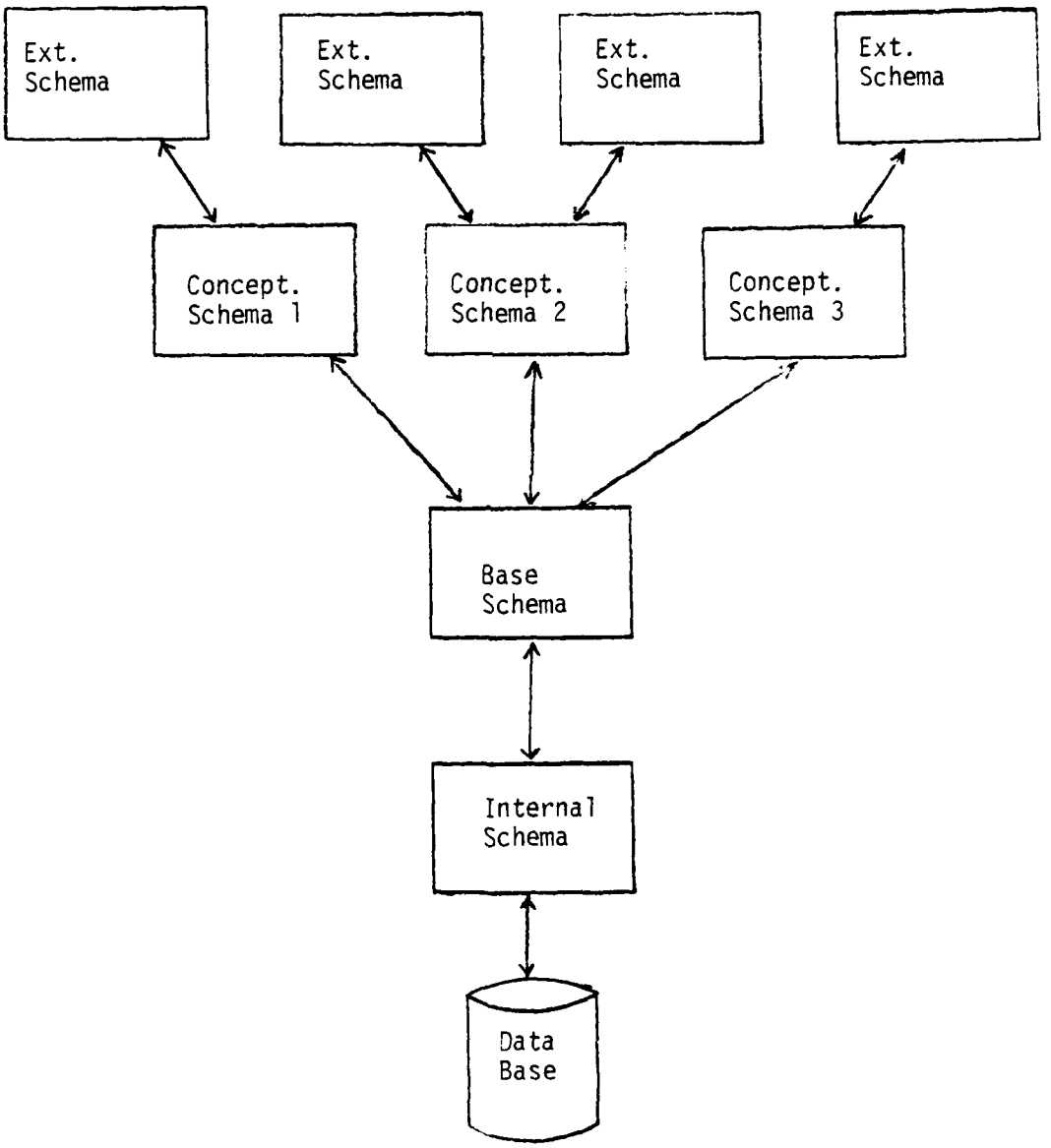


Figure: Architecture of INFOPLEX with respect to data abstraction

The upper two levels in the figure are aggregatedly called the veiw facility. In the current version, the view facility is not availale, and the base data model implemented at the base schema level is an entity network data model. Therefore, this level is called the Base Schema Level or the Entity Level in FSTV.

This overview chapter describes what the system can do and how a user may interact with the system. It also provides a brief description of the entity network data model and the internal schema used in FSTV. A sample terminal session is included in the Appendix.

2.1 STRUCTURE OF USER SESSION

FSTV interacts with a user through a user session in the User Interface Level. Since currently only the DBA session is available, the user will enter the DBA session, in which he may define or manipulate the database. If the former is chosen, a data definition session is initiated. If the user chooses data manipulation, then he may create, delete, modify and query the database. The User Interface Level guides the user into the desired seeion and then accepts input strings, parses them and formulates proper requests to go down to the next level. It also displays error messages or return data at the user's terminal

based on the return from the next level. The structure of the user interface is depicted in the following figure.

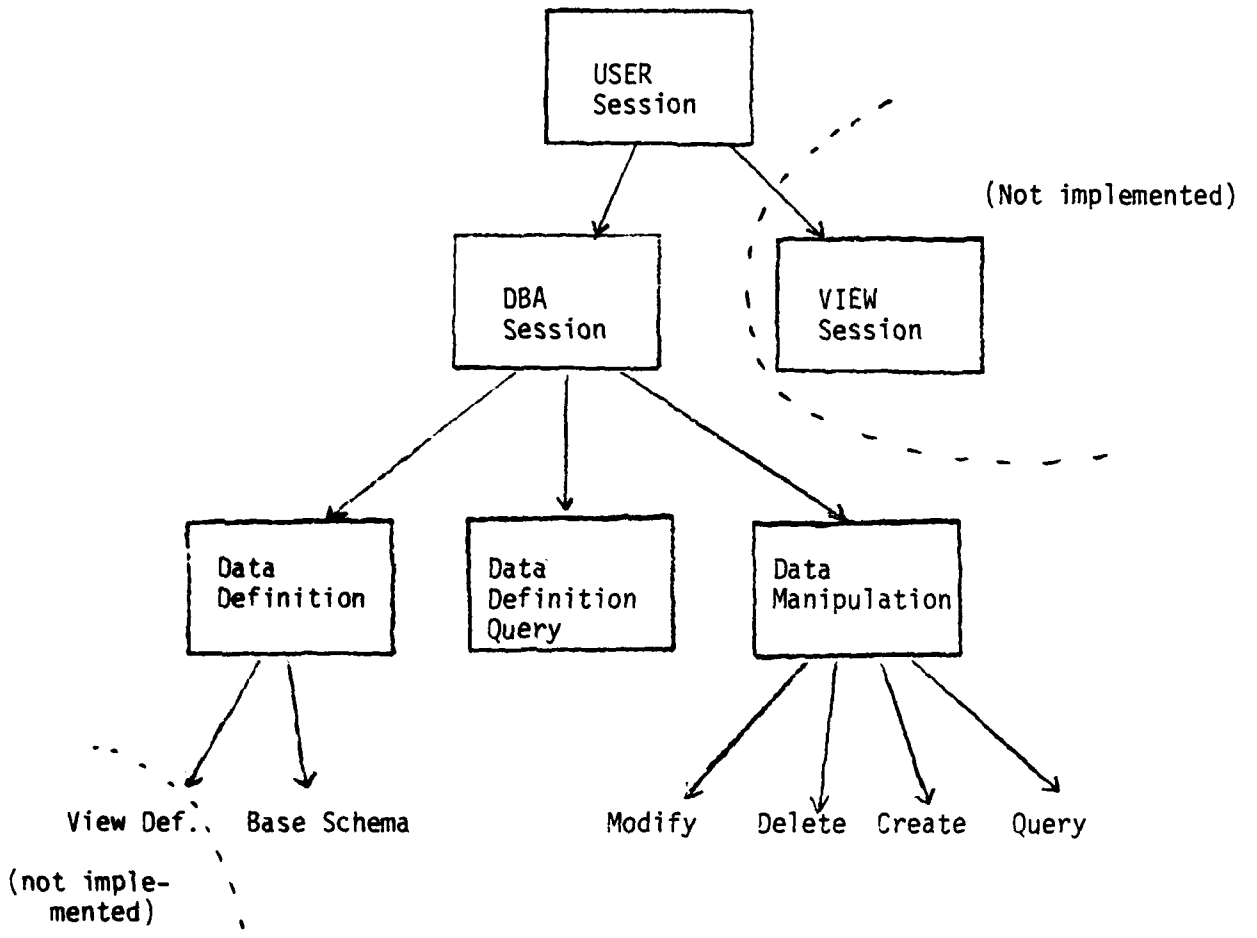


Figure: Structure of the User Sessions

2.2 DATA DEFINITION SESSION

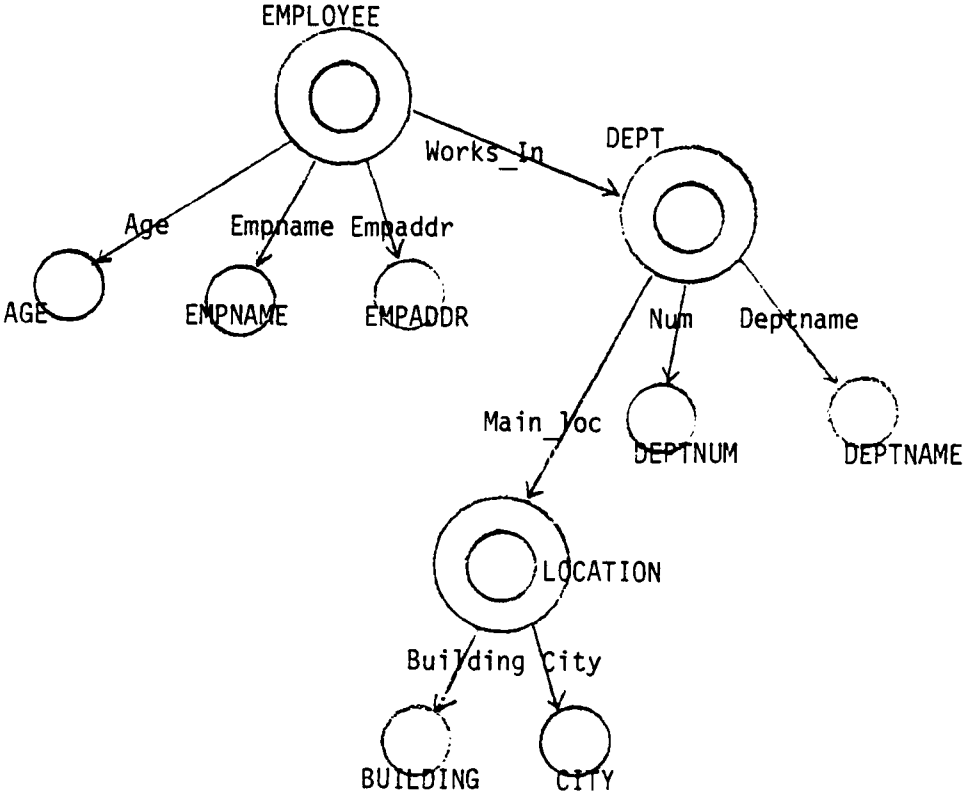
The DBA defines the data base in an entity network data model. This model views the world as a collection of entities and attributes of these entities. Generically similar entities are grouped into entity sets. Entities in one entity set will share the same set of attributes.

The DBA will model the application environment into entity sets. For each entity set, he may define an arbitrary number of attributes. The domain of an attribute can be either a value set or an entity set. It is through the latter that the entity sets defined can be made to explicitly relate to each other and form a network.

This entity network data model bears many properties that recent research in the area of data modelling has advocated. Its basic principles are outlined in <Hsu80> and a draft of a detailed description of this model, including its retrieval and update primitives, is forthcoming. However, the current implementation extracts only a subset of its intended set of semantics. In particular, concepts such as generalization and inter-entity-set aggregation are not provided. A virtual information facility is also lacking. Nevertheless, the basic structure of the data model is preserved such that additional

semantics may be added to without a complete rewrite of the current implementation.

Before we describe the data definition session in detail, we remind the reader of a few definitions used in the entity network data model. An attribute with a value set as its domain is called a value attribute; on the other hand, if its domain is an entity set, the attribute is called an entity attribute. Also, a direct attribute is an attribute which is directly related to the entity set. In contrast, a derived attribute is an attribute of an attribute. For example, in the following figure which describes an entity network data base schema, EMPNAME is a direct attribute of the entity set EMPLOYEE, while DEPTNUM is a derived attribute of EMPLOYEE because it is a direct attribute of the entity set DEPT, which in turn is the domain of the WORKS_IN attribute of EMPLOYEE.



: Entity Set

: Value Set

: Attribute

Figure: An Example of the Entity Network Schema

The Base Scehma Level, as currently implemented, will accept definitions of entity sets and their attributes. To define the base schema, the DBA enters the Base Schema Data Definitionn Session. An entity set is to be defined by giving an entity set name. An attribute definition, on the other hand, includes the name of the attribute, its domain type (i.e., value set or entity set), and its funciton type (i.e., many-to-one, one-to-one, or key). If the attribute is a value attribute, the definition also specifies the value type (i.e., numeric or character) and maximum length of the value (in number of characters), and if the value type is numeric, the allowable numeric range. If it is an entity attribute, only the name of the domain entity set need be given.

To facilitate ease of use, the user is provided with a panel to describe the data definition. Appropriate prompts are given in the panel.

Data definition can be incremental; i.e., new entity sets and additional attributes for existing entity sets may be defined after some data has been inserted into existing entity sets.

2.3 DATA DEFINITION QUERY SESSION

A user may choose to enter this session to discover what entity sets and attributes have been defined for the database. He may choose to look at the definition of all entity sets or that of a particular entity set. He may also choose to look at the definition of all attributes of an entity set or that of a particular attribute. The system will provide appropriate prompts as the user proceeds.

2.4 DATA MANIPULATION SESSION

In an entity network data base, entities may be created and deleted. Attribute values can be inserted, modified or deleted. A non-procedural query language can be used to retrieve information from the database. To do these, the user will get into the data manipulation session.

2.4.1 DATABSASE QUERY

To do a query, the user enters the Query Subsession. He will give the name of the entity set and a list of attribute names of this entity set he would like to see. If any of the attributes

has a predicate (e.g., >150, ='Stu Madnick'), the user will give it along with the attribute list. Multiple predicates are allowed on an 'AND' basis. (No 'OR' capability is provided for in the current version.)

The attribute list may also include derived attributes. The user expresses the derived attribute as a series of direct attributes that would lead to that derived attribute. For example, the city of the main location of the department an employee works in, using the example database shown in the above figure, is expressed by

```
WORKS_IN (MAIN_LOC (CITY))
```

as a derived attribute of the employee entity. The derivation path can be arbitrarily deep. An example query statement is given below:

```
EMPLOYEE
EMPNAME, EMPADDR, AGE > 50, WORKS_IN (DEPTNUM = 15)
```

This query wants the employee name, address, age and the department number of the department the employee works in, for those employees whose ages are greater than 50 and who work in department number 15.

2.4.2 CREATING ENTITIES

To create entities, the user enters the Create Subsession. The user will give the name of the entity set he is to create entities for, and a list of attributes that he will give values to. He then will provide the values of these attributes at the prompt for data. The attribute list may include derived attributes, in which case the value given for that attribute is used to identify an existing entity that the new entity is to associate with. For example, in creating an employee, one of the attributes may be WORKS_IN (DEPTNUM), and the corresponding attribute value given by the user is used to identify the department the user works in. However, the new entity cannot be made to relate to a non-existing entity. In our example, if the department number given does not identify an existing department entity, the employee cannot be created.

To create an entity, the user can select the set of attribtes of which values will be given. That is, an entity may be created while values of some attributes of the entity are not known.

The system will also intercept values of attributes that violate the attribute definition. This includes key violation, value type violation and numeric range violation.

2.4.3 MODIFYING ENTITIES

Values of attributes of existing entities (i.e., entities already created using the CREATE command introduced in the previous subsection) may be modified. To do so, the user must enter the Modification Subsession. Modification of attributes takes 3 forms: insert, modify and delete. However, the user must first identify the entity to be modified by giving value(s) of its attribute(s) that will uniquely identify the entity in the entity set. Then he will specify the attributes to be modified, prefixing them with a modification operator (i.e., insert, modify or delete). At the prompt for data, the user will enter new values of these attributes. The following is an example of a modify command:

```
EMPLOYEE
-ID:EMPNUM,-INSERT:WORKS_IN(DEPTNUM),-MODIFY:EMPADDR
105, 14, '550 Memorial Drive'
```

where EMPLOYEE in the first line specifies the entity set name, and the second line provides the information about what to be modified: the employee entity to be modified is identified by employee number 105, the attribute of this employee to be inserted is a derived attribte which asserts that this employee now works in department 14, and this employee's address is to be changed to '550 Memorial Drive'.

Notice that the interpretation of a derived attribute appearing in the modify command is the same as that in the create command; that is, it merely associates the entity being modified to an existing entity, but does not create any new entities or values. An error message will be given if the associated entity cannot be found in the existing data base. Also, violation of attribute definition will cause the modify command to be rejected.

2.4.4 DELETING ENTITIES

To delete an entity, the user enters into the Delete Subsession. All that the user has to specify is a set of attributes that will uniquely identify the entity to be deleted. All other attributes of this entity will automatically be removed. An example is given below:

```
EMPLOYEE
EMPNUM
105
```

This command deletes the employee entity whose number is 105.

2.5 DATA BASE INTERNAL SCHEMA

The data base internal schema describes how data in the data base is actually stored. It includes aspects such as access paths and record formats. This section provides a brief description of the internal schema used in the current version of FSTV, which is implemented at the Internal Schema Level and the Memory Management Level.

The internal schema used in FSTV is a simple network type of link list. Data elements are grouped into Primitive Sets (Psets). For each primitive set, an arbitray number of Binary Associations (Bsets) can be defined. When a data element is to be stored, the level of FSTV above the Internal Schema Level identifies the Pset this data element belongs to, and passes this to the Internal Schema level to be stored in that Pset. Likewise, the data elements a data element associates with through certain Bsets can also be given, and later can be retrieved together with that data element. In order to do so, the upper level will have to define Psets and Bsets for the Internal Schema Level, and remember the meaning of each Pset and Bset. The Internal Schema Level will provide IDs for Psets and Bsets defined. This concept of BEU is depicted in the following figure.

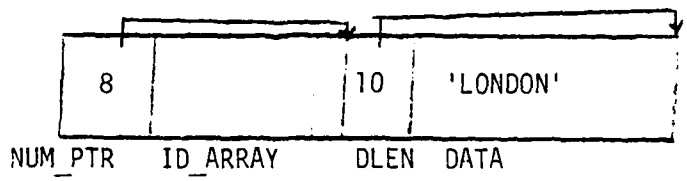


Figure: An example BEU

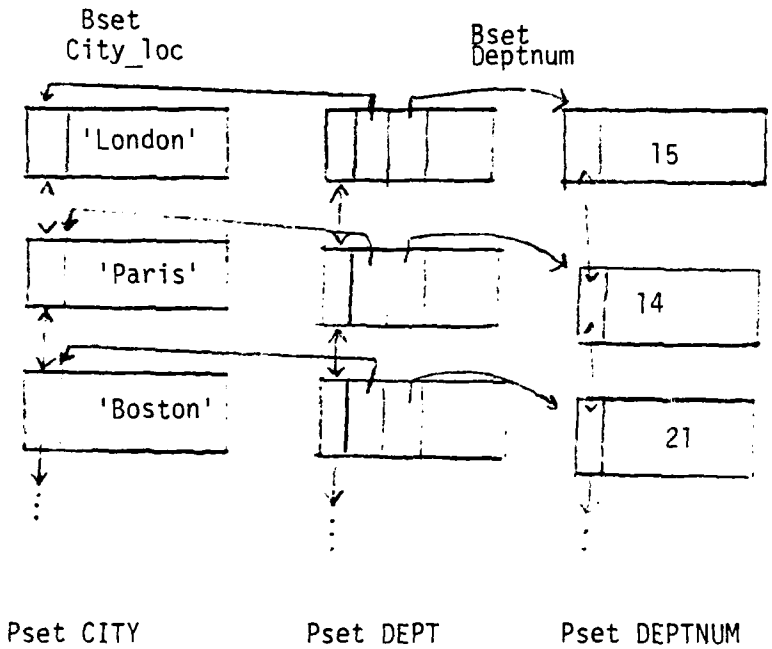


Figure: An example BEU network

The way the internal schema level organizes these data elements is through linkage. Each data element is implemented as a Basic Encoding Unit (BEU). A BEU contains the data and some control information. In the current version, a BEU contains an ID array, the size of which is indicated by the NUM_PTR field of a BEU, and a data field, the size of which is indicated by the DLEN field of a BEU. BEUs of one Pset are chained together through a bi-directional link list, making use of two slots in the ID array. Association with other data elements is implemented by storing the IDs of the latter in pre-allocated slots in the ID array. By 'pre-allocated' we mean that the position of the slot determines the meaning of the association, or the Bset id. Through this simple mechanism, data elements and their associations are remembered by the data base.

In the current version, there are no special access paths built for the purpose of improving performance of a search, such as indices and hash tables. There is also no intelligent access path selection algorithm to identify the most efficient way of locating a data element given its direct or derived associations. While these capabilities occupy the highest priority in future refinement, the basic functionalities of creating, deleting, modifying and retrieving data elements are all present.

The Memory Management Level responds to the Internal Schema Level's requests to store and retrieve BEU's. When a BEU is stored, the Memory Management Level provides an id which can be

used as a key for retrieval of the BEU. This level currently manages the virtual storage by linearly assigning BEU's to be stored in the next available address. No segmentation or clustering is provided for.

3.0 IMPLEMENTATION CHARACTERISTICS OF FSTV

In Chapter 2, discussion centered around the capabilities of the FSTV from the point of view of a DBA or a database user. In this chapter we shall describe implementation considerations which characterize FSTV as a Software Test Vehicle for the Functional Hierarchy of INFOPLEX. The content of this chapter can also be viewed as a set of strategies and programming conventions used in the STV project, and guidelines to be followed in future software modifications and additions. The following two subsections describe implementation characteristics relating to the emulation and functional decomposition aspects, respectively, of FSTV.

3.1 EMULATION OF A MULTI-LEVEL MULTI-PROCESSOR ARCHITECTURE

As mentioned earlier in Chapter 1, one of the purposes of the STV is to validate proposed algorithms in an emulated multi-level multi-processor architecture and to observe their behavior. The Control Structure Subsystem of the FSTV is devised to provide

this emulated environment. While the modules in the FSTV do not have to be aware of details of the (emulated) hardware environment, its program structure and communications between modules must conform to those demanded by the Control Structure. The Control Structure, on the other hand, will provide mechanisms for multi-processing and global bus and gateway controller emulation during execution of the FSTV modules.

Two types of programming conventions are observed during implementation of FSTV in order to facilitate working with the Control Structure. They are described in the following subsections.

3.1.1 SCOPE AND STORAGE CLASS OF VARIABLES

The context switching mechanism used in the Control Structure requires that the static and the controlled variables be used with caution. (Refer to chapter 7 of <To81>.) In essence, the FSTV program modules should make use of the based and automatic varialbes and refrain from using the static and controlled variables as much as possible. In the current implementation of the FSTV, static variables are used in association with the initialization modules only; i.e., once the FSTV is initialized, values of these variables will stay constant during run time.

The controlled variables are not used in any module. By observing this rule, the FSTV modules are made re-entrant, therby making it a relatively simple task to generate the multi-threaded version of the FSTV.

In addition, the architectural requirement that there be no shared data across levels necessarily limits the scope of the variables to being within the modules of the same level only. External variables known to modules across levels should not be used. A simple rule to follow therefore is to avoid using external variables. In the current implementaton, use of the external variables are associated with the initialization variables mentioned in the previous paragraph, and none are known across levels. Appendix 2 contains a list of all external and/or static variables used in the current version.

3.1.2 INTER-LEVEL COMMUNICATION

The hierarchical structure of INFOPLEX requires that inter-level procedure calls go through gateway controllers and the global bus. In simulating this architecture, the Control Structure Subsystem of the STV assumes that the functional modules observe the inter-level procedure call convention.

Inter-level communications in the FSTV are in the form of message passing between adjacent levels. Direct communication between unadjacent levels is not used.

Message passing vs. subroutine call. Each level of INFOPLEX contains a number of functional modules. In the STV, they are represented by program modules. Normally, when a module encounters a sub-task which is to be processed by another module, it uses a subroutine to activate the latter. However, since the FSTV is constructed for use in a multi-level multi-processor system, a number of pre-defined tasks are not invoked directly, but are invoked through message passing, a facility which is provided by the Control Structure. In fact, all cross-level invocations are done through message passing. This conforms to the requirement of the architecture as well as to the need of informing the Control Structure in the event of an inter-level call, so that the Control Structure can emulate the load on the gateway controller and the global bus.

The Control Structure provides a set of message passing protocols (including message passing among modules in the same level.) (Refer to chapter 5 and 7 of <To81>.) When a module at a higher level in the Functional Hierarchy needs to invoke a module at a lower level, it necessarily makes use of these protocols in place of a direct subroutine call. Likewise, when the module at the lower level completes the task, it makes use of these protocols to send the results back to the invoking module.

To this end, all modules in the FSTV are divided into two groups: the Top-Level Procedures (T-proc's) and the Subroutine Procedures (S-proc's). The T-proc's are always invoked by message passing, and the S-proc's by direct subroutine calls. Each level is consisted of a set of T-proc's which can be invoked by the upper level, and a set of S-proc's whose names are unknown to any other levels, and are not directly invocable by the upper level. It is noted that to invoke a T-proc at the same level a process will have to send a message to it rather than to use a traditional subroutine call. Therefore the convention for inter-level procedure call is more properly termed as the convention for inter-T-proc call.

Since T-proc's represent entry points to a level, special attention is given to its invocation. The arguments needed to invoke a T-proc are grouped into a set of Control Blocks. A control block is a based variable which cntains a pre-defined data structure. The set of control blocks associated with a T-proc includes both the calling control blocks and the return control blocks. When the T-proc is invoked, it is passed a link list of calling control blocks; when it returns, it puts the return messages and values in a link list of return control blocks. The names of the T-proc's at a level and the control blocks associated with them collectively form the interface of the level seen by the upper level. This mechanism satisfies the requirement that "..each level of the functional hierarchy interacts with other levels through a clean set of interfaces"

<Hsu80>. Messages passed between levels are, therefore, link lists of control blocks. Details of control block formatting conventions are described in the following.

Control Block Format. According to the hardware architecture of INFOPLEX which requires that no shared memory is used between levels, when a message is to be passed to a process at another level the message is to be physically moved from the local memory of the calling level to that of the called level. In simulating this activity, the Control Structure has to be given a pointer to the message and the length of the message. While it is possible to give the Control Structure a message in a contiguous block, it is more desirable from the programmers' point of view to be able to give the Control Structure a link list and the necessary information for the Control Structure to collect all data in the link list. To this purpose, a convention for interacting with the Control Structure is devised: every element (a Control Block) of the link list contains a CNTL_INFO as the first field of the data structure as follows:

```
2 CNTL_INFO,
  3 LEN FIXED BIN (15),
  3 CBTP FIXED BIN (15),
  3 PTR PTR,
2 DATA,
   .....
```

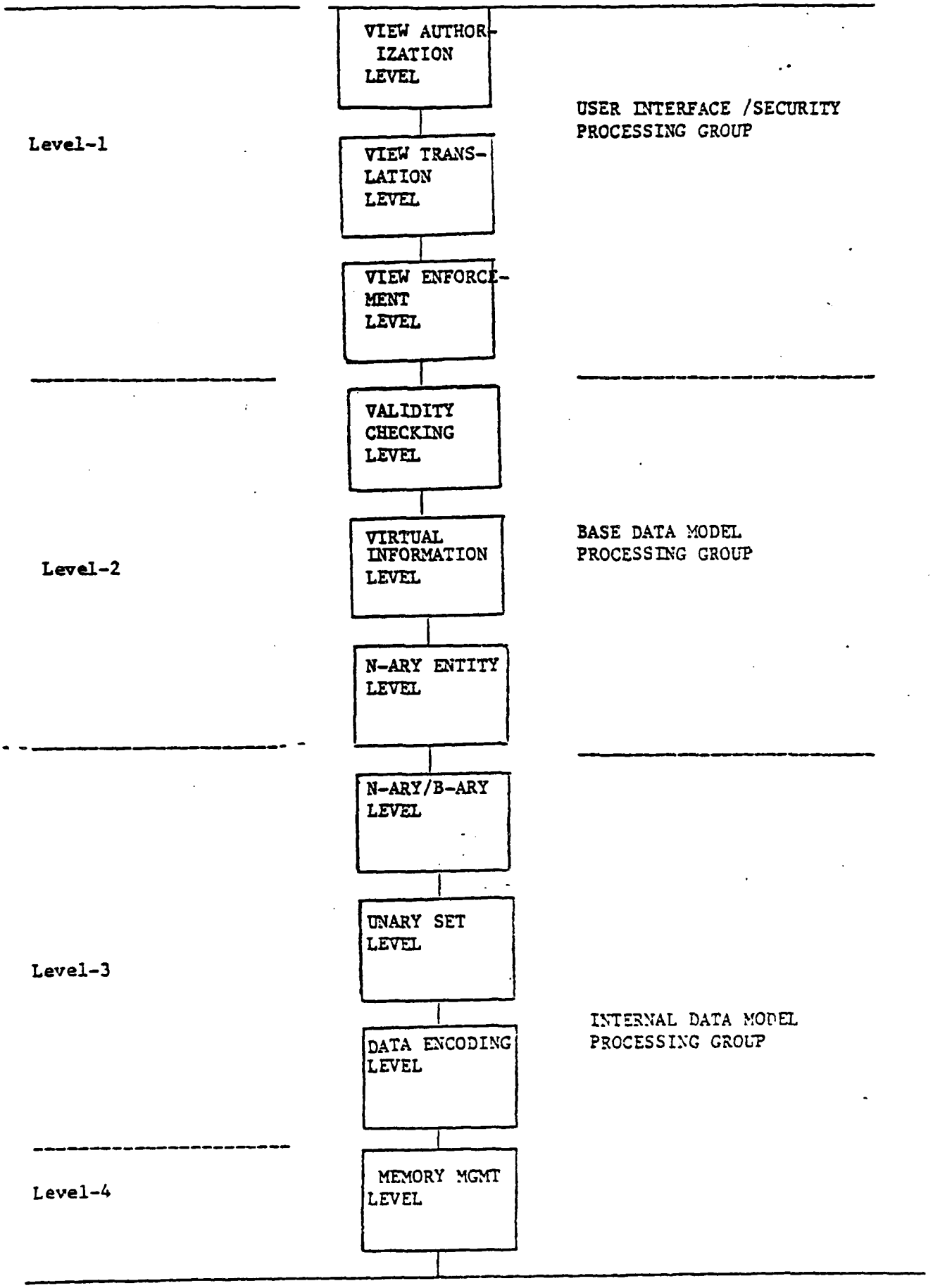
where LEN describes the length of the data portion of the control block, CBTP is a unique number assigned to this type of control block, and PTR points to the next control block in the link list. With the information in CNTL_INFO the Control Structure will be able to collect all elements and move them to the other level. Similary, the called process simply follows the pointers to obtain all arguments it needs. The data structure of the DATA portion of a control block list represents the agreed upon interface between sending and receiving processes. This convention aplies to both argument passing and return data. Examples of Control Blocks are given in Appendix 2.

Service routines and programming conventions. In order to insulate the FSTV modules from furture changes to the Control Structure-provided communication protocols, a set of service routines are used to centralize the use of these protocols. These are SVCS1 and SVCS2. In adiition, since the current version of FSTV does not make use of intra-request parallelism, another set of service routines are devised on top of SVCS1 and SVCS2 in such a way that, if these routines are used to invoke protocols provided by the Control Structure, the existence of the Control Structure could be completely shielded from the FSTV. Therefore, for example, it is possible to run FSTV alone without the Control Structure, so long as the second set of service routines is replaced by a dummy set. This feature enables independent debugging and testing of the FSTV from the Control Structure, thereby reducing the complexity of the software

development process. The next chapter contains a description of these service routines, their interfaces and how to use them.

3.2 FUNCTIONAL DECOMPOSITION

The database management functions in the current version of the FSTV have been hierarchically decomposed into four levels. Each level communicates with only the adjacent level. The following figure illustrates the relationship between the current implementation and the target implementation as outlined in <Hsu80>. While the current implementation is a simplified version of the target system, it illustrates the process of reducing a reasonably complicated user query to accesses/updates in a linear address space.


VIEW AUTHOR-
IZATION
LEVEL
VIEW TRANS-
LATION
LEVEL
VIEW ENFORCE-
MENT
LEVEL
VALIDITY
CHECKING
LEVEL
VIRTUAL
INFORMATION
LEVEL
N-ARY ENTITY
LEVEL
N-ARY/B-ARY
LEVEL
UNARY SET
LEVEL
DATA ENCODING
LEVEL
MEMORY MGMT
LEVEL
STORAGE HIERARCHY
Level-1
Level-2
Level-3
Level-4
USER INTERFACE /SECURITY
PROCESSING GROUP
BASE DATA MODEL
PROCESSING GROUP
INTERNAL DATA MODEL
PROCESSING GROUP

One important 'feature' of the design is to be stressed. To eliminate duplication of data storage and its management functions, system databases (e.g., catalogues) used at a level are stored and manipulated as regular data at the next level. That is, all the catalogues used at a level are defined as data sets at the next lower level during the initialization process. This means that the local memory at a level is strictly used for storing the context of active processes, and that all the structuring information such as catalogues and indices as well as the database itself is stored in the virtual memory (therefore in the Storage Hierarchy) managed by the Memory Management Level.

The four levels in the current implementation are briefly described as follows. Details of the functional modules are the subject of chapter 5.

1. User Interface Level (Level 1): This level interacts with the user directly. It accepts users' commands to define and manipulate the databases. It parses input character strings, identifies the meaning of the input tokens, and generate proper requests to the next level for validation and execution of the input commands. For ease of use, the data definition portion of the user interface makes use of the panel facility, which enables the DBA to define the database by filling up blanks in a pre-defined panel. This level does not create or make use of any system databases.

2. Database Semantics Level (Level 2, or The Entity Level): This level manages the entity/attribute catalogues, validates database insertions and updates, and transforms the high level database definition and manipulation requests into commands to operate on the data structures of the internal schema. The system database at this level, the entity/attribute catalogues, is itself defined in terms of entities and attributes, and is stored as regurlar database data in the next level.

3. Internal Schema Level (Level 3, or The N-ary Level): This level manages the Pset/Bset catalogues and performs retrieval and update of the content of the pre-defined Psets and Bsets. It accepts a tree-structured database command and decides upon the P-elements to be retrieved or affected and executes the command. It ultimately implements information in the database in the form of BEU's (Basic Encoding Units); a BEU is given to the next lower level to be stored as a unit of bit string. The internal schema implemented in the current version is very simple: it implements P-elements within the same Pset as an unsorted bi-directional link list, and binary associations among the P-elements through uni-directional linkage pointers.

4. Memory Management Level (Level 4): This level manages a linear virtual address space which is initialized by the FSTV Console Program as a based area of a certain size.

Store, retrieval and update of BEU's are carried out at this level as operations on this based area.

4.0 FSTV INTER-LEVEL COMMUNICATION SERVICE ROUTINES

This chapter documents the set of service routines used by FSTV functional programs to interface with, and to insulate from, the Control Structure-provided message passing protocols. The first section describes the two service routines, SVCS1 and SVCS2, which are used by FSTV functional modules to insulate themselves from directly using the SEND protocol provided by the Controld Structure. The second section describes another set of service routines, TCALL, TBEG and TRTN, which are used to insulate the FSTV completely from the Control Structure.

4.1 INSULATING FROM SEND PROTOCAL -- USE OF SVCS1 AND SVCS2

Convention for the calling procedure. When a process wishes to invoke a T-proc, it must send a message to the operating system of the target level. The message will contain the name of the procedure to be invoked and a pointer to a message to the invoked procedure. The latter message contains the return address (e.g., the mail box number) and calling arguments if any.

The calling process will prepare the calling arguments as a link list of control blocks. (The meaning and the format of a control block has been described in the previous chapter.) This link list is called a calling control block link list. The calling control block link list will then be prefixed with a special control block, the MSG control block. The purpose of this MSG control block is to inform the invoked procedure of the process id (i.e., the process address) of the calling process, so that return messages shall be routed correctly. The format of this MSG control block is exemplified by the following declaration:

```
DCL 1 MSG BASED (P),
      2 CNTL_INFO,
        3 LEN FIXED BIN (15) INIT(6),
            /* LENGTH OF DATA PORTION */
        3 DUM_CBTP FIXED BIN, /* NOT USED */
        3 PTR PTR,
          /* POINT TO FIRST CONTRO BLOCK IN CALLING
             ARGUMENT LINK LIST */
      2 DATA,
        3 PROC_ADDR,
         /* ADDRESS OF THE PROCESS INITIATING
            THIS MESSAGE */
         (4 LEVEL,
          4 VPID,
          4 BOXID) FIXED BIN (15);
```

The entire message link list is sent by calling SVCS1 service routine. SVCS1 takes 2 arguments: the name of the procedure to be invoked (CHAR(8)) and a pointer to the message link list (PTR). It does not care about the content of the message link list. SVCS1 returns a return code (FIXED BIN(15)):

```
CALL SVCS1 (PROC_NAME,PTR,RTN_CODE);
```

After the call, the calling process may continue until the point at which a return message from the call is needed. It then calls the WAIT protocol provided by the Control Structure, specifying the mailbox it waits on:

```
CALL WAIT (BOXID);
```

after which it may obtain the return message by moving the relevant data pointed to by VP.WAIT.MSG (set by the Control Structure) to the appropriate variables.

A calling process may send multiple messages at the same time, thereby achieving the benefit of intra-request parallelism.

Convention for the called procedure. All T-proc processes will have to respond to the message passing protocols set up by the Control Structure. When it is created, it must first locate the message link list passed to it from the caller in the base variable pointed to by VP.WAIT.MSG. Before it finishes, it must

have sent the return message, if any, to the calling process. The format of the message passed to it is as described in the previous subsection. To send a return message to an existing process, however, SVCS2 is used. SVCS2 also takes 2 arguments: the address of the receiving process (LEVEL,VPID,BOXID) and a pointer to the return message link list: (The return message link list is constructed in a way similar to that of the calling message; i.e., a return control block link list prefixed with a MSG control block.)

```
CALL SVCS2 (PROC_ADDR,P,RTN_CODE);
```

Note that it is possible for 2 processes to engage in a conversation, in the sense that the calling process may send a second message to the called process after the calling process has already obtained some return message from it. That is to say, a called process needs not "finish" itself immediately after sending the return message.

Summary. The functional program modules accomplish inter-T-procedure calls through SVCS1 and SVCS2 service routines. SVCS1 is used to invoke a new process, while SVCS2 is used to send (return) messages to a known process. Both SVCS1 and SVCS2 take two arguments: the name of the procedure to be invoked or the address of the existing process the message should go to, and a pointer to the message link list.

Text inclusion. A module which wishes to use SVCS1 and SVCS2 should include the macro SVCS. Declarations of MSG, PROC_ADDR, PROC_NAME are provided for in the macro.

4.2 INSULATING FROM CONTROL STRUCTURE -- ADDITIONAL SERVICE ROUTINES

To avoid direct use of WAIT and FINISH protocols and VP.WAIT.MSG provided by the Control Structure as well as SVCS1 and SVCS2, a functional module may use a set of 3 service routines to accomplish inter-T-proc communication: TBEG, TRTN and TCALL.

Convention for the called T-proc. When a T-proc process is first invoked, it calls TBEG to obtain the return address (i.e., the process address of the calling process.) The call takes the following form:

```
CALL TBEG (PROC_ADDR,P);
```

where PROC_ADDR will be loaded with the identifier of the calling process, and P points to the first control block in the argument link list.

Note that the MSG control block described in the previous section is stripped off by TBEG and no longer concerns the functional module. When a T-proc returns, it calls TRTN to pass the return control block link list to the calling process. The call takes the following form:

```
CALL TRTN (PROC_ADDR,P);
```

where PROC_ADDR is as described above and P points to the first control block in the return control block link list. Note also that the user of TRTN no longer needs to prefix its return link list with the MSG control block, since the latter will be added to by the TRTN routine.

Convention for the calling T-proc. When a module wishes to invoke T-proc, it formats the arguments in the form of a control block link list, and then issues the following call:

```
CALL TCALL (PROC_NAME,BOXID,ARG_PTR,RTN_PTR);
```

where PROC_NAME is the name of the T-proc to be invoked, BOXID is the mailbox number where the caller wishes the return messages to be stored, ARG_PTR points to the first control block in the argument control link list, and RTN_PTR (a return value) will point to the first control block of the return control block link list.

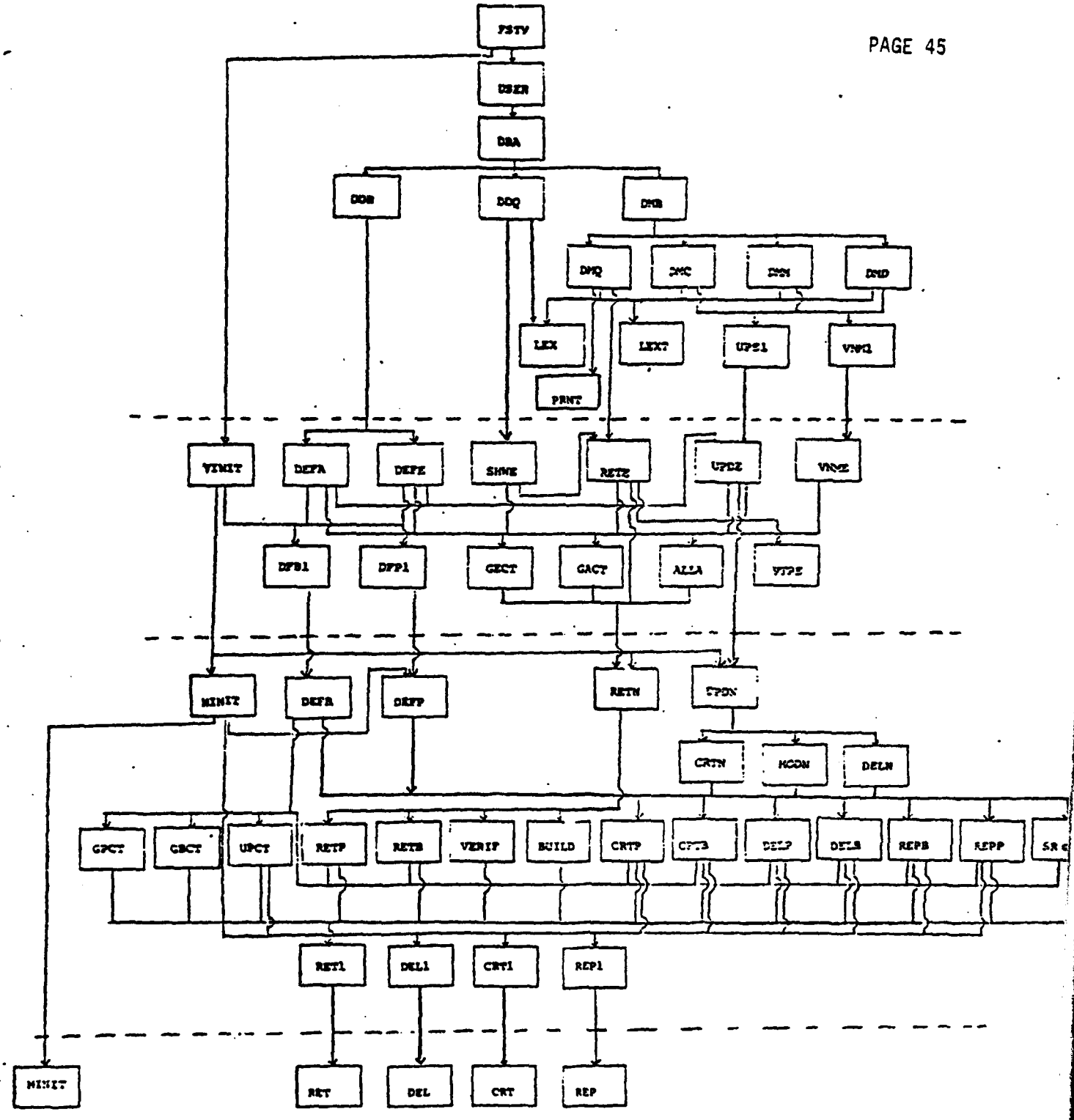
NOTES:

1. This set of service routines can not be used to engage two communicating processes in a "conversation mode" (refer to section 2), nor can it be used when there is intra-request parallelism involved.

2. To use this set of routines, the SERVICE macro must be included.

5.0 OVERVIEW OF FSTV FUNCTIONAL PROGRAM MODULES

This chapter contains a description of the logical function and the data structure of program modules in the current version of FSTV. The description follows the four hierarchical levels employed in FSTV, starting from the bottom level. For each level, we will discuss the data model (i.e. the data structure of objects processed at the level) and the system databases used, and provide an overview of the data definition, data manipulation, and level initialization modules that make up the level. The reader will notice that modules at a level are classified into T-proc's and S-proc's as discussed in the previous chapters, and that conventions and constraints mentioned previously are observed. In addition, the reader will see how information abstraction proceeds from the bottom level which recognizes only a linear virtual address space used as the database store to the database semantics level which processes high level constructs such as entities and attributes. A complete listing of these program modules is included in Appendix 3. The following figure contains a diagram showing all modules in the current implementation.


PSTV
DBER
DBA
DDB
DDQ
DMB
DMQ
DMC
DMN
DMD
LEX
LEXT
UPS1
VNM1
PRNT
VINIT
DEFA
DEFE
SHWE
RETE
UPDE
VNME
DFB1
DFP1
GECT
GACT
ALLA
VIPE
MINIT
DEFA
DEFP
RETN
UPDN
CRTN
MODN
DELN
GPCT
GBCT
UPCT
RETP
RETB
VERIF
BUILD
CRTP
DELP
DELB
REPB
REPP
RET1
DEL1
CRT1
REP1
MINIT
RET
DEL
CRT
REP

System Initialization.

Functional Hierarchy initialization consists of activities which render the functional modules in a runnable condition. Presently an assumption is made of the initial condition of the system when it is 'powered on': each level will already have all the functional modules loaded into its local memory. This may not be the case in reality. A bootstrap procedure which loads the minimum set of the functional modules (the core set) from the Storage Hierarchy may have to be added to each level. However, for the purpose of this first version, we may assume that this type of bootstrap is taken care of by the local operating system, which is simulated as part of the Control Structure and is out of the scope of FSTV.

Initialization of catalogues is also necessary before the system can respond to normal processing. For a system without an existing database (i.e., an INFOPLEX just bought off the shelf), the initialization module at a level will define the catalogue structures used at the level in terms of data sets implemented at the next lower level and establish a set of 'key' values which will be used at this level when normal processing starts. This means that initialization ought to start from the bottom level. To initialize a system which already has a database, the catalogue definition activities are omitted, and the initialization module simply restores the key values. The

individual initialization modules will be discussed further in the following subsections.

5.1 MEMORY MANAGEMENT LEVEL

The function of this level is to manage the virtual memory address space that Functional Hierarchy is provided for by Storage Hierarchy and to interface with Storage Hierarchy.

5.1.1 DATA MODEL AT THE MEMORY MANAGEMENT LEVEL

The basic element stored in SH is the Storage Encoding Unit (SEU). Each stored SEU is assigned a SEU id, which is equivalent to the virtual address of this stored SEU in the Storage Hierarchy. An SEU has the following format:

```
DCL 1 SEU BASED (SEUPTR)
  /* 2 MOVE_PTR BIT(32), NOT USED CURRENTLY */
     2 N FIXED BIN,
     2 INVAL CHAR(1) INIT('N'),
     2 DATA CHAR(I REFER (SEU.N));
```

where MOVE_PTR is an id in the SH which is always null unless the SEU has been moved, in which case MOVE_PTR contains the new location in SH, INVAL is a flag which, when set, signifies that this SEU has been deleted, and finally DATA is a character string with a variable length indicated by N and contains the data of this SEU.

The format of SEU is known to modules within this level only. The level above will pass to this level a bit string of data to be stored as a unit in return for an id, and retrieve this bit string back by presenting to this level the id. The Memory Management Level will allocate a SEU for each unit of data to be stored, and put the data in the DATA field of the SEU. It then finds a free area in the virtual memory and passes the SEU to the Storage Hierarchy.

5.1.2 SYSTEM DATABASES AND DATA DEFINITION MODULES AT THE MEMORY MANAGEMENT LEVEL

Presently a very simple memory management function is provided. Storage in the virtual memory is assigned incrementally starting from address 16. Therefore the only "system database" needed is a counter indicating the next

available virtual address. This counter is stored in address 0 of the Storage Hierarchy.

No data dictionary is used in the current version. There is also no data definition necessary.

5.1.3 DATA MANIPULATION MODULES AT THE MM LEVEL

There are four T-proc modules at this level to perform creation, deletion, modification and retrieval of data elements. Each of these functions is performed by one module.

CRT. This module responds to requests from the upper level to store data elements. Its calling control block contains the data and the length, while the return control block contains an ID, which is the bit representation of the virtual address assigned to the SEU encompassing the data. This module calls subroutine GMEM to get the virtual address, and subroutine PSEU to decompose the SEU into 8-byte packets and to store them into the Storage Hierarchy.

RET. This module processes requests to retrieve a data elements given its id. It calls subroutine GSEU to retrieve the

SEU of the data element from SH using the id and extracts the DATA portion of the SEU to return to the upper level.

MOD. This module processes requests to modify content of a data element given its id and new data. Like RET, it calls GSEU to retrieve the SEU of the data element from SH and replaces the DATA field of the SEU with the new data, and calls PSEU to put it back into SH.

DEL. This module processes requests to delete a data element given its id. It calls GSEU to retrieve the SEU, sets its INVAL field, and calls PSEU to put it back. No garbage collection is performed in the present implementation.

5.1.4 INITIALIZATION AT THE MEMORY MANAGEMENT LEVEL

MINIT. This module performs all the necessary steps to initialize this level during IPL. The calling control block contains the method of initialization ('file' or 'new'). If it initializes from a Storage Hierarchy that doesn't contain any data yet (i.e., a 'new' initialization), it bootstrap a value '16' into the first word of the Storage Hierarchy, which is to be used as a register for the next availabe address. If it initializes from an existing database, it retrieves a one-word

'key' for the upper level from a fixed location in SH and includes it in the return control block. (The upper level will use this key to 'recover' its system databases.) This module also provides an interface (i.e., when the OP field of the calling control block contains 'S') for the upper level to store a key.

5.1.5 SUBROUTINES AT THE MEMORY MANAGEMENT LEVEL

There are 4 subroutines at this level. GMEM (Get MEMory) returns a virtual address of a free area in SH given the length of the area requested. PSEU (Put SEU) decomposes a given SEU into 8-byte packets and stores them consecutively into a given area in SH. GSEU (Get SEU) performs the reverse of the task of PSEU: given an address of an area, reconstruct the SEU contained in that area by retrieving a packet at a time from SH. And, finally, PAKT (PAcKeTing) is a subroutine called by the above 3 subroutines to get a packet of data from SH given its byte address. PAKT is the only routione in the FSTV which is aware of the interface to the top level of SH. The interface is declared as a data structure ARG.

5.2 INTERNAL SCHEMA LEVEL

This level provides various 'stored structures' to store the database. It accepts definitions of data element sets and their relationships, and performs search, retrieval and update of these data.

5.2.1 DATA STRUCTURES AT THE INTERNAL SCHEMA LEVEL

This level envisions that the world is composed of a network of some basic data elements called Primitive elements (P-elements.) P-elements and their relationships are implemented at this level in the form of a Basic Encoding Unit (BEU). The reader is referred to section 2.5 for a description of the BEU concept employed in this implementation. A BEU is declared as follows:

```
/*DCL OF BASIC ENCODING UNIT (BEU) USED IN THE N-ARY LEVEL***/
DCL 1 BEU BASED(P),
      2 NUMPTR FIXED BIN,
      2 ID_ARRAY(I REFER (BEU.NUMPTR))BIT(32) INIT ((I)UNSPEC(NULL)),
      2 DLEN FIXED BIN,
      2 DATA BIT(J REFER (BEU.DLEN));
```

where ID_ARRAY is an array of ID's of related BEUs'. NUMPTR indicates the number of slots its ID_ARRAY contains, DATA is the bit string of the data element, and DLEN indicates the length of the data in terms of number of bits.

The format of the BEU is known only to this level. The upper level is not aware of how PSETs and their inter-relationships are stored.

5.2.2 SYSTEM DATABASES AND DATA DEFINITION AT INTERNAL SCHEMA LEVEL

Definition of Psets and Bsets are kept in two catalogues: the Primitive Set Cataluge (PCAT) and the Binary Association Set Catalogue (BCAT). Each entry in PCAT describes a primitive set. Its format is described by the following declaration:

```
/*DCLS OF PSET CATALOGUE AT THE N-ARY LEVEL******/
DCL 1 PINFO,
      3 NUMPTR FIXED BIN,
      3 PLEN /*IN NUMBER OF BYTES*/ FIXED BIN,
      3  PTYPE CHAR (1), /*'N' OR'X' OR 'B' OR 'C'*/
      3 LTYPE BIT (8), /*'00000001' IS LINK LIST*/
```

```
        3 L_ID BIT (32), /*ID OF THE FIRST BEU IN THIS SET**/
        3 L_POS FIXED BIN , /*POS IN ID_ARRY USED FOR LINEAR CHAINING
                    OF BEU'S WITHIN THIS SET*/
        3 L_POS2 FIXED BIN,
        3 MAP BIT (32) ; /*UP TO 16 PTRS ARE ALLOWED FOR A BEU*/
```

where NUMPTR describes the size of the ID_ARRAY of the BEU's in this Pset; PLEN is the legnth of the data element; PTYPE describes the data type of this primitive set, which can be numeric, bit or character string; PTYPE 'X' is a special type which is used when the data value of the Pset in question is insignificant (in this case the Pset is mainly used as an anchor for binary associations); L_POS and L_POS2 are the ID_ARRAY slots used for forward and backward chaining of all BEU's in the Pset; MAP is used to record assignment of ID_ARRAY slots: if the n-th slot is assigned, the n-th bit in MAP will be set; LTYPE describes how BEU's in this Pset are grouped together: currently only the bi-directional link list is in use. Among these parameters, NUMPTR, PLEN and PTYPE are provided by the upper level (refer to the description of DEFP below), while the rest are internally generated.

Each entry in BCAT describes a binary association set. Its format is described by the declaration of BCAT:

```
/*DCL OF BCAT TEMPLATE*/
DCL 1 BINFO,
```

```
2 PSETID(2) BIT (32),
2 POS FIXED BIN, /*POS OF POINTER ARRAY OF PSETID1 USED */
2 FUNC CHAR (1); /*'S' OR 'M'*/;
```

where PSETID(1) and PSETID(2) contain the ID's of the source Pset and the targe Pset involved in this binary association (note that all binary associations are directional); POS conatins the ID_ARRAY slot number used by BEU's in the source Pset to record the ID's of the associated BEU's in the target Pset; FUNC indicates the type of relationship between the source and the target: one-to-one ('S') or many-to-one ('M').

Both PCAT and BCAT are themselves implemented as Psets. Therefore each catalogue entry is implemented as a BEU. These two Psets are defined during system initialization (refer to section 5.2.4), creating the very first two PCAT entries of the system. This is exemplified by the following figure.

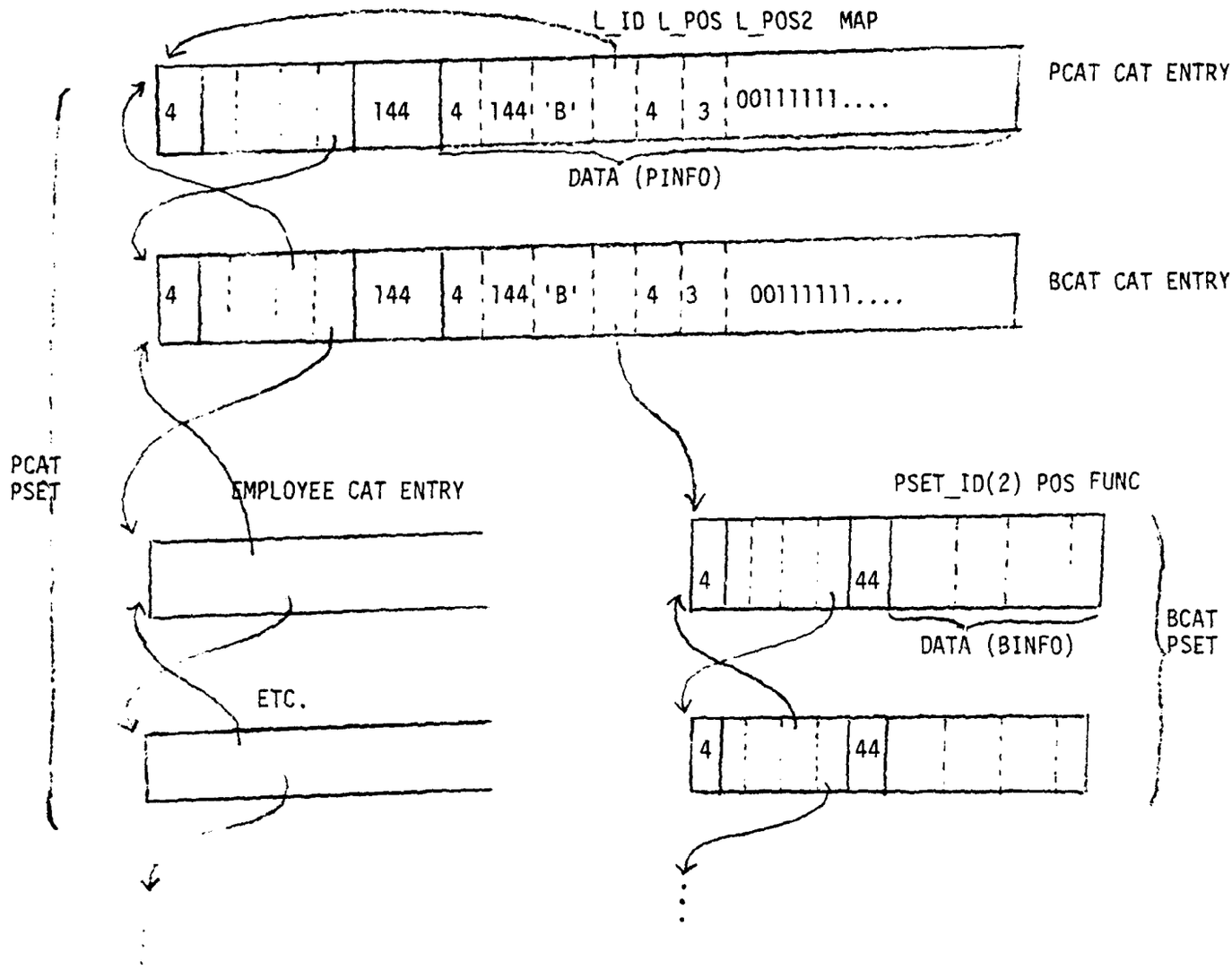


Figure: Structure of the Catalogue Implementation at the Internal Schema Level (N-ary Level)

To remember the ID's of the two BEU's that implement these two catalogue entries, two static variables are given values to during initialization: PCATID and BCATID. Their values will remain constant for the rest of the system processing. Procedures for defining Psets and Bsets will reference these two variables in order to insert new entries into the catalogue entry chain.

Two T-proc's form the data definition interface of this level. They are described below:

DEFP. This module (DEFine Pset) responds requests for primitive set definition. The calling control block contains the length and the type of the data field, and the size of the ID_ARRAY. (It could be argued that the last item shouldn't be passed since it relates to the internal, stored format of primitive elements which is theoretically unknown to its definer. Presently, however, it is included in the interface merely because a separate interface for the DBA to express his opinion on this matter is lacking. This issue will be further discussed.) The return control block contains the ID of this Pset, which is (but the upper level shouldn't care) the ID that the Memory Management Level assigned to the catalogue entry BEU of this Pset.

DEFB. This module (DEFine Bset) responds to requests for binary association set defintion. The calling control block

contains the Pset ID's of the source and the target Psets involved and its relationship type (i.e., one-to-one, many-to-one). The return control block contains the ID assigned to this Bset, which is equivalent to the ID that the Memory Management Level assigned to the catalogue entry BEU of this Bset.

5.2.3 DATA MANIPULATION AT THE INTERNAL SCHEMA LEVEL

This section describes the two T-proc's that form the data manipulation interface of this level: RETN and UPDN.

RETN. This module (RETrieval at iNternal schema level) accepts a reasonably complicated query expressed in a tree form in which nodes are Psets and arcs are Bsets. An arbitrary number of nodes can be included in the retrieval tree. As an example, a query of the form

RETRIEVE EMP (NAME, DEPT(DNAME), PROJ(NAME)) WHERE (AGE>50 AND DEPT(LOC) = 'New York')

can be expressed in a RETN tree shown in the following figure.

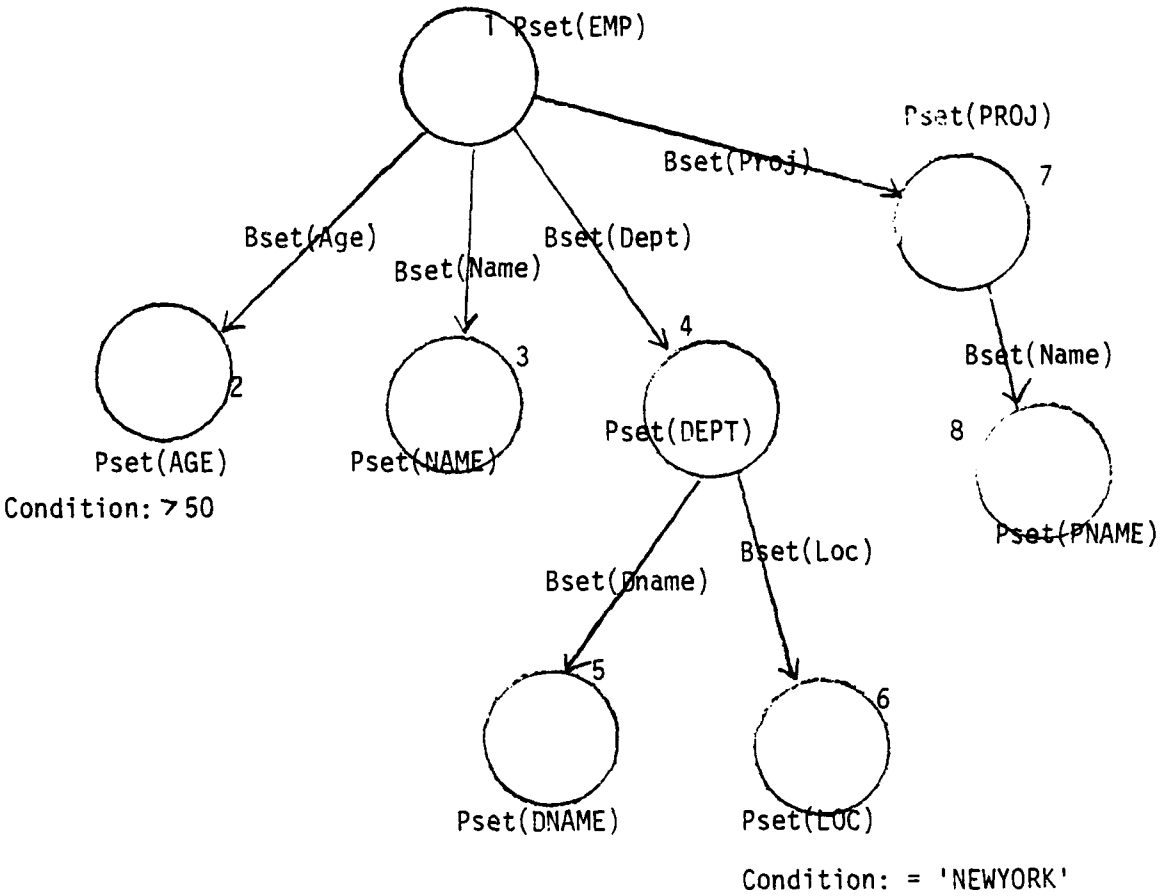


Figure: The RETN Tree of an Example Query

The caller of RETN passes a link list of RETN calling control blocks, each of the control blocks describing a node in the tree. The declaration of this control block is:

```
DCL 1 RETN_ARG   BASED(P),
      2 LEN     FIXED BIN (15) INIT (61),
      2 CBTP FIXED BIN (15) INIT( 25 ),
      2 PTR PTR INIT (NULL),
      2 NODE FIXED BIN (15) INIT(1), /*NODE NUMBER*/
      2 PSETID BIT(32), /*PSETID OF THE NODE*/
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/
      2 BSETID BIT(32), /*BSETID OF THE PARENT PSET*/
      2 GET BIT(8) INIT ('00000000'B), /*SEE RETNGET FOR DETAIL*/
      2 N FIXED BIN INIT(0), /*NUMBER OF RELEVANT 'OR' PREDICATES*/
      2 PRED (1),/*IN CURRENT VERSION ONLY ONE PREDICATE EACH TIME
            IS ALLOWED; FUTURE VERSION MAY USE REFER OPTION TO N*/
       3 CN FIXED BIN (15), /*CONDITION NUMBER*/
       3 SETOP BIT(8), /*RELEVANT IF BSET IF MULTI-TARGET;
                        NOT USED IN CURRENT VERSION*/
       3 OP BIT(8), /*COMPARISON OPS, SEE RETNOP FOR DETAIL*/
       3 DLEN FIXED BIN (15) INIT(0), /* LEN OF COMPARE DATA IN BYTES*/
       3 DATA BIT(320); /*40 BYTES MAXIMUM*/
```

where NODE is the node number of the node being described, PARENT is the node number of the predecessor, BSETID is the ID of the Bset which leads the parent node to this node (i.e., the 'arc'), GET indicates whether all, any or none of the qualified

occurrences of this Pset are to be retrieved, and PRED describes restriction of values of this node (i.e., a predicate). N, CN and SETOP are currently unused.

RETN returns occurrences of the tree. The return data is a link list of the RETN return control blocks. Every control block in the link list describes one of the primitive data elements retrieved. The format of the return control block is:

```
DCL 1   RETN_RTN BASED(P),
      2 LEN     FIXED BIN (15) INIT (4),
      2 CBTP FIXED BIN (15) INIT( 26 ),
      2 PTR PTR INIT (NULL),
      2 RTN_CODE FIXED BIN INIT (0),
      2 N FIXED BIN (15);/*NUMBER OF INSTANCES OF ROOT NODE RETURNED*/
DCL 1  RETN_RTN1 BASED(P),
      2 LEN     FIXED BIN (15) INIT (46),
      2 CBTP FIXED BIN (15) INIT( 27 ),
      2 PTR PTR INIT (NULL),
      2 NODE FIXED BIN (15), /*NODE NUMBER*/
      2 INDX FIXED BIN (15), /*IF MULTI OCCUR*/
      2 DLEN FIXED BIN, /*LEN OF DATA IN BYTES*/
      2 DATA  BIT(320);  /*DATA RETURNED*/
```

As an example, suppose two employees are found to satisfy the previous query, the link list representing the return will look like that in the following figure.

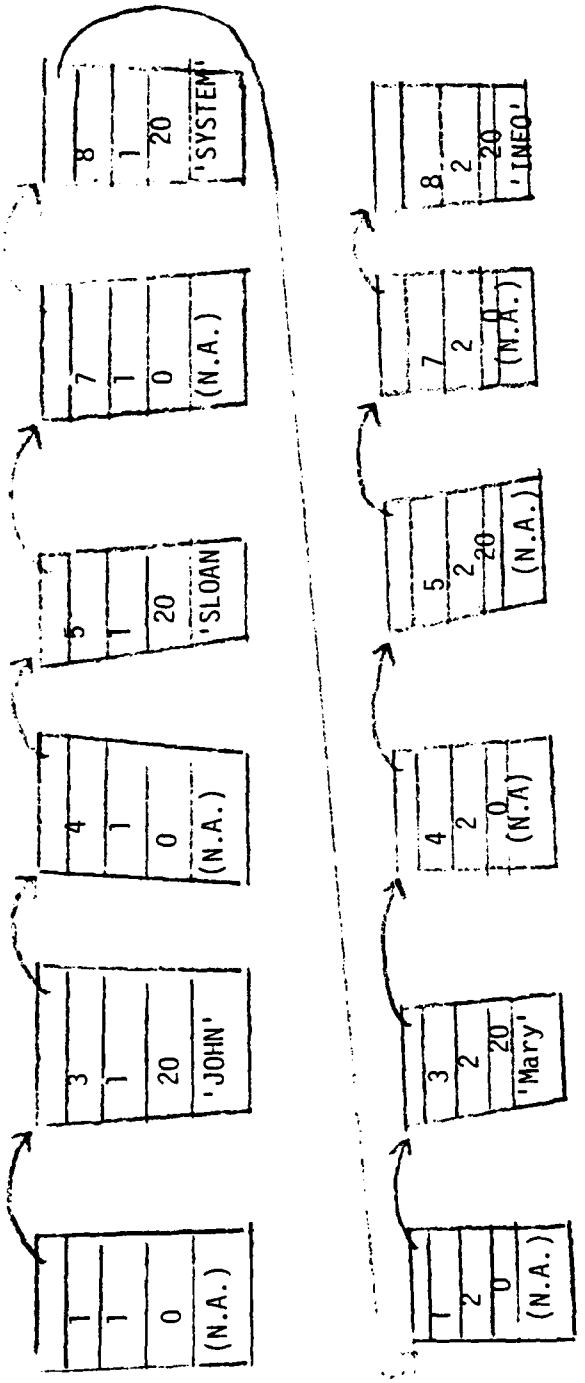


Figure: Examle RETN Return Control Block Link list (Based on the RETN Request shown in the Previous Figure)

This module, given the RETN request, uses a very simple method to traverse the database. It first visits the root node and obtains the first occurrence of the root node. It then verifies the occurrence of the rest of the tree against the predicate (if any). If the occurrence is found to be valid, a subroutine BUILD is called to generate the data and chain them to the return link list. If the occurrence is not valid, then it discards the current occurrence and gets the next occurrence of the root node. It continues this procedure until all occurrences in the root node are examined, or when one valid occurrence is found if the GET field of the root node in the request tree indicated 'ANY'. The way it verifies an occurrence is to follow the RETN tree to establish occurrences of its associated elements and to verify their content against their predicates. If any mismatch is found, the current occurrence is considered invalid and a new root occurrence started.

This module uses several subroutines to complete its procedure. RETP is called to retrieve the ID of the next P-element of a Pset on the forward chain of the Pset; RETB retrieves the ID of the associated P-element in a certain Bset of a P-element whose ID is given; VERIF verifies the value of a P-element against a predicate; and BUILD builds the RETN return control block link list based on a data structure CURSOR, which contains the ID's of P-elements of one occurrence of the tree. In addition, it calls GPCT and GBCT to obtain catalogue entries of relevant Psets and Bsets.

UPDN. This module (UPDate at iNternal schema level) creates, deletes and modifies data elements and their relationship in the database. Like RETN, the update argument is expressed in a tree called an UPDN request tree. The UPDN tree is slightly different from RETN tree in that the operations are different. However, the general fromat, i.e., a link list of control blocks each describing a node in the tree, is the same.

Update operations in a database ultimately include creating, deleting and modifying primitive data elements and their relationship to others. To alleviate the caller from the burden of creating individual P-elements one by one and then creating linkages among them one by one, the tree form enables it to express a meaningful aggregate of them in one UPDN request. For example, a database update request 'Remove employee John from Project A' can be translated by the levels above this level into a UPDN tree shown in the following figure.

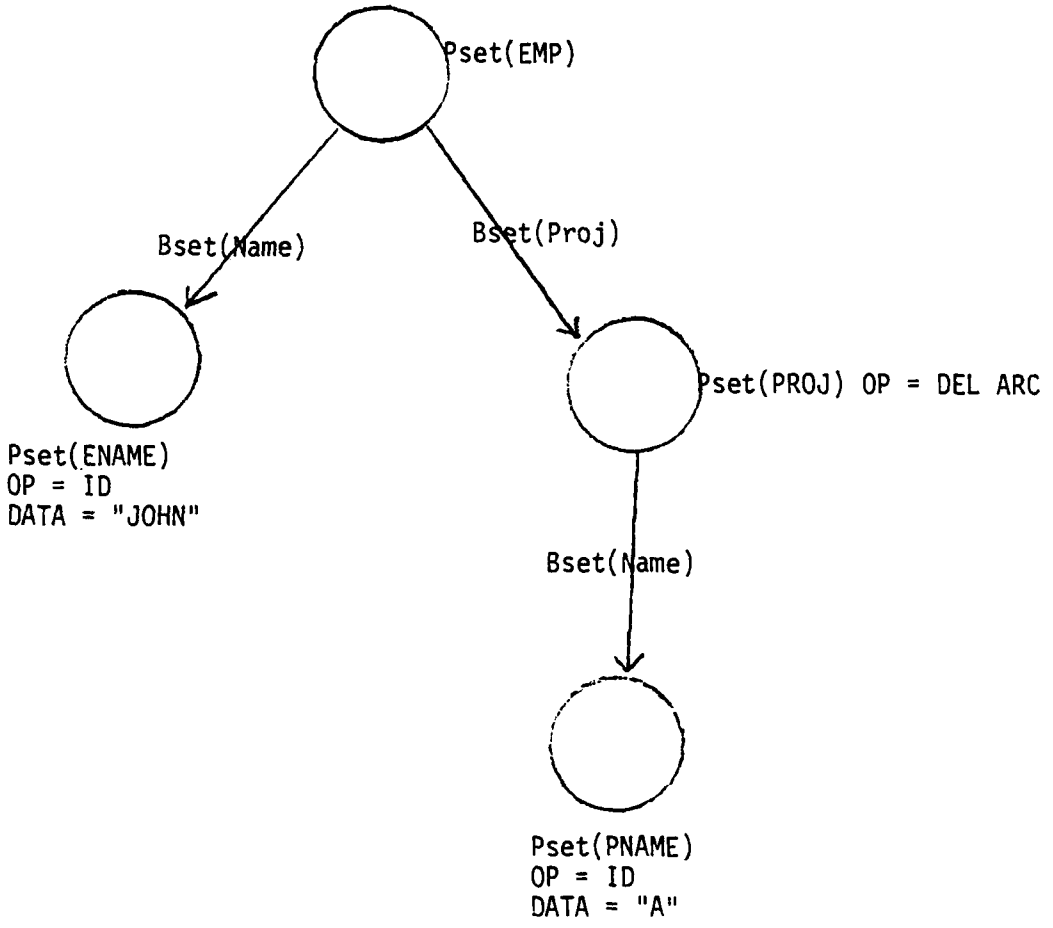


Figure: Example of a UPDN Tree

As shown in the figure, each non-root node carries with it an identifier for the arc (i.e., the Bset ID), an operation (i.e., OP) and possibly a data (i.e., ='John'). The possible operations are Insert arc, Delete arc, Replace arc, Create P-element and then insert arc, Delete arc and also the P-element, Replace value of P-element, and Identity. The identify operator is used when the node does not require any update but is included in the tree for the purpose of identifying the occurrence of its predecessor nodes. The root node operations are Delete, Create and Modify. (These operations are declared in the data structure UPDNOP.) In our example, the operation involved is to 'de-associate' the P-element representing employee John from the P-element representing Project A. Therefore an arc between these two P-elements must be deleted. However, to 'pin point' the occurrence of these two P-elements, two ID nodes are used (i.e., ENAME='John' and PNAME='A'.)

The UPDN request is expressed as a link list of control blocks UPDN_ARG. Declaration of UPDN_ARG is as follows:

```
DCL 1 UPDN_ARG   BASED(P),
      2 LEN      FIXED BIN (15) INIT (56),
      2 CBTP FIXED BIN (15) INIT( 30 ),
      2 PTR PTR INIT (NULL),
      2 NODE FIXED BIN (15) INIT(1),/*NODE NUMBER OF UPDN TREE*/
      2 PSETID BIT (32), /*PSETID OF THE NODE*/
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/
```

```
2 BSETID BIT (32), /*BSETID OF PARENT PSET LEADING HERE*/
2 OP FIXED BIN (15), /*OP CODE FOR UPDN; SEE UPDNOP FOR DETAIL*/
2 DLEN FIXED BIN (15) INIT(0), /*IN NUMBER OF BYTES*/
2 DATA BIT(320); /*MAX 40 BYTES OF IMMED DATA */
```

where NODE, PSETID, PARENT and BSETID have the same meaning as those in RETN_ARG described above, OP is the operation code, and DLEN and DATA are data to be created or data used to identify itself and others.

The UPDN_RTN control block returns a return code either confirming the operation or indicating problems encountered in the processing.

The method used by UPDN to process a request is as follows: 1. Make a pass through the UPDN request to fill up a data structure TEMP which is used to remember all nodes which have any child node whose operation code is 'ID' (i.e., Identity); 2. Use TEMP, working from the bottom of the tree to top, resolve all predicates; reduce the original tree to a 2-level tree without any ID-operation nodes; 3. Pass the reduced tree to subroutines CRTN, DELN or MODN depending on UPDN.OP at the root node.

The UPDN module makes use of RETN to resolve identity of nodes which have child nodes with OP code 'ID'. Subroutines MODN, DELN and CRTN will further call CRTP, CRTB, DELP,DELB, REPP, and REPB

subroutines to perform actual update of BEU's based on the OP codes of the nodes in the reduced tree.

5.2.4 INITIALIZATION AT INTERNAL SCHEMA LEVEL

NINIT. This module (iNternal schema level INITialization) is invoked during IPL by the upper level to initialize system databases. It first invokes MINIT which performs initialization of all levels below. If initialization is 'file', then MINIT will return a one-word key which NINIT can use to recover its system databases by retrieving a BEU using the key as its ID. This BEU contains values for PCATID, BCATID and another key that NINIT shall return to the upper level. In this way NINIT sets up the static variable PCATID and BCATID, which are to be read-accessed by several modules in this level when normal processing starts.

If the initialization is 'new', then NINIT has to bootstrap its system databases. It does this by allocating the PCAT catalogue entry BEU and directly calls subroutine CRT1 to store it in the Storage Hierarchy. (CRT1 is a subroutine which interfaces with the CRT procedure at the MM Level.) Once PCAT catalogue entry is in the database, NINIT uses DEFP to define the second Pset of the system database: the BCAT Pset. After both

Psets are defined, PCATID and BCATID values will be set and NINIT then returns.

NINIT also provides an interface for the upper level to store a one-word key for that level. This key, together with PCATID and BCATID, will be stored as a BEU in the Storage Hierarchy. The ID of this BEU then becomes the key of this level, and is stored by calling the MINIT Store Key function.

5.2.5 SUBROUTINES AT INTERNAL SCHEMA LEVEL

Most of the subroutines in this level have been briefly described in the previous subsecton. There are 3 for catalogue management: GPCT (Get Pset CaTalouge entry), GBCT (Get Bset CaTalogue entry) and UPCT (UPdate Pset CaTalogue entry), 4 for retrieval operations: RETP (RETrieve next P-element), RETB (RETrieve the Binary associated element), VERIF (VERIFy) and BUILD, 6 for update operations: CRTP (CReaTe P-elemtn), CRTB (CReaTe Binary association), DELP (DELete P-element), DELB (DELete Binary association), MODP (MODify P-element) and MODB (MODify Binary association), and 4 for interfacing with the MM Level: CRT1, DEL1, REP1 and RET1. The last group each interfaces with one procedure at the MM Level, and basically takes a BEU as input and generate appropriate request to be passed down.

5.3 ENTITY LEVEL

This level processes semantics of the database. It accepts definition of entity sets, attributes and constraints and responds to commands to manipulate entities.

5.3.1 DATA STRUCTURES AT ENTITY LEVEL

This level views the world as a collection of entities and attributes of these entities. This entity network data model has been described in section 2.2.

Mapping between the entity network and the data structures implemented in the Internal Schema Level is as follows. Every node in the entity network graph is implemented as a Pset, and every arc in the graph a Bset. While the Entity Level is responsible for making these Psets and Bsets known to the Internal Schema Level (through DEFP and DEFB calls), it is not concerned about how data in these sets are actually stored.

5.3.2 SYSTEM DATABASES AND DATA DEFINITION MODULES AT ENTITY LEVEL

This level keeps two catalogues: the entity catalogue and the attribute catalouge. An entry in the former describes an entity set and an entry in the latter describes an attribute of an entity set. These two catalogues are themselves implemented as two entity sets: E*ESET and E*ASET. They are displayed in the following figure.

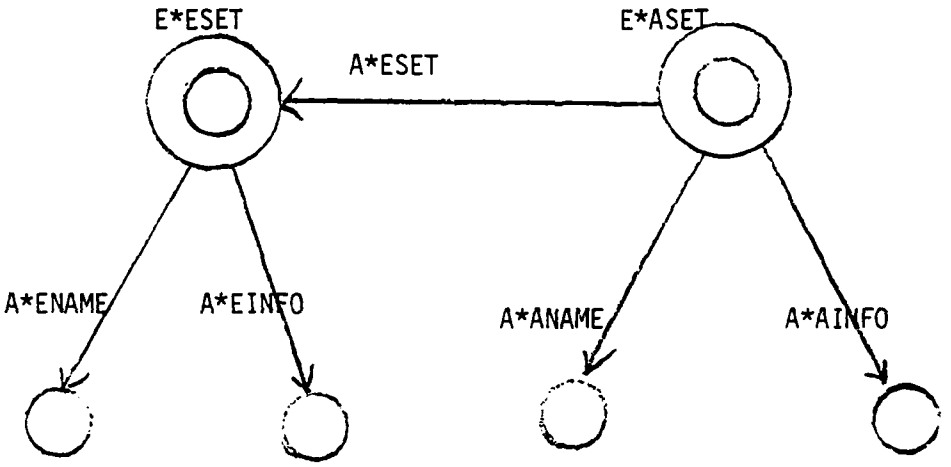


Figure: Catalogue structure at Entity Level

E*ESET has two 'attributes' A*ENAME and A*EINFO. The former contains the name of the entity set this catalogue entry is describing, and the latter contains the ID of the Pset implementing this entity set, which is provided for by the Internal Schema Level in response to a DEFP request.

E*ASET has three 'attributes'. A*ANAME conatins the name of the attribute. A*AINFO, an encoded attribute, contains a data structure declared as follows:

```
/** CATALOGUE STRUCTURES AT THE ENTITY LEVEL****/
DCL 1 AINFO,
     2 BSETID BIT (32),  /*BSETID OF THIS ATTRIBUTE**/
     2 PSETID BIT (32),  /*PSETID OF TARGET NODE**/
     2 ENAME CHAR (8), /*IF TARGET IS A ENTITY SET*/
     2 MAX FIXED BIN (31),
     2 MIN FIXED BIN (31),
     2 MLEN FIXED BIN,
     2 FUNC CHAR(1),
     2 TYPE CHAR (1),
     2 VTYPE CHAR (1);
```

where BSETID is the ID of the Bset implementing the association between the Pset that implements the entity set this attribute belongs to and the Pset that implements the domain of this attribute. This ID is produced by the Internal Schema Level in response to a DEFB request. PSETID is the ID of the Pset

implementing the domain of the attribute. ENAME is the name of the domain entity set if this attribute is an entity attribute. TYPE specifies the type of this attribute (i.e., entity or value). If it is a value type attribute, VTYPE specifies the value type (i.e., character string or numeric), and MLEN specifies the maximum length in terms number of characters or number of digits this attribute will have, and MAX and MIN provide the value range of the numeric attribute. These parameters are given by the DBA during the data definition session described in section 2.2.

The third attribute of E*ASET is A*ESET. It is an entity attribute whose domain is the E*ESET entity set. This attribute has a relationship type many-to-one, which means that many E*ASET entities may be related to one E*ESET entity. This attribute specifies the entity set this attribute belongs to.

Two T-proc's form the data definition interface at this level: DEFE (DEFine Entity set) and DEFA (DEFine Attribute). They are invoked by the DDB module in the User Interface Level. Both modules will check a definition request against duplicate names. They will then call subroutines DFP1 and DFB1 to formulate DEFP and DEFB requests to be passed to the Internal Schema Level. To update catalogues, both modules make use of UPDE, the data manipulation module at this level, since the catalogues are themselves entity sets. This greatly simplifies the logic of these two modules.

Another module, SHWE (SHoW Entity level catalogue) will, when invoked, retrieve the catalogue entries and decode the attribute information for display by the User Interface Level. In retrieving catalogue entries, all three modules (DEFE, DEFA and SHWE) make use of subroutines GECT, GACT and ALLA, which will be described later.

5.3.3 DATA MANIPULATION AT ENTITY LEVEL

RETE. This module (RETrieve Entities) is passed a tree-like retrieval request. The root node describes the entity set of which information is being requested, and the leaf nodes describe the attributes of interest of this entity set. Predicates on the attributes, if any, are passed along with the leaf nodes. The request is represented by a link list of RETE calling control blocks RETE_ARG. Each control block describes one node in the retrieval tree. Declaration of RETE_ARG is as follows:

```
DCL 1 RETE_ARG   BASED(P),
      2 LEN      FIXED BIN (15) INIT (61),
      2 CBTP FIXED BIN (15) INIT( 21 ),
      2 PTR PTR INIT (NULL),
      2 NODE FIXED BIN (15) INIT(1), /*NODE NUMBER*/
      2 NAME CHAR (8), /*ANAME OR ENAME*/
```

```
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/
      2 GET BIT(8) INIT ('00000000'B), /*SEE RETEGET FOR DETAIL*/
      2   N   FIXED   BIN  INIT(0),  /*NUMBER  OF  RELEVANT  'OR'
PREDICATES*/
      2 PRED (1)/*IN THIS VERSION ONLY ONE PREDICATE EACH TIME IS
          ALLOWED; IN THE FUTURE MAY USE REFER OPTION TO N*/,
        3 CN FIXED BIN (15) INIT(1),/*CONDITION NUMBER*/
        3 SETOP BIT(8), /*RELEVANT IF BSET IF MULTI-TARGET;
                         NOT USED IN CURRENT VERSION*/
        3 OP BIT(8), /*COMPARISON OPS, SEE RETEOP FOR DETAIL*/
        3  DLEN  FIXED BIN (15) INIT(0), /* LEN OF COMPARE DATA IN
BYTES*/
        3 CDATA CHAR (40);/*40 BYTES MAXIMUM*/
```

where NODE and PARENT have similar meaning as those in the RETN_ARG control block (section 5.2.3), NAME is either the entity set name if it is a root node, or the attribute name if it is a leaf node, GET indicates whether to get all, any or none of the qualified occurrences of this attribute. Under PRED, OP refers to a comparison operator (i.e., >, < or =), DLEN and CDATA are the length and the character form of the predicate data. N, CN and SETOP are currently not used.

The RETE module retrieves catalogue entries of all entity sets and attributes involved in the RETE request by subroutine calls to GACT and GECT, which retrieve attribute information and entity set information. Any undefined name found will cause the

request to be rejected with the unfound name pointed out. Catalogue information is also used to verify and transform the predicate values passed along with the request. Transformation is necessary since the internal representation of data may not always take the charactaer form. Data type transformation and verification are done through a subroutine VTPE. Once all these are done, a RETN request will be formulated and passed to the RETN module at the Internal Schema Level. When the latter returns, its answer set will be used to generate RETE's answer set, which is a link list of RETE_RTN control blocks. The RETE_RTN control block is similar to that of the RETN_RTN control block except that DATA is now in character form ready to be displayed.

UPDE. This module (UPDate Entities) is invoked to create, delete and modify entities in the entity sets already defined. UPDE is also passed a tree-like request, where the root node describes the entity set of which occurrences are to be updated, and the leaf nodes describe the attributes which are either used to identify the occurrences to be updated, created or added to. The UPDE request and its return control blocks are similar to those of the UPDN module.

UPDE performs validity checking on the attributes being passed. Any violation of validity constraints defined for the values of the attributes will cause the request to be rejected entirely, with the faulty attribute pointed out. UPDE also calls

subroutines GECT, GACT and VTPE for catalogue retrieval and data type verification and transformation. After all these are done, UPDE formulates a UPDN request to be passed to the Internal Schema Level to update P-elements and their relationships.

UPDE returns a return code confirming the success of the operation or pointing out why it has failed.

VNME. This module (Verify NaMEs) is invoked by some User Interface Modules to verify the entity names and attribute names a user has typed in before he enters the data. This is provided so that a user will get a more responsive system interface. The calling control block VNME_ARG contains an array of names structured in a tree form. VNME will verify these names against the catalogues at the Entity Level and point out problems if there is any.

5.3.4 INITIALIZATION AT ENTITY LEVEL

VINIT. This module initializes system databases at this level. Like NINIT, when it is invoked, it first calls the initialization module at the lower level to initialize all the lower levels. Depending on the method of initialization (file or new), it either receives a one-word key back based on which it

recovers its system databases, or it has to bootstrap the system databases into the Storage Hierarchy.

To bootstrap its system databases which are the first two entity sets of the system, VINIT calls DEFP 6 times and DEFB 5 times to set up the Psets and Bsets involved in the catalogues. The 11 ID's thus generaged are then stored in yet another special Pset (the 7th Pset that VINIT has defined) by calling UPDN. The ID of the 7th Pset is then stored as the key of this level by invoking the Store Key function of the NINIT procedure. The 11 ID's are also retained in a static array KEY which can be read-accessed by the catalogue management subroutines GACT, GECT and ALLA at this level.

Another subroutine EBOT (Entity catalog BOotstrap) is called by VINIT to put into these 6 Psets and 5 Bsets the definition of the catalogues. (Since the catalogues are themselves entity sets with attributes, their definitions will be the first to appear as data in these Psets and Bsets.) EBOT uses 7 UPDN calls to accomplish this task.

If initialization is from an existent database, VINIT will retrieve the 11 ID's back by accessing the Pset whose ID is returned to VINIT by NINIT as the one-word key.

5.3.5 SUBROUTINES AT ENTITY LEVEL

The six subroutines have been introduced briefly in the previous subsections. Three are catalogue management subroutines: GECT (Get Entity CaTalogue entry), GACT (Get Attribute CaTalogue entry) and ALLA (get ALL Attributes). ALLA is invoked by SHWE when the user wishes to find out all attributes defined for a particular entity set. All three modules make use of the KEY array to get the ID's of the Psets and Bsets forming the catalogues, and formulates a RETN request to obtain these data. Subroutines for retrieving attribute catalogue entries also perform the task of decoding a bit string into elements in the AINFO data structure for direct use by the caller.

Theoretically there is no need for a set of special retrieval routines just to retrieve the catalogue entries. Since the catalogue entries are themselves attributes of entities in the database, RETE should be able to accomplish the same task. These special routines are devised mainly for performance reasons. RETE, as it turns out, is a very expensive module. Using RETE to retrieve catalogue entries would degrade performance dramatically. While performance of the system is not a primary concern at this stage, it was felt that this deviation from theoretical elegance was worthwhile in the current implementation. Performance of a multi-layered database

architecture is a future research issue which may provide clues to whether this deviation can be (or should be) avoided.

Another two of the subroutines, DFB1 and DFP1, are used to generate DEFB and DEFP requests in the proper format to be passed to the next level. Finally, subroutine VTPE (Verify TyPE) is used to verify and transfrom data passed along with the RETE or UPDE requests.

5.4 THE USER INTERFACE LEVEL

This level interacts directly with the user. It does not create or use any system database. Its function is primarily one which parses users' requests and generates in proper formats requests to be passed down to the Entity Level. Therefore the usual classification of modules into data definition, data manipulation and initialization modules is not used here.

5.4.1 USER INTERFACE MODULES

USER. This module is invoked when a user 'logs on'. (Presently there is no security checking facility.) USER simply asks for the subsystem the user wishes to get in. Only DBA subsystem is currently available. USER then passes control to the DBA module.

DBA. This module (Data Base Administrator session) takes control when a user enters the DBA session. It calls various subsroutines depending on what the user wishes to do: data definition (DDB), data definition query (DDQ) or data manipulation (DDM). These three subroutines are described in the following paragraphs.

DDB. This module (Data Definition for Base schema) is invoked to interface with a DBA's request to do data definition for the base schema. (A base schema is a schema which describes the integrated database in terms of constructs in the entity network data model. It is different from a view schema which describes a view in terms of a data model which is not necessarily the base data model. Presently no view facility is available.) A user may define an entity set and then its attributes. He may also add attributes to an entity set defined previously. The DDB module interfaces with the user through a panel facility. The panel facility will display pre-defined panels at the user's terminal and allow the user to respond by filling out blanks on the panel.

This module will formulate DEFE and DEFA requests to be passed down to the Entity Level based on parameters the user specifies in the panels.

DDQ. This module (Data Definition Query) is invoked to display content of data definitions. Its major task is to formulate properly a SHWE request to be passed to the Entity Level. When SHWE returns, DDQ displays the result at the user's terminal.

DMB. This module (Data Manipulation for Base data) is invoked when a user enters a data manipulation session. It calls subroutines DMQ, DMC, DMM and DMD based on the type of manipulation operations. These subroutines are described below.

DMQ. This module (Data Manipulation - Query) accepts a user's query and parses it into meaningful tokens to be placed in a RETE request. It calls subroutin LEX to break the input characater string into tokens. Then it uses subroutine LEXT to recognize the tree structure of the query. Once data structures describing the tree have been produced by LEXT, DMQ formulates a RETE request and passes it to the Entity Level. Upon receiving return data from the latter, DMQ calls subroutine PRNT to display the result at the user's terminal.

DMC. This module (Data Manipulation - Creating entities) accepts a user's request to create entities in pre-defined entity

sets. The user first provides a list of attributes whose values will be provided when an entity is created. DMC will, like DMQ, call LEX and LEXT to build a tree out of the list of attribute names and formulate a VNME request to be passed to the Entity Level. The latter validates these names against the entity level catalogues. If no error is detected in the attribute list, DMC then asks the user to enter data for these attributes. Every line of data entered will cause one entity to be created. DMC uses subroutine UPE1 to input data lines and to formulate UPDE requests. Violation of constraints on data types or data values will be signaled through a return code from UPDE and DMC will display proper error messages based on the return code.

DMM and DMD. These two modules (Data Manipulation - Modify entities and Data Manipulation - Delete entities) are similar to DMC in thier operations.

5.4.2 OTHER SUBROUTINES AT USER INTERFACE LEVEL

This subsection describes the 5 subroutines used by modules described in the previous subsection.

LEX. This module (LEXical analyzer) breaks input lines into tokens. Token delimiters recognized are ',' and ':'. Delimiters

enclosed in quotes are not treated as exceptions. More than one line of input can be strung together by ending previous lines in back slashes. Blanks can be optionally taken out of the input character string. However, blanks enclosed in quotes are not suppressed. This module returns an array TOKEN which contains the tokens and their lengths. (Note that the fact that delimiters found in quotes are not taken as exceptions means that this current implementation of lexical analyzer cannot accommodate data values that contain these special characters.)

LEXT. This module (LEXical Tree structure) is given the array TOKEN produced by subroutine LEX and is asked to identify the hierarchical structure of names and data values implied by the nested parantheses. For example, if TOKEN(1) is 'EMPLOYEE(', TOKEN(2) is 'EMPNAME' and TOKEN(3) is 'EMPADDR)', then EMPNAME and EMPADDR are the children names of the name EMPLOYEE. This module is also required to recognize non-name tokens. There are two types of non-name tokens: the modification operators (MOP) and the predicate values. The former is distinguished from regular name tokens by its special starting character '-'. The latter always trails a comparison operator ('=', '>', '<') behind a name. Therefore if a token is found to contain a comparison operator, it is broken into a name token and a value token. If a value is surrounded by quotes, the quotes are removed.

This module returns several data structures describing its parsing results. Array T1 contains the hierarchical structure of

names, array PRED contains value tokens and array MOP contains modificaton operators found in the input character string.

UPE1. This module (UPdE interface) is used by data update modules (DMC, DMD and DMM) to get input data and to formulate a UPDE request for each input line. The caller will pass to UPE1 the tree structure of enitity and attribute names that have been recognized by LEXT as well as the update operators (e.g., insert, delete, replace) corresponding to the input data items. UPE1 makes use of these data structures, calls LEX to break the input line into data items, and formulates proper UPDE requests to be passed to the Entity Level.

VNM1. This module (VNMe interface) is invoked by data update modules (DMC, DMD and DMM) to formulate proper VNME requests to be passed to the Entity Level. This is done to confirm the legality of entity and attribute names before accepting the update data from the user. This module is passed a data structure describing the tree structure of the names and will return a return code indicating any problem found.

PRNT. This module (PRiNT) performs the task of displaying the result of a user query. It is passed a pointer pointing to the RETE_ARG chain and a pointer pointing to the RETE_RTN chain and will use information contained in these chains to display header lines and data lines at the terminal.

5.5 THE DRIVER MODULE - FSTV

The FSTV module is invoked by the Control Structure when starting the simulation run. It is the driver program of the Functional Hierarchy. The purpose of this module is to start up initialization of the entire Functional Hierarchy and to invoke the user session.

System Traces. FSTV first calls subroutine SETFH which sets up run time parameters. The only run time parameters used in the current implementation are the trace options. There are five types of trace options. The TCALL trace option will display messages when a T-proc is invoked as well as when it returns. This trace can be selected on a level basis. It is implemented through the TCALL program, since all T-proc invocations necessarily go through TCALL.

The system error message trace will display messages when an unusual situation is detected. This normally signals that some system software errors have been detected. This trace can also be turned on and off on a level basis.

The third type of trace is the memory request trace. If this is turned on, every data element that has been created, deleted or updated in the Storage Hierarchy will be displayed along with its virtual address. It displays these data elements using the

BEU format. It is implemented within subroutines CRT1, RET1, DEL1 and REP1.

The fourth type of trace is the Internal Schema Level data manipulation request trace. It displays all the requests that go into the RETN and UPDN modules. This trace is implemented within these two modules.

The last type of trace is the timing report option. It keeps track of the virtual CPU time consumed by each T-proc (including subroutine calls within the T-proc) and the number of times a T-proc is invoked. A sample timing report is shown in the following figure. This option is implemented through the TCALL module, and the result is displayed at the end of the run by the program FSTV.

LEVEL	PROCNAME	COUNT	TOT ELAPSED TIME	TOT RUN TIME	RUN TIME/INVOCATION
1	USER	1	1505283	80955	80955
2	VINIT	1	123976	5233	5233
2	VNME	0	0	0	0
2	DEFE	0	0	0	0
2	RETE	1	1146343	17482	17482
2	UPDE	0	0	0	0
2	DEFA	0	0	0	0
2	SHWE	1	1424328	11921	11921
3	DEFP	0	0	0	0
3	DEFB	0	0	0	0
3	RETN	8	1503249	567142	70892
3	UPDN	0	0	0	0
3	NINIT	1	10419	5637	5637
4	RET	138	939625	939625	6808
4	DEL	0	0	0	0
4	REP	0	0	0	0
4	CRT	0	0	0	0
4	MINIT	1	1264	1264	1264
5	L1	801	180119	180119	224

Figure: Sample Timing Report

After SETFH is finished, FSTV invokes VINIT to initialize the Functional Hierarchy. Then FSTV invokes USER, the T-proc at the User Interface Level to start up the user session.

6.0 CONCLUDING REMARKS AND FUTURE DIRECTIONS

This report describes the current implementation of the Software Test Vehicle of the Functional Hierarchy of the INFOPLEX database computer. It provides an introduction to the STV project and the relationships among its various components, and documents the programming conventions used in FSTV.

This current version of FSTV is implemented with future research and expansions in mind. It is not meant to provide detailed performance statistics. Nor is it meant to incorporate all the functionalities that a large DBMS should have. However, it has provided some interesting insights into issues that ought to be explored further, and a basis for incorporating future efforts.

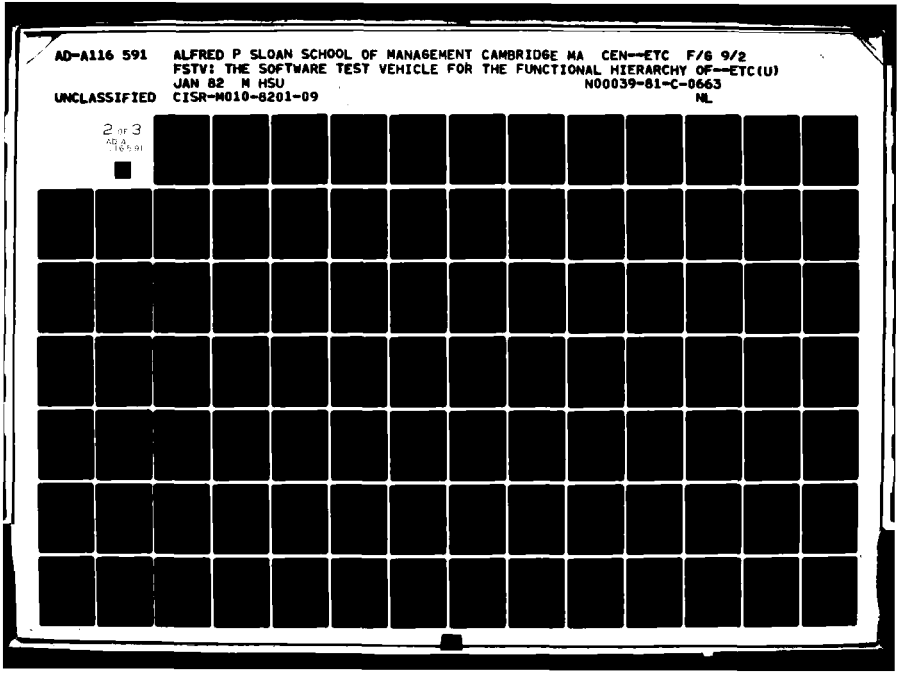
6.1 MEMORY MANAGEMENT LEVEL REVISITED

Segmentation and garbage collection are among the more important functions that are missing in the current

implementation. An interface which allows the upper level to define segments and to request that a certain element be stored in a pre-defined segment will produce a much more logical grouping of data elements in the virtual memory address space. This means that the Memory Management Level will maintain its own system database which describe the availability of each segment. Where and how this system database is maintained are also issues that need to be resolved.

6.2 INTERNAL SCHEMA LEVEL REVISITED

The purpose of this level is to provide the upper levels with a powerful data structure management facility. However, capabilities implemented in the current version leave very much to be desired. In terms of functionalities, the binary association types, currently limited to one-to-one and many-to-one, should be expanded to include one-to-many and many-to-many. Accompanying this expansion is the enrichment of the set of primitives used to express retrieval and update predicates. Set-oriented primitives such as INCLUDE, BELONG_TO, ALL and ANY are more efficiently processed at this level than at a higher level, therefore they ought to be included in the RETN requests. The predicate structure should also include the OR boolean operator. In addition, the INVERSE function should also


AD-A116 591
ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA CEN--ETC F/G 9/2
FSTV: THE SOFTWARE TEST VEHICLE FOR THE FUNCTIONAL HIERARCHY OF--ETC(U)
JAN 82 M HSU
N00039-81-C-0663
UNCLASSIFIED
CISR-M010-8201-09
NL
2 OF 3
AD A
116 591

be provided for defintion of a binary association which is the inverse of one already defined. This provides more flexibility for the user in phrasing a query while leaves the access path selection problem entirely to the Internal Schema Level.

In terms of interface, the current definition and manipulation interface structure should be more flexible. For example, several Bsets whose source Psets are the same may be defined in one DEFB request. Additional construct defintion may be desirable. For example, an often used retrieval tree may be defined to be a construct such that future retrieval requests may simply cite the ID of that construct without constructing the same tree again. These enhancements will improve convenience of use as well as efficiency of operations.

Those described above are potential improvements that the upper level can see. Equally important are the improvements of internal processing. Intelligent query decomposition and more powerful search mechanisms will have to be incorporated to cope with problems of traversing a large data base. A preliminary design which further decomposes this level into two levels has been proposed, where the lower level is responsible for fast search in a flat file and the upper level is responsible for query decomposition and linkage of data. A major 'surgery' is expected on the current implementation in order to bring about these improvements.

6.3 THE ENTITY LEVEL REVISITED

The main purpose of this level is to build semantics and enforce constraints. One powerful arm in building semantics, the Virtual Information Facility, is largely missing in the current implementation. This facility may run parallel to the current level on a 'pipeline' basis. That is, retrieval requests will first pass through a 'virtual information filter' which transforms the request into one which contains only references to actual, stored entities and attributes. When the latter has been retrieved by the RETE facility, the return data will go through the second stage of the virtual information facility, a 'virtual information encoder', which will compute the virtual information values based on virtual information definitions. Architecturally this facility could lie above the Entity Level and form a Virtual Information Level; however, data structures being dealt with in this facility are the same as those in the Entity Level. That is to say, there will not be another level of data abstraction.

6.4 LOCATION AND MANAGEMENT OF CATALOGUES AND WORKING BUFFERS

There exists a spectrum of design philosophies for a stratified architecture. On the one extreme, all software at one

level may be treated as 'data' at the next level, effecting a hierarchy of levels each of which implements its 'virtual address space' at the next lower level; on the other extreme, the software at one level is divided into two distinct parts; one part (i.e., code and catalogues) always exists at the local level, while the other part (e.g., user data) always exists at the next level (and eventually in the Storage Hierarchy.)

The strategy used in the preliminary design of the Functional Hierarchy is somewhere in between: catalogues and regular data are stored as data at the next lower level while program modules and their contexts reside within a level.

Intuitively, some duplication of cataluge and data within a level may be desirable based on performance considerations. As the FSTV runs show, catalogue retrievals at the higher levels consume a great percentage of the system processing power. This overhead may prove to be even more costly if the database is large and more functionalities are incorporated.

Another aspect in this issue is that of a 'stored procedure' at the lower levels. In other words, in addition to storing data at the next lower level, a level may choose to declare some frequently encountered requests to the next lower level, so that the next lower level may have some specialized, more efficient code to process them. This is analoguous to a 'compiled query' mechanism adopted in System R.

Yet another issue is that of a 'program module overfow'. As a level develops to include more functions, its program size may become too large to be all stored economically within the level's local memory. Less frequently used modules may be removed and stored as data at the next lower level, which in turn will store them as data at the next lower level, until it is stored as part of the data in the Storage Hierarchy. Similar arguments may be devised for a program context of a request that requires a large amount of buffer space.

However, while these are issues FSTV has exposed, they are not likely to be resolved without a rigorous investigation of tradeoffs among alternatives. It also requires efforts in designing additional algorithms that will ensure data integrity in the case of data duplication.

REFERENCES

Blumberg81: Blumberg, B. INFOSAM - A sample database management system. Master Thesis, Sloan School of Management, MIT (May 1981)

Hsu80: Hsu, M. A preliminary architectural design for the Functional Hierarchy of the INFOPLEX database computer. TR #:M010-8011-05, Sloan School of Management, MIT (Nove 1980)

Madnick79: Madnick, S.E. The INFOPLEX database computer: Concepts and directions. Proc. IEEE Computer Conference (Feb 1979)

To81: To, T. Shell: A simulation for the Software Test Vehicle of the INFOPLEX database computer. Bachelor Thesis, Sloan School of Management, MIT (may 1981)

APPENDIX I

SAMPLE TERMINAL SESSION (a)

Sample Terminal Session

```
fstv isa;-90k)
ENTER TCALL TRACE OPTION (4 BITS)

:
0000
ENTER SYSTEM ERROR MESSAGE TRACE OPTION (4 BITS)

:
0000
ENTER MEMORY REQUEST TRACE OPTION (1 BIT):
0
ENTER UPDN/RETN REQUEST TRACE OPTION (1 BIT):
0

--FUNCTIONAL HIERARCHY STV CONSOLE PROGRAM--

INITIALIZATION: FILE OR NEW:
FILE

FILE NAME?

:
TEST20   <- load file created in the previous session
FSTV CALLS VINIT

FSTV: VINIT RETURNS

SUBSYSTEMS: DBA, BV(BASE VIEW) OR RV(RELATIONAL VIEW)?

:
DBA

-- DATABASE ADMINISTRATOR (DBA) SESSION --


DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINITION QUERY (DDQ)?

:
DD

DD: BASE DATA (BASE) OR VIEW DATA (VIEW)

:
```

101

```
DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINITION QUERY (DDQ)?

:
DDQ
-- DATA DEFINITION QUERY SESSION --

ENTITY SET NAME? USE * IF ALL ENTITY SET NAMES ARE DESIRED.

:
*
NUMBER OF ENTITY SETS DEFINED                    5

PROJECTS
 DEPT
 EMPLOYEE
 E*ASET
 E*ESET

ENTITY SET NAME? USE * IF ALL ENTITY SET NAMES ARE DESIRED.

:
PROJECTS
ATTRIBUTE NAMES? SEPERATE BY COMMAS. USE * IF ALL ATTRIBUTES ARE DESIRED
:
*
ENTITY SET NAME          PROJECTS

ATTRIBUTE NAME   FUNCTION  TYPE  ENAME     VTYPE  MAX LEN  MAX VALUE  MIN VALUE

      UNDER          M:1     E  DEPT

      PROJNUM        KEY     V                 N
        5
      99999      -99999

      PROJNAME       1:1     V                 C
        20

ENTITY SET NAME? USE * IF ALL ENTITY SET NAMES ARE DESIRED.

:
EMPLOYEE
```

← Want all Entity Sets defined in the system

5 entity sets in the system

Want all attributes defined for entity set PROJECTS

3 attributes defined for PROJECTS

```
ATTRIBUTE NAMES? SEPERATE BY COMMAS. USE * IF ALL ATTRIBUTES ARE DESIRED

:
*
ENTITY SET NAME          EMPLOYEE

ATTRIBUTE NAME  FUNCTION  TYPE  ENAME     VTYPE  MAX LEN  MAX VALUE  MIN VALUE

      WORKS_IN        M:1     E  DEPT

      BOSS            M:1     E  EMPLOYEE

      AGE             1:1     V                N
        2
          99          10

      EMPADDR         1:1     V                C
       40

      EMPNAME         1:1     V                C
       20

ENTITY SET NAME? USE * IF ALL ENTITY SET NAMES ARE DESIRED.

:
DEPT
ATTRIBUTE NAMES? SEPERATE BY COMMAS. USE * IF ALL ATTRIBUTES ARE DESIRED

:
*
ENTITY SET NAME          DEPT

ATTRIBUTE NAME  FUNCTION  TYPE  ENAME     VTYPE  MAX LEN  MAX VALUE  MIN VALUE

      DEPTNAME        1:1     V                C
       20

      DEPTNUM         KEY     V                C
        5

ENTITY SET NAME? USE * IF ALL ENTITY SET NAMES ARE DESIRED.
```

```
:

DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINITION QUERY (DDQ)?

:
DM
-- DATA MANIPULATION SESSION --

DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
CRT <- enter CREATE session
CREATE: ENTER ENTITY SET NAME

:
DEPT
ENTER NAMES OF ATTRIBUTES SEPARATED BY COMMA

:
DEPTNUM,DEPTNAME
ENTER DATA:
15,'Sloan School'

:
14,Economics

:

CREATE: ENTER ENTITY SET NAME

: <- leave CREATE session

DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
QUE <- enter QUERY session
--QUERY SESSION--
```

```
TYPE YES IF NEED HELP

:

ENTER ENTITY SET NAME

:
DEPT
ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  COMMAS

:
DEPTNAME,DEPTNUM
ENTITY SET NAME          DEPT

 | DEPTNAME | DEPTNUM |

 | Economics          | 14    |

 | 'Sloan School'     | 15    |

ENTER ENTITY SET NAME

:
```

Answer to the query

create QUERY session

```
DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
MOD
TYPE YES IF NEED HELP

:

MODIFY: ENTER ENTITY SET NAME

:
DEPT
```

enter MODIFY session

```
ENTER MODIFICATION OPERATORS AND NAMES OF ATTRIBUTES SEPARATED BY COMMAS

:
-ID:DEPTNUM,-REP: DEPTNAME
ENTER DATA:
15,Sloan School
:

MODIFY: ENTER ENTITY SET NAME

:
DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
QUE
--QUERY SESSION--

TYPE YES IF NEED HELP

:

ENTER ENTITY SET NAME

:
DEPT
ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  COMMAS

:
DEPTNUM,DEPTNAME
ENTITY SET NAME          DEPT

 | DEPTNUM  | DEPTNAME |

 | 14       | Economics          |

 | 15       | Sloan School       |
```

the way to modify: identify by DEPTNUM, then replace DEPTNAME

change DEPTNAME of DEPT with DEPTNUM 15 to Sloan School

leave modify session

back to QUERY session

Answer to query

```
ENTER ENTITY SET NAME

:
DEPT
ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  COMMAS

:
DEPTNAME,DEPTNUM=15
ENTITY SET NAME          DEPT

 | DEPTNAME | DEPTNUM  |

 | Sloan School         | 15    |

ENTER ENTITY SET NAME

:
DEPT
ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  COMMAS

:
DEPTNAME='Sloan School',DEPTNUM
ENTITY SET NAME          DEPT

 | DEPTNAME | DEPTNUM  |

 | Sloan School         | 15    |

ENTER ENTITY SET NAME

:
DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
CRT
```

a query with a predicate

Answer for the query

Answer

Leave Query session

Enter Create session

```
CREATE: ENTER ENTITY SET NAME

:
DEPT
ENTER NAMES OF ATTRIBUTES SEPARATED BY COMMA

:
DEPTNUM,DEPTNAME
ENTER DATA:
6,EE-Computer Science
:
21,Humanities
:

CREATE: ENTER ENTITY SET NAME

:

DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
CRT
CREATE: ENTER ENTITY SET NAME

:
EMPLOYEE
ENTER NAMES OF ATTRIBUTES SEPARATED BY COMMA

:
EMPNAME,EMPADDR,AGE,WORKS_IN(DEPTNUM)
ENTER DATA:
Hoo-min Toong,E53-300,50,15
:
Mike Abraham,somewhere in Somerville,40,14
:

CREATE: ENTER ENTITY SET NAME
```

create two other DEPT entities

create 2 other EMPLOYEE entities

```
:

DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:
QUE
--QUERY SESSION--

TYPE YES IF NEED HELP

:

ENTER ENTITY SET NAME

:
EMPLOYEE
ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  COMMAS

:
EMPNAME,WORKS_IN(DEPTNUM,DEPTNAME)
ENTITY SET NAME          EMPLOYEE

 | EMPNAME   | WORKS_IN (DEPTNUM  | DEPTNAME) |

 | Mike Abraham          |    14    | Economics          |

 | Hoo-min Toong         |    15    | Sloan School       |

ENTER ENTITY SET NAME

EMPLOYEE
:
ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  COMMAS

:
EMPNAME, AGE<45 <- (another query with predicate)
ENTITY SET NAME          EMPLOYEE
```

| EMPNAME | AGE |

| Mike Abraham | 40 |

Answer to the query

ENTER ENTITY SET NAME

:

<- get out

DM: ENTER MANIPULATION COMMAND:

CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)

:

<- get out

DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINITION QUERY (DDQ)?

:

<- get out

-- END OF DBA SESSION --

SUBSYSTEMS: DBA, BV(BASE VIEW) OR RV(RELATIONAL VIEW)?

:

<- get out

-- END OF USER SESSION--

SAVE FILE: FILE NAME?

:

TEST21 <- Saved under TEST21

TEST ENDS

R; T=12.30/14.94 23:14:04
spool console stop

APPENDIX I

SAMPLE TERMINAL SESSION(b)

```
GENMOD FSTV MODULE Q ( FROM PLISTART
R; T=1.46/2.51 22:45:07
fstv isa(-90k)

ENTER TCALL TRACE OPTION (4 BITS)

:
0000 <-- turn off all traces
ENTER SYSTEM ERROR MESSAGE TRACE OPTION (4 BITS)

:
0000
ENTER MEMORY REQUEST TRACE OPTION (1 BIT):
0
ENTER UPDN/RETN REQUEST TRACE OPTION (1 BIT):
0

--FUNCTIONAL HIERARCHY STV CONSOLE PROGRAM--

INITIALIZATION: FILE OR NEW:
FILE

FILE NAME?

:
TEST19 <-- TEST19 already contains an initial database
FSTV CALLS VINIT

FSTV: VINIT RETURNS

SUBSYSTEMS: DBA, BV(BASE VIEW) OR RV(RELATIONAL VIEW)?

:
DBA

-- DATABASE ADMINISTRATOR (DBA) SESSION --

DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINITION QUERY (DDQ)?

:
DD

DD: BASE DATA (BASE) OR VIEW DATA (VIEW)

:
BASE
```

-BASE DATA DEFINITION SESSION-

YOU MAY DEFINE NEW ENTITY SETS OR ADD ATTRIBUTES TO EXISTING ENTITY SETS.

NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?

:

OLD ← a panel session followed which defined an AGE attribute for an existing entity set EMPLOYEE

ATTRIBUTE AGE DEFINED

PRESS ENTER TO CONTINUE

:

← Another panel session defined the BOSS attribute

ATTRIBUTE BOSS DEFINED

PRESS ENTER TO CONTINUE

:

NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?

:

NEW ← a panel session defined the DEPT entity set

ENTITY SET DEPT DEFINED

PRESS ENTER TO CONTINUE

:

← Attribute DEPTNUM defined here

ATTRIBUTE DEPTNUM DEFINED

PRESS ENTER TO CONTINUE

:

← attribute DEPTNAME defined here

ATTRIBUTE DEPTNAME DEFINED

PRESS ENTER TO CONTINUE

:

NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?

:
NEW
ENTITY SET PROJECTS DEFINED

Entity set PROJECTS defined here

PRESS ENTER TO CONTINUE

:

ATTRIBUTE PROJNAME DEFINED

attribute PROJNAME here

PRESS ENTER TO CONTINUE

:

ATTRIBUTE PROJNUM DEFINED

attribute PROJNUM defined here

PRESS ENTER TO CONTINUE

:

NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?

:
OLD
ATTRIBUTE WORKS_IN DEFINED

defined attribute WORKS_IN for EMPLOYEE

PRESS ENTER TO CONTINUE

:

NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?

:
OLD
ATTRIBUTE UNDER DEFINED

defined attribute UNDER for PROJECTS entity set

```
PRESS ENTER TO CONTINUE

:

NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?

:

DD: BASE DATA (BASE) OR VIEW DATA (VIEW)

:

DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINITION QUERY (DDQ)?

:

-- END OF DBA SESSION --

SUBSYSTEMS: DBA, BV(BASE VIEW) OR RV(RELATIONAL VIEW)?

:
TEST20
TEST20 IS NOT A VALID COMMAND

SUBSYSTEMS: DBA, BV(BASE VIEW) OR RV(RELATIONAL VIEW)?

:

-- END OF USER SESSION--

SAVE FILE: FILE NAME?

:
TEST20 <- (Save as TEST20)
TEST ENDS

R; T=9.86/12.17 22:59:08
spool console class h start
R; T=0.01/0.02 23:00:41
```

APPENDIX 2

MACRO DECLARATIONS

(Control Blocks, Catalogues, Entries)

```
LIBPDS     )
 DCL 1 CRT_ARG BASED(P),                                                F1A00020
       2 LEN FIXED BIN (15) INIT (102),                                 F1A00030
       2 CBTP FIXED BIN (15) INIT(3),                                   F1A00040
       2 PTR PTR INIT (NULL),                                           F1A00050
       2 N FIXED BIN, /*LEN OF DATA IN BIT*/                            F1A00060
       2 DATA BIT(800); /*UP TO 100 BYTES LONG PER DATA UNIT*/          F1A00070
                                                                        F1A00080
 DCL 1 CRT_RTN BASED(P),                                                F1A00090
       2 LEN FIXED BIN (15) INIT ( 6),                                  F1A00100
       2 CBTP FIXED BIN (15) INIT(4),                                   F1A00110
       2 PTR PTR INIT (NULL),                                           F1A00120
       2 RTN_CODE FIXED BIN INIT (0),                                   F1A00130
       2 ID BIT (32);                                                   F1A00140
/  /
 /*DCLS OF ARGUMENTS TO COMMUNICATE WITH THE MODLE DEFA*/               F1A00020
 DCL 1 DEFA_ARG   BASED(P),                                             F1A00030
       2 LEN     FIXED BIN (15) INIT (37),                            F1A00040
       2 CBTP FIXED BIN (15) INIT( 5 ),                               F1A00050
       2 PTR PTR INIT (NULL),                                         F1A00060
       2 ENAME1 CHAR (8) , /*PARENT ENTITY NAME*/                     F1A00070
       2 NAME CHAR (8) INIT ((8)' '),  /*ATTRIBUTE NAME*/             F1A00080
       2 TYPE CHAR (1) INIT (' '), /*'E'OR 'V'*/                      F1A00090
       2 VTYPE CHAR (1) INIT (' '),/*'N'OR 'C'*/                      F1A00100
       2 MLEN FIXED BIN,    /*DEFAULT IS 20*/                         F1A00110
       2 MAX FIXED BIN(31), /*DEFAULT IS 99999*/                      F1A00120
       2 MIN FIXED BIN(31), /*DEFAULT IS -99999*/                     F1A00130
       2 ENAME CHAR (8) INIT ((8)' '),                                F1A00140
       2 FUNC CHAR (1) INIT(' ');/*'S' FOR 1:1,'M' FOR M:1,'K'FOR KEY*/ F1A00150
 DCL 1  DEFA_RTN  BASED(P),                                             F1A00160
       2 LEN     FIXED BIN (15) INIT (2 ),                            F1A00170
       2 CBTP FIXED BIN (15) INIT( 6 ),                               F1A00180
       2 PTR PTR INIT (NULL),                                         F1A00190
       2 RTN_CODE FIXED BIN (15) INIT(0);                             F1A00200
/  /
 DCL 1 DEFB_ARG   BASED(P),                                             F1A00020
       2 LEN     FIXED BIN (15) INIT (17),                            F1A00030
       2 CBTP FIXED BIN (15) INIT( 7 ),                               F1A00040
       2 PTR PTR INIT (NULL),                                         F1A00050
       2 FUNC CHAR (1), /*'S'=SINGULAR(I.E.,1:1), 'M'=MULTIPLE        F1A00060
                         (I.E.,M:1), 1:N AND N:M NOT AVAILABLE IN     F1A00070
                          CURRENT VERSION*/                           F1A00080
       2 PSETID (2) BIT(32), /*PSETID OF THE ORIGINATING PSET AND THE F1A00090
                              TARGET PSET OF THIS BINARY RELATION*/   F1A00100
       2 IMP,                                                         F1A00110
        3  LINK FIXED BIN (15); /*SEE DEFBLINK FOR DETAIL*/           F1A00120
           /*THE FOLLOWING FIELD IS NOT USED IN CURRENT VERSION  ,    F1A00130
        3 REV_LINK FIXED BIN (15); WHICH DESCRIBES THE IMP OF REV LINK*/ F1A00140
 DCL 1 DEFBLINK,                                                        F1A00150
       2 EMB FIXED BIN INIT(1), /*TARGET ELEM EMBEDDED IN SOURCE ELEM*/ F1A00160
       2 POINTER FIXED BIN INIT (3), /*POINTER CONNECTION*/            F1A00170
         /*THE FOLLOWING ARE NOT USED IN CURRENT VERSION*/             F1A00180
       2 FEMB FIXED BIN INIT (2), /*FREE FORMAT EMBEDDING*/            F1A00190
       2 NONE FIXED BIN INIT (4); /*NOT IMPLEMENTED, RELY ON REVERSE*/ F1A00200
 DCL 1 DEFB_RTN BASED(P),                                               F1A00210
```

```
      2 LEN     FIXED BIN (15) INIT (12),                          F1A00220
      2 CBTP FIXED BIN (15) INIT( 8 ),                             F1A00230
      2 PTR PTR INIT (NULL),                                       F1A00240
      2 RTN_CODE FIXED BIN INIT (0),                               F1A00250
      2 BSETID BIT(32);                                            F1A00260
/ /
 /*DCLS OF ARGUMENTS TO COMMUNICATE WITH THE MODLE DEFE*/          F1A00020
 DCL 1 DEFE_ARG   BASED(P),                                       F1A00030
      2 LEN     FIXED BIN (15) INIT (8),                          F1A00040
      2 CBTP FIXED BIN (15) INIT( 9 ),                             F1A00050
      2 PTR PTR INIT (NULL),                                       F1A00060
      2 NAME CHAR (8) INIT ((8)' ');                               F1A00070
 DCL 1  DEFE_RTN  BASED(P),                                       F1A00080
      2 LEN     FIXED BIN (15) INIT (2 ),                         F1A00090
      2 CBTP FIXED BIN (15) INIT( 10 ),                            F1A00100
      2 PTR PTR INIT (NULL),                                       F1A00110
      2 RTN_CODE FIXED BIN (15) INIT (0);                          F1A00120
/ /
                                                                   F1A00020
 /*ARGUMENT DCLS FOR T-PROC DEFP*/                                 F1A00030
 DCL 1 DEFP_ARG   BASED(P),                                       F1A00040
      2 LEN     FIXED BIN (15) INIT (6),                          F1A00050
      2 CBTP FIXED BIN (15) INIT( 11 ),                            F1A00060
      2 PTR PTR INIT (NULL),                                       F1A00070
      2 PTYPE CHAR (1), /*'C'=CHAR, 'N'=NUM, 'B'=BIT, 'X'=NULL*/   F1A00080
       /*NULL TYPE PSET IS A PSET OF TOKENS WHICH DO NOT CONTAIN    F1A00090
         VALUE DATA. RETRIEVAL OF NULL TYPE PSET RESULTS IN        F1A00100
         A RETURN DATA WITH LENGTH 0.****/                         F1A00110
      2 PLEN FIXED BIN (15), /*LEN OF PRIMITIVE ELEM OF THE PSET,  F1A00120
                        IN BYTES, IF =-1 THEN VARIABLE IN LENGTH*/ F1A00130
      2 IMP,                                                       F1A00140
        3 HOW BIT (8), /*SEE DEFPHOW FOR DETAIL*/                  F1A00150
        3 NUMPTR FIXED BIN (15); /*RELV ONLY IF HOW = SA*/         F1A00160
 DCL 1 DEFPHOW,                                                   F1A00170
      2 SA BIT(8) INIT ('00000001'B),                              F1A00180
      2 EMB BIT (8) INIT ('00000000'B);                            F1A00190
 DCL 1 DEFP_RTN   BASED(P),                                       F1A00200
      2 LEN     FIXED BIN (15) INIT (6),                          F1A00210
      2 CBTP FIXED BIN (15) INIT( 12 ),                            F1A00220
      2 PTR PTR INIT (NULL),                                       F1A00230
      2 RTN_CODE FIXED BIN (15), /*0 IS O.K.*/                     F1A00240
      2 PSETID BIT (32); /*ID OF PSET JUST DEFINED*/               F1A00250
   /
 DCL 1   DEL_ARG  BASED(P),                                       F1A00020
      2 LEN     FIXED BIN (15) INIT (4 ),                         F1A00030
      2 CBTP FIXED BIN (15) INIT( 13 ),                            F1A00040
      2 PTR PTR INIT (NULL),                                       F1A00050
      2 ID BIT(32);                                                F1A00060
 DCL 1   DEL_RTN BASED(P),                                        F1A00070
      2 LEN     FIXED BIN (15) INIT ( 2),                         F1A00080
      2 CBTP FIXED BIN (15) INIT( 14 ),                            F1A00090
      2 PTR PTR INIT (NULL),                                       F1A00100
      2 RTN_CODE FIXED BIN INIT (0);                               F1A00110
/ /
 DCL 1 MINIT_ARG  BASED(P),                                       F1A00020
```

```
      2 LEN FIXED BIN (15) INIT(13),                                F1A00030
      2 CBTP FIXED BIN (15) INIT( 15 ),                             F1A00040
      2 PTR PTR INIT(NULL()),                                       F1A00050
      2 OP CHAR (1), /*'I'=INIT, 'S'=STORE KEY*/                    F1A00060
      2 FNAME CHAR(8), /*NAME OF FILE*/                             F1A00070
      2 KEY BIT(32);                                                F1A00080
 DCL 1  MINIT_RTN BASED(P),                                         F1A00090
      2 LEN FIXED BIN (15) INIT(6),                                 F1A00100
      2 CBTP FIXED BIN (15) INIT( 16 ),                             F1A00110
      2 PTR PTR INIT(NULL()),                                       F1A00120
      2 RTN_CODE FIXED BIN (15) INIT (0), /*0=O.K.*/                F1A00130
      2 KEY BIT(32); /*KEY VALUE RETURNED IN A FILE INIT*/          F1A00140
/  /
 DCL 1 NINIT_ARG  BASED(P),                                         F1A00020
      2 LEN FIXED BIN (15) INIT(13),                                F1A00030
      2 CBTP FIXED BIN (15) INIT( 17 ),                             F1A00040
      2 PTR PTR INIT(NULL()),                                       F1A00050
      2 OP CHAR (1), /*'I'=INIT, 'S'=STORE KEY*/                    F1A00060
      2 FNAME CHAR(8), /*NAME OF FILE*/                             F1A00070
      2 KEY BIT(32);                                                F1A00080
 DCL 1  NINIT_RTN BASED(P),                                         F1A00090
      2 LEN FIXED BIN (15) INIT(6),                                 F1A00100
      2 CBTP FIXED BIN (15) INIT( 18 ),                             F1A00110
      2 PTR PTR INIT(NULL()),                                       F1A00120
      2 RTN_CODE FIXED BIN (15) INIT (0), /*0=O.K.*/                F1A00130
      2 KEY BIT(32); /*KEY VALUE RETURNED IN A FILE INIT*/          F1A00140
/  /
 DCL 1 REP_ARG    BASED(P),                                         F1A00020
      2 LEN     FIXED BIN (15) INIT (106),                          F1A00030
      2 CBTP    FIXED BIN (15) INIT (  1),                          F1A00040
      2 PTR PTR INIT (NULL),                                        F1A00050
      /* ------- */                                                F1A00060
      2 ID BIT (32),                                                F1A00070
      2 N FIXED BIN, /*LEN OF DATA IN BIT*/                         F1A00080
      2 DATA BIT(800); /*UP TO 100 BYTES LONG*/                     F1A00090
                                                                    F1A00100
 DCL 1   REP_RTN BASED(P),                                          F1A00110
      2 LEN     FIXED BIN (15) INIT ( 2),                           F1A00120
      2 CBTP    FIXED BIN (15) INIT ( 2),                           F1A00130
      2 PTR PTR INIT (NULL),                                        F1A00140
      /* ------- */                                                F1A00150
      2 RTN_CODE FIXED BIN INIT (0);                                F1A00160
/  /
 DCL 1 RET_ARG    BASED(P),                                         F1A00020
      2 LEN     FIXED BIN (15) INIT (4),                            F1A00030
      2 CBTP FIXED BIN (15) INIT( 19 ),                             F1A00040
      2 PTR PTR INIT (NULL),                                        F1A00050
      2 ID BIT (32);                                                F1A00060
                                                                    F1A00070
 DCL 1   RET_RTN BASED(P),                                          F1A00080
      2 LEN     FIXED BIN (15) INIT (104),                          F1A00090
      2 CBTP FIXED BIN (15) INIT( 20 ),                             F1A00100
      2 PTR PTR INIT (NULL),                                        F1A00110
      2 RTN_CODE FIXED BIN INIT (0),                                F1A00120
      2 N FIXED BIN, /*LEN OF DATA IN BIT*/                         F1A00130
```

```
      2 DATA BIT(800); /*UP TO 100 BYTES LONG*/                          F1A00140
/  /
 DCL 1 RETE_ARG   BASED(P),                                              F1A00020
      2 LEN     FIXED BIN (15) INIT (61),                                F1A00030
      2 CBTP FIXED BIN (15) INIT( 21 ),                                  F1A00040
      2 PTR PTR INIT (NULL),                                             F1A00050
      2 NODE FIXED BIN (15) INIT(1), /*NODE NUMBER*/                     F1A00060
      2 NAME CHAR (8), /*ANAME OR ENAME*/                                F1A00070
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/            F1A00080
      2 GET BIT(8) INIT ('00000000'B), /*SEE RETEGET FOR DETAIL*/        F1A00090
      2 N FIXED BIN INIT(0), /*NUMBER OF RELEVANT 'OR' PREDICATES*/      F1A00100
      2 PRED (1)/*IN THIS VERSION ONLY ONE PREDICATE EACH TIME IS        F1A00110
          ALLOWED; IN THE FUTURE MAY USE REFER OPTION TO N*/,            F1A00120
        3 CN FIXED BIN (15) INIT(1),/*CONDITION NUMBER*/                 F1A00130
        3 SETOP BIT(8), /*RELEVANT IF BSET IF MULTI-TARGET;              F1A00140
                          NOT USED IN CURRENT VERSION*/                  F1A00150
        3 OP BIT(8), /*COMPARISON OPS, SEE RETEOP FOR DETAIL*/           F1A00160
        3 DLEN FIXED BIN (15) INIT(0), /* LEN OF COMPARE DATA IN BYTES*/  F1A00170
        3 CDATA CHAR (40);/*40 BYTES MAXIMUM*/                           F1A00180
  DCL 1 RETE_ARG1 BASED (P), /*USED FOR 'BELONG' COMP OP*                 F1A00190
                             *NOT USED IN CURRENT VERSION*/              F1A00200
      2 LEN FIXED BIN INIT (8 ), /*PLUS VALUE LEN*/                      F1A00210
      2 CBTP FIXED BIN (15) INIT( 23 ),                                  F1A00220
      2 PTR PTR, /*POINT TO THE NEXT RETE_ARG*/                          F1A00230
      2 N FIXED BIN,  /* # OF ELEMENTS IN THIS IMMED. VALUE SET*/        F1A00240
      2 VALUE (I REFER(RETE_ARG1.N)) CHAR(40); /*MAX OF 40 BYTES*/       F1A00250
  DCL RETEVALUE_LEN FIXED BIN (15) INIT (40);                            F1A00260
  DCL 1 RETEGET,                                                         F1A00270
      2 ANY BIT(8) INIT ('00000001'B),                                   F1A00280
      2 ALL BIT (8) INIT ('00000011'B),                                  F1A00290
      2 NO BIT (8) INIT ('00000000'B); /*NOTE ROOT NODE CANNOT           F1A00300
                                         ASSUME THIS VALUE*/             F1A00310
 DCL 1 RETEOP,                                                           F1A00320
      2 EQ BIT (8) INIT ('00000000'B),                                   F1A00330
      2 GT BIT(8)  INIT ('00000010'B),                                   F1A00340
      2 LT BIT(8) INIT  ('00000001'B),                                   F1A00350
      2 BEL BIT(8) INIT ('00000011'B),                                   F1A00360
      2 NULL BIT (8) INIT ('00000100'B); /*NOT SPECIFIED*/               F1A00370
 DCL 1   RETE_RTN BASED(P),                                              F1A00380
      2 LEN     FIXED BIN (15) INIT (4),                                 F1A00390
      2 CBTP FIXED BIN (15) INIT( 22 ),                                  F1A00400
      2 PTR PTR INIT (NULL),                                             F1A00410
      2 RTN_CODE FIXED BIN INIT (0),                                     F1A00420
      2 N FIXED BIN (15); /*NUMBER OF INSTANCES OF ROOT NODE RETURNED*/  F1A00430
 DCL 1  RETE_RTN1 BASED(P),                                              F1A00440
      2 LEN     FIXED BIN (15) INIT (46),                                F1A00450
      2 CBTP FIXED BIN (15) INIT( 24 ),                                  F1A00460
      2 PTR PTR INIT (NULL),                                             F1A00470
      2 NODE FIXED BIN (15), /*NODE NUMBER*/                             F1A00480
      2 INDX FIXED BIN (15), /*IF MULTI OCCUR*/                          F1A00490
      2 DLEN FIXED BIN, /*LEN OF DATA IN BYTES*/                         F1A00500
      2 CDATA  CHAR(40);  /*DATA RETURNED*/                             F1A00510
/  /
 DCL 1 RETN_ARG   BASED(P),                                              F1A00020
      2 LEN     FIXED BIN (15) INIT (61),                                F1A00030
```

```
      2 CBTP FIXED BIN (15) INIT( 25 ),                                  F1A00040
      2 PTR PTR INIT (NULL),                                             F1A00050
      2 NODE FIXED BIN (15) INIT(1), /*NODE NUMBER*/                     F1A00060
      2 PSETID BIT(32), /*PSETID OF THE NODE*/                           F1A00070
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/            F1A00080
      2 BSETID BIT(32), /*BSETID OF THE PARENT PSET*/                    F1A00090
      2 GET BIT(8) INIT ('00000000'B), /*SEE RETNGET FOR DETAIL*/        F1A00100
      2 N FIXED BIN INIT(0), /*NUMBER OF RELEVANT 'OR' PREDICATES*/      F1A00110
      2 PRED (1),/*IN CURRENT VERSION ONLY ONE PREDICATE EACH TIME       F1A00120
            IS ALLOWED; FUTURE VERSION MAY USE REFER OPTION TO N*/       F1A00130
       3 CN FIXED BIN (15), /*CONDITION NUMBER*/                         F1A00140
       3 SETOP BIT(8), /*RELEVANT IF BSET IF MULTI-TARGET;               F1A00150
                         NOT USED IN CURRENT VERSION*/                  F1A00160
       3 OP BIT(8), /*COMPARISON OPS, SEE RETNOP FOR DETAIL*/            F1A00170
       3 DLEN FIXED BIN (15) INIT(0), /* LEN OF COMPARE DATA IN BYTES*/   F1A00180
       3 DATA BIT(320); /*40 BYTES MAXIMUM*/                            F1A00190
  DCL RETNPRED_LEN FIXED BIN INIT (46);                                  F1A00200
  DCL 1 RETN_ARG1 BASED (P), /*USED FOR 'BELONG' COMP OP*                F1A00210
                             *NOT USED IN CURRENT VERSION*/              F1A00220
       2 ARG_LEN FIXED BIN INIT (8 ), /*PLUS VALUE LEN*/                 F1A00230
       2 PTR PTR, /*POINT TO THE NEXT RETN_ARG*/                        F1A00240
       2 N FIXED BIN,  /* # OF ELEMENTS IN THIS IMMED. VALUE SET*/       F1A00250
       2 VALUE (I REFER(RETN_ARG1.N)) BIT(320); /*MAX OF 40 BYTES*/     F1A00260
  DCL RETNVALUE_LEN FIXED BIN (15) INIT (40);                            F1A00270
  DCL 1 RETNGET,                                                         F1A00280
       2 ANY BIT(8) INIT ('00000001'B),                                  F1A00290
       2 ALL BIT (8) INIT ('00000011'B),                                 F1A00300
       2 NO BIT (8) INIT ('00000000'B); /*NOTE ROOT NODE CANNOT          F1A00310
                                          ASSUME THIS VALUE*/           F1A00320
 DCL 1 RETNOP,                                                           F1A00330
       2 EQ BIT (8) INIT ('00000000'B),                                  F1A00340
       2 GT BIT(8)  INIT ('00000010'B),                                  F1A00350
       2 LT BIT(8) INIT  ('00000001'B),                                  F1A00360
       2 BEL BIT(8) INIT ('00000011'B),                                  F1A00370
       2 NULL BIT (8) INIT ('00000100'B); /*NOT SPECIFIED*/              F1A00380
 DCL 1   RETN_RTN BASED(P),                                              F1A00390
       2 LEN     FIXED BIN (15) INIT (4),                              F1A00400
       2 CBTP FIXED BIN (15) INIT( 26 ),                               F1A00410
       2 PTR PTR INIT (NULL),                                          F1A00420
       2 RTN_CODE FIXED BIN INIT (0),                                  F1A00430
       2 N FIXED BIN (15);/*NUMBER OF INSTANCES OF ROOT NODE RETURNED*/ F1A00440
 DCL 1  RETN_RTN1 BASED(P),                                              F1A00450
       2 LEN     FIXED BIN (15) INIT (46),                             F1A00460
       2 CBTP FIXED BIN (15) INIT( 27 ),                               F1A00470
       2 PTR PTR INIT (NULL),                                          F1A00480
       2 NODE FIXED BIN (15), /*NODE NUMBER*/                          F1A00490
       2 INDX FIXED BIN (15), /*IF MULTI OCCUR*/                       F1A00500
       2 DLEN FIXED BIN, /*LEN OF DATA IN BYTES*/                      F1A00510
       2 DATA  BIT(320);  /*DATA RETURNED*/                           F1A00520
/  /
 DCL 1 SHWE_ARG  BASED(SHWE_ARG_P),                                      F1A00020
      2 CTL,                                                             F1A00030
       3 LEN     FIXED BIN (15) INIT (210),                            F1A00040
       3 CBTP FIXED BIN (15) INIT (36),                                F1A00050
       3 PTR PTR INIT (NULL),                                          F1A00060
```

```
     2 DATA,                                                          F1A00070
      3 NAME1 CHAR (8),                                               F1A00080
      3 NAME2(25) CHAR (8),                                           F1A00090
      3 NUM_NAME2 FIXED BIN;                                          F1A00100
                                                                      F1A00110
 DCL 1  SHWE_RTN BASED(SHWE_RTN_P),                                   F1A00120
     2 CTL,                                                           F1A00130
      3 LEN     FIXED BIN (15) INIT (2),                            F1A00140
      3 CBTP FIXED BIN (15) INIT (37),                              F1A00150
      3 PTR PTR INIT (NULL),                                          F1A00160
     2 DATA,                                                          F1A00170
      3 RTN_CODE FIXED BIN INIT (0);                                  F1A00180
                                                                      F1A00190
 DCL 1 SHWE_RTN1 BASED (SHWE_RTN_P),                                  F1A00200
       2 CTL,                                                         F1A00210
         3 LEN FIXED BIN (15) INIT (29),                              F1A00220
         3 CBTP FIXED BIN (15) INIT (38),                             F1A00230
         3 PTR PTR INIT (NULL),                                       F1A00240
       2 DATA,                                                        F1A00250
         3 ANAME CHAR (8),                                            F1A00260
         3 ENAME CHAR (8),   /* THE FOLLOWING SAME AS AINFO */        F1A00270
         3 MAX FIXED BIN (31),                                        F1A00280
         3 MIN FIXED BIN (31),                                        F1A00290
         3 MLEN FIXED BIN,                                            F1A00300
         3 TYPE CHAR (1),                                             F1A00310
         3 FUNC CHAR (1),                                             F1A00320
         3 VTYPE CHAR (1);                                            F1A00330
   DCL (SHWE_ARG_P, SHWE_RTN_P) PTR;                                  F1A00340
   DCL 1 SHWE_RTN2 BASED (SHWE_RTN_P),                                F1A00350
       2 CTL,                                                         F1A00360
         3 LEN FIXED BIN INIT (202),                                  F1A00370
         3 CBTP FIXED BIN INIT (39),                                  F1A00380
         3 PTR PTR INIT (NULL),                                       F1A00390
       2 DATA,                                                        F1A00400
         3 NUM_ENAME FIXED BIN,                                       F1A00410
         3 ENAME (25) CHAR (8);                                       F1A00420
/  /
 DCL 1 UPDE_ARG   BASED(P),                                           F1A00020
      2 LEN     FIXED BIN (15) INIT (56),                           F1A00030
      2 CBTP FIXED BIN (15) INIT( 28 ),                             F1A00040
      2 PTR PTR INIT (NULL),                                          F1A00050
      2 NODE FIXED BIN (15) INIT(1),/*NODE NUMBER OF UPDE TREE*/      F1A00060
      2 NAME CHAR(8),/*ENAME OR ANAME*/                               F1A00070
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/         F1A00080
      2 OP FIXED BIN (15), /*OP CODE FOR UPDE; SEE UPDE.OP FOR DETAIL*/ F1A00090
      2 DLEN FIXED BIN (15) INIT (0), /*IN NUMBER OF BYTES*/          F1A00100
      2 CDATA CHAR(40); /*MAX 40 BYTES OF IMMED DATA */              F1A00110
      /*THE FOLLOWING NOT USED CURRENTLY                              F1A00120
      2 DLEN2 FIXED BIN (15),                                         F1A00130
      2 DATA2 BIT (320),                                              F1A00140
      2 DTYPE2 CHAR (1);*/                                            F1A00150
 DCL 1 UPDEOP,                                                        F1A00160
        3 ID FIXED BIN INIT(1),                                       F1A00170
        3 IST FIXED BIN INIT(2), /*INSERT ATTR*/                      F1A00180
        3 ADD FIXED BIN INIT(3), /*ADD ATTR*/                         F1A00190
```

```
        3 DEL FIXED BIN INIT(4), /*DEL ATTR*/                          F1A00200
        3 SUB FIXED BIN INIT (5), /*SUBTRACT ATTR*/                    F1A00210
        3 REP FIXED BIN INIT (6), /*REPLACE ATTR*/                     F1A00220
        3 MOD FIXED BIN INIT (12), /*MODIFY ENTITY*/                   F1A00230
        3 CRT FIXED BIN INIT (13), /*CREATE ENTITY*/                   F1A00240
        3 DEE FIXED BIN INIT (11), /*DELETE ENTITY*/                   F1A00250
        3 NULL FIXED BIN INIT (0); /*NOT ASSIGNED BY USER*/             F1A00260
 DCL 1  UPDE_RTN  BASED(P),                                            F1A00270
      2 LEN     FIXED BIN (15) INIT (2),                               F1A00280
      2 CBTP FIXED BIN (15) INIT( 29 ),                                F1A00290
      2 PTR PTR INIT (NULL),                                           F1A00300
      2 RTN_CODE FIXED BIN (15) INIT(0); /*0=O.K.*/                    F1A00310
/  /
 DCL 1 UPDN_ARG   BASED(P),                                            F1A00020
      2 LEN     FIXED BIN (15) INIT (56),                              F1A00030
      2 CBTP FIXED BIN (15) INIT( 30 ),                                F1A00040
      2 PTR PTR INIT (NULL),                                           F1A00050
      2 NODE FIXED BIN (15) INIT(1),/*NODE NUMBER OF UPDN TREE*/       F1A00060
      2 PSETID BIT (32), /*PSETID OF THE NODE*/                        F1A00070
      2 PARENT FIXED BIN (15) INIT(0), /*PARENT NODE NUMBER*/          F1A00080
      2 BSETID BIT (32), /*BSETID OF PARENT PSET LEADING HERE*/        F1A00090
      2 OP FIXED BIN (15), /*OP CODE FOR UPDN; SEE UPDNOP FOR DETAIL*/ F1A00100
      2 DLEN FIXED BIN (15) INIT(0), /*IN NUMBER OF BYTES*/            F1A00110
      2 DATA BIT(320); /*MAX 40 BYTES OF IMMED DATA */                F1A00120
      /*THE FOLLOWING NOT USED CURRENTLY                               F1A00130
      2 DLEN2 FIXED BIN (15),                                          F1A00140
      2 DATA2 BIT (320),                                               F1A00150
      2 DTYPE2 CHAR (1);*/                                             F1A00160
 DCL 1 UPDNOP,                                                         F1A00170
        3 ID FIXED BIN INIT(1),                                        F1A00180
        3 IST FIXED BIN INIT(2), /*INSERT ARC*/                        F1A00190
        3 ADD FIXED BIN INIT(3), /*ADD ARC-FOR MULTI-TARG*/            F1A00200
        3 DEL FIXED BIN INIT(4), /*DEL ARC*/                           F1A00210
        3 SUB FIXED BIN INIT (5), /*SUBTRACT ARC-FOR MULTI-TARG*/       F1A00220
        3 REP FIXED BIN INIT (6), /*REPLACE ARC*/                      F1A00230
        3 CI FIXED BIN INIT (14), /*CREATE TARGET NODE THEN INST ARC*/  F1A00240
        3 DD FIXED BIN INIT (15), /*DELETE ARC AND TARGET NODE TOO*/    F1A00250
        3 REPP FIXED BIN INIT (16), /*REPLACE VALUE OF TARG NODE*/      F1A00260
        3 NULL FIXED BIN INIT (0), /*RESERVED*/                        F1A00270
        3 MOD FIXED BIN INIT (12), /*MODIFY TREE--ROOT NODE OP ONLY*/   F1A00280
        3 CRT FIXED BIN INIT (13), /*CREATE TREE--ROOT NODE OP ONLY*/   F1A00290
        3 DEE FIXED BIN INIT (11); /*DELETE TREE--ROOT NODE OP ONLY*/   F1A00300
 DCL 1  UPDN_RTN  BASED(P),                                            F1A00310
      2 LEN FIXED BIN (15) INIT (2),                                   F1A00320
      2 CBTP FIXED BIN (15) INIT( 31 ),                                F1A00330
      2 PTR PTR INIT (NULL),                                           F1A00340
      2 RTN_CODE FIXED BIN (15) INIT(0); /*0=O.K.*/                    F1A00350
/  /
 DCL 1 VINIT_ARG  BASED(P),                                            F1A00020
      2 LEN     FIXED BIN (15) INIT (13),                              F1A00030
      2 CBTP FIXED BIN (15) INIT( 32 ),                                F1A00040
      2 PTR PTR INIT (NULL),                                           F1A00050
      2 OP CHAR(1), /*NOT USED IN CURRENT VERSION*/                    F1A00060
      2 FNAME CHAR(8),                                                 F1A00070
      2 KEY BIT (32);   /*NOT USED IN CURRENT VERSION*/                F1A00080
```

```
 DCL 1 VINIT_RTN  BASED(P),                                              F1A00090
       2 LEN     FIXED BIN (15) INIT (6 ),                               F1A00100
       2 CBTP FIXED BIN (15) INIT( 33 ),                                 F1A00110
       2 PTR PTR INIT (NULL),                                            F1A00120
       2 RTN_CODE FIXED BIN INIT (0 ),                                   F1A00130
       2 KEY BIT(32);                                                    F1A00140
/  /
 DCL 1 VNME_ARG   BASED(P),                                              F1A00020
       2 LEN     FIXED BIN (15) INIT (260),                              F1A00030
       2 CBTP FIXED BIN (15) INIT( 34 ),                                 F1A00040
       2 PTR PTR INIT (NULL),                                            F1A00050
       2 T,                                                              F1A00060
         3 ENAME CHAR(8),                                                F1A00070
         3 TN FIXED BIN,                                                 F1A00080
         3 T1 (25) ,                                                     F1A00090
          4 PARENT FIXED BIN,                                            F1A00100
          4 ANAME CHAR(8);                                               F1A00110
 DCL 1  VNME_RTN BASED(P),                                               F1A00120
       2 LEN     FIXED BIN (15) INIT (4),                                F1A00130
       2 CBTP FIXED BIN (15) INIT( 35 ),                                 F1A00140
       2 PTR PTR INIT (NULL),                                            F1A00150
       2 RTN_CODE FIXED BIN INIT (0),                                    F1A00160
       2 N FIXED BIN;                                                    F1A00170
/  /
 /*DCL OF BCAT TEMPLATE*/                                                F1D00020
 DCL 1 BINFO,                                                            F1D00030
       2 PSETID(2) BIT (32),                                             F1D00040
       2 POS FIXED BIN, /*POS OF POINTER ARRAY OF PSETID1 USED */        F1D00050
       2 FUNC CHAR (1); /*'S' OR 'M'*/;                                  F1D00060
/  /
 /*DCL OF BASIC ENCODING UNIT (BEU) USED IN THE N-ARY LEVEL***/          F1D00020
 DCL 1 BEU BASED(P),                                                     F1D00030
       2 NUMPTR FIXED BIN,                                               F1D00040
       2 ID_ARRAY(I REFER (BEU.NUMPTR))BIT(32) INIT ((I)UNSPEC(NULL)),   F1D00050
       2 DLEN FIXED BIN,                                                 F1D00060
       2 DATA BIT(J REFER (BEU.DLEN));                                   F1D00070
/  /
 /** CATALOGUE STRUCTURES AT THE ENTITY LEVEL****/                       F1D00020
 DCL EINFO BIT (32); /*HOLDS PSETID OF A ENTITY SET**/                   F1D00030
 DCL 1 AINFO,                                                            F1D00040
       2 BSETID BIT (32),  /*BSETID OF THIS ATTRIBUTE**/                 F1D00050
       2 PSETID BIT (32),  /*PSETID OF TARGET NODE**/                    F1D00060
       2 ENAME CHAR (8), /*IF TARGET IS A ENTITY SET*/                   F1D00070
       2 MAX FIXED BIN (31),                                             F1D00080
       2 MIN FIXED BIN (31),                                             F1D00090
       2 MLEN FIXED BIN,                                                 F1D00100
       2 FUNC CHAR(1),                                                   F1D00110
       2 TYPE CHAR (1),                                                  F1D00120
       2 VTYPE CHAR (1);                                                 F1D00130
/  /
  /*** DCLS OF STATIC KEYS USED AS THE ENTITY LEVEL *****/               F1D00020
  /*** ONLY MODULE VINIT HAS THE WRITE ACCESS: OTHER MODULES R/O**/      F1D00030
  DCL KEY (11) BIT (32) EXT STATIC;                                      F1D00040
  DCL 1 KY /*INDEX OF 11 KEYS*/  STATIC,                                F1D00050
       2 PESET FIXED BIN INIT (1),                                       F1D00060
```

```
      2 PENAME FIXED BIN INIT (2),                                    F1D00070
      2 PEINFO FIXED BIN INIT (3),                                    F1D00080
      2 PASET FIXED BIN INIT (4),                                     F1D00090
      2 PANAME FIXED BIN INIT (5),                                    F1D00100
      2 PAINFO FIXED BIN INIT (6),                                    F1D00110
      2 BENAME FIXED BIN INIT (7),                                    F1D00120
      2 BEINFO FIXED BIN INIT (8),                                    F1D00130
      2 BESET FIXED BIN INIT (9),                                     F1D00140
      2 BANAME FIXED BIN INIT (10),                                   F1D00150
      2 BAINFO  FIXED BIN INIT (11);                                  F1D00160
                                                                      F1D00170
/  /
 /**DCL OF KEYS (BCATID AND PCATID) USED AT THE N_ARY LEVEL. ONLY     F1D00020
  ** NINIT MODULE HAS THE WRITE ACCESS. OTHER MODULES WITHIN N-ARY     F1D00030
  ** LEVEL MAY HAVE READ ACCESS*******************************/      F1D00040
  DCL (PCATID,BCATID) BIT (32) EXTERNAL STATIC;                       F1D00050
/  /
 /*DCLS OF PSET CATALOGUE AT THE N-ARY LEVEL******/                   F1D00020
 DCL 1 PINFO,                                                         F1D00030
       3 NUMPTR FIXED BIN,                                            F1D00040
       3 PLEN /*IN NUMBER OF BYTES*/ FIXED BIN,                       F1D00050
       3  PTYPE CHAR (1), /*'N' OR'X' OR 'B' OR 'C'*/                 F1D00060
       3 LTYPE BIT (8), /*'00000001' IS LINK LIST*/                   F1D00070
       3 L_ID BIT (32), /*ID OF THE FIRST BEU IN THIS SET**/          F1D00080
       3 L_POS FIXED BIN , /*POS IN ID_ARRY USED FOR LINEAR CHAINING  F1D00090
                 OF BEU'S WITHIN THIS SET*/                           F1D00100
       3 L_POS2 FIXED BIN,                                            F1D00110
       3 MAP BIT (32) ; /*UP TO 16 PTRS ARE ALLOWED FOR A BEU*/       F1D00120
   DCL 1 PINFOLTYPE,                                                  F1D00130
       2 LL BIT (8) INIT ('00000001'B);                               F1D00140
/  /
 /*DCL OF SEU (STORAGE ENCODING UNIT) USED AT THE MM LEVEL AND THE    F1D00020
 ** STORAGE HIERARCHY VIRTUAL MEMORY*******************/             F1D00030
 DCL SH AREA (30000) BASED(SH_BASE),                                  F1D00040
     SH_BASE PTR EXT;                                                 F1D00050
 DCL O OFFSET (SH);                                                   F1D00060
 DCL 1 SEU BASED (O),                                                 F1D00070
      2 MOVE_PTR OFFSET(SH) INIT (NULL()) ,                           F1D00080
      2 N FIXED BIN,                                                  F1D00090
      2 INVAL BIT(8) INIT ('00000000'B),                              F1D00100
      2 DATA BIT (I REFER (SEU.N));                                   F1D00110
                                                                      F1D00120
/  /
 DCL 1 FDEBUG,                                                        FDE00020
       2 LEVO BIT (1) INIT ('1'B),                                    FDE00030
       2 LEV1 BIT (1) INIT ('1'B),                                    FDE00040
       2 LEV2 BIT(1) INIT ('1'B),                                     FDE00050
       2 LEV3 BIT(1) INIT ('1'B),                                     FDE00060
       2 LEV4 BIT(1) INIT ('1'B),                                     FDE00070
       2 LEV5 BIT(1) INIT ('1'B);                                     FDE00080
/  /
 /*DCL OF SEU (STORAGE ENCODING UNIT) USED AT THE MM LEVEL AND THE    DSE00020
 ** STORAGE HIERARCHY VIRTUAL MEMORY: FOR INTEGRATED VERSION *****/   DSE00030
 DCL 1 SEU BASED (SEUPTR),                                            DSE00040
  /*  2 MOVE_PTR OFFSET(SH) INIT (NULL()) . NOT USED CURRENTLY */     DSE00050
```

```
      2 N FIXED BIN,                                                    DSE00060
      2 INVAL CHAR(1) INIT ('N'),                                       DSE00070
      2 DATA CHAR(I REFER (SEU.N)),                                     DSE00080
 (SEUPTR) PTR;                                                          DSE00090
/  /
 DCL 1 ARCH(19),                        /* ARCHITECTURAL LEVELS      */  F1F00020
      2 LEVEL FIXED BIN(31) INIT (1,2,2,2,2,                            F1F00030
                                  2,2,2,3,3,                            F1F00040
                                  3,3,3,4,4,                            F1F00050
                                  4,4,4,5),                             F1F00060
      2 PROCNAME CHAR (8) INIT ('USER','VINIT','VNME','DEFE','RETE',    F1F00070
                                'UPDE','DEFA','SHWE','DEFP','DEFB',     F1F00080
                                'RETN','UPDN','NINIT','RET','DEL',      F1F00090
                                'REP','CRT','MINIT','L1');              F1F00100
/  /
 /* SIMULATED STORAGE HIERARCHY, KNOWN TO DUMFSTV, DUML1,FHL1 */        F1S00020
 DCL SH AREA (30000) BASED (SH_BASE);                                   F1S00030
 DCL SH_BASE PTR EXT;                                                   F1S00040
 DCL PACKET(3000) CHAR (8) BASED(PP), PP OFFSET(SH) EXT;                F1S00050
/  /
 /*  THIS IS THE PROC_TBL TO BE INCLUDED IN THE SVCS1 ROUTINE **/       SVC00010
 DCL 1 PROC_TBL (18),                                                   SVC00020
      2 N CHAR(7) INIT ('VINIT','DEFA','DEFE','RETE','UPDE','SHWE',    SVC00030
          'VNME',                                                       SVC00040
          'RETN','UPDN','DEFP','DEFB','NINIT',                         SVC00050
          'RET','REP','CRT','DEL','MINIT','USER'),                     SVC00060
      2 LEVEL FIXED BIN INIT ((7)6,(5)5,(5)0,7);                        SVC00070
   /*NOTE THAT T-PROC'S 1-7 ARE IN THE ENTITY LEVEL (6) AND 8-12       SVC00080
     ARE IN INTERNAL SCHEMA LEVEL (ASSIGNED LEVEL 5) AND T-PROC'S      SVC00090
     13-17 ARE IN THE MEMORY MGMT LEVEL (ASSIGNED LEVEL 0). 'L1'       SVC00100
     IS THE INTERFACE MODULE AT THE SH-STV (ASSIGNED LEVEL 1). */      SVC00110
                                                                       SVC00120
 DCL PROC_TBL_LEN FIXED BIN INIT (18);                                 SVC00130
/  /
ACRT            ADEFA           ADEFB           ADEFE           ADEFP          ¢
ADEL            AMINI          >ANINI          •AREP            ARET
ARETE          yARETN           ASHWE           AUPDE           AUPDN
AVINI          #AVNME          1DBCAT           DBEU            DECAT          ×
DKEY            DNKEY          EDPCAT           DSEU           QFDEBUG         U
DSEU1           FSTRUC         5SH              SVCTB           SVCTB
```

```
LIBPDS
 DCL VTPE ENTRY /*ENTRY DCL FOR MODULE VTPE*/                          F1E00020
       (1,2 BIT(32), 2 BIT(32), 2 CHAR(8),                             F1E00030
           2 FIXED BIN (31), 2 FIXED BIN (31),2  FIXED BIN,            F1E00040
           2 CHAR (1), 2 CHAR(1), 2 CHAR (1), FIXED BIN,               F1E00050
          BIT (320), CHAR(40),FIXED BIN,FIXED BIN);                    F1E00060
/  /
 DCL ALLA ENTRY /*ENTRY DCL FOR MODULE ALLA*/                          F1E00020
    (CHAR (8), (25) CHAR (8), FIXED BIN, FIXED BIN );                  F1E00030
/  /
 DCL GBCT ENTRY /*ENTRY DCL FOR MODULE GBCT WHICH RETRIEVES BINF*/     F1E00020
     (BIT(32), 1,2 (2) BIT(32), 2 FIXED BIN, 2 CHAR(1), FIXED BIN);    F1E00030
/  /
 /*ENTRY DCL OF MODULE BNV*/                                           F1E00020
 DCL BNV ENTRY;                                                        F1E00030
/  /
 DCL CRTB ENTRY /*ENTRY DCL FOR MODULE CRTB*/                          F1E00020
       (BIT(32),BIT(32),BIT(32),BIT(*),CHAR(1),FIXED BIN);             F1E00030
/  /
 DCL CRTN ENTRY  /*ENTRY DCL FOR MODULE CRTN*/                         F1E00020
     (PTR,(25) BIT(1), FIXED BIN);                                     F1E00030
/  /
 DCL CRTP ENTRY /*ENTRY DCL FOR MODULE CRTP*/                          F1E00020
    (BIT(32),BIT(*),BIT (32), FIXED BIN);                              F1E00030
/  /
  /*ENTRY DECL OF DBA MODULE*/                                         F1E00020
  DCL DBA ENTRY;                                                       F1E00030
/  /
 DCL DDB ENTRY; /*ENTRY DCL FOR MODULE DDB*/                           F1E00020
/  /
 DCL DDV ENTRY; /*ENTRY DCL FOR MODULE DDV*/                           F1E00020
/  /
 DCL DELB ENTRY /*ENTRY DCL FOR MODULE DELB*/                          F1E00020
       (BIT(32),BIT(32),BIT(32),CHAR(1),FIXED BIN);                    F1E00030
/  /
 DCL DELN ENTRY  /*ENTRY DCL FOR MODULE DELN*/                         F1E00020
     (PTR,(25)BIT (1), FIXED BIN);                                     F1E00030
/  /
 DCL DELP ENTRY /*ENTRY DCL FOR MODULE DELP*/                          F1E00020
      (BIT(32),BIT(32),FIXED BIN);                                     F1E00030
/  /
 DCL DFB1 ENTRY  /*ENTRY DCL FOR MODULE DFB1*/                         F1E00020
     (CHAR(1), BIT (32), BIT(32), FIXED BIN, BIT(32));                 F1E00030
/  /
 DCL DFP1 ENTRY (CHAR(1), FIXED BIN, CHAR(1), FIXED BIN, BIT(32));     F1E00020
/  /
 DCL DMB ENTRY; /*ENTRY DCL FOR MODULE DMB*/                           F1E00020
/  /
 DCL GACT ENTRY /*ENTRY DCL FOR MODULE GACT*/                          F1E00020
       (CHAR(8),CHAR(8),1,2 BIT(32), 2 BIT(32), 2 CHAR(8),             F1E00030
           2 FIXED BIN (31), 2 FIXED BIN (31),2  FIXED BIN,            F1E00040
           2 CHAR (1), 2 CHAR(1), 2 CHAR (1), FIXED BIN);              F1E00050
/  /
 DCL GECT ENTRY /*ENTRY DCL FOR MODULE GECT*/                          F1E00020
      (CHAR(8), BIT(32), FIXED BIN);                                   F1E00030
```

```
/  /
 DCL LEX ENTRY /*ENTRY DCL FOR MODULE LEX*/                          F1E00020
      (FIXED BIN, 1 (25), 2 CHAR (40), 2 FIXED BIN, FIXED BIN,       F1E00030
         FIXED BIN);                                                 F1E00040
/  /
 DCL LEXT ENTRY /*ENTRY DCL FOR MODULE LEXT*/                        F1E00020
      (FIXED BIN, 1 (25), 2 CHAR (40), 2 FIXED BIN, 1 (25),          F1E00030
          2 FIXED BIN, 2 CHAR (8), FIXED BIN, 1 (25), 2 CHAR (1),    F1E00040
          2 CHAR (40), 2 FIXED BIN, (25) CHAR(1), FIXED BIN);        F1E00050
                                                                     F1E00060
/  /
 DCL MODN ENTRY  /*ENTRY DCL FOR MODULE MODN*/                       F1E00020
     (PTR,(25) BIT(1), FIXED BIN);                                   F1E00030
/  /
 DCL MPST ENTRY /*ENTRY DCL FOR MODULE MPST*/                        F1E00020
   (BIT(32),FIXED BIN);                                              F1E00030
/  /
 /*ENTRY DCL FOR PCAT SERVICE SUBROUTINES*/                          F1E00020
 DCL GPCT ENTRY /*ENTRY DCL FOR MODULE GPCT:GET PCAT*/               F1E00030
   (BIT(32),1,2 FIXED BIN, 2 FIXED BIN,2 CHAR(1), 2 BIT (8),         F1E00040
    2 BIT (32), 2 FIXED BIN, 2 FIXED BIN, 2 BIT (32), FIXED BIN);   F1E00050
 DCL UPCT ENTRY /*ENTRY DCL FOR MODULE UPCT:UPDATE PCAT*/            F1E00060
   (BIT(32),1,2 FIXED BIN, 2 FIXED BIN,2 CHAR(1), 2 BIT (8),         F1E00070
    2 BIT (32), 2 FIXED BIN, 2 FIXED BIN, 2 BIT (32), FIXED BIN);   F1E00080
/  /
 DCL PRNT ENTRY /*ENTRY DCL FOR MODULE PRNT*/                        F1E00020
    (PTR,PTR,FIXED BIN);                                             F1E00030
/  /
 /*ENTRY DCL OF MODULE RELV*/                                        F1E00020
 DCL RELV ENTRY;                                                     F1E00030
/  /
 DCL REPB ENTRY /*ENTRY DCL FOR MODULE REPB*/                        F1E00020
      (BIT(32),BIT(32),BIT(32),BIT(*),CHAR(1),FIXED BIN);            F1E00030
/  /
 DCL SRCH ENTRY /*ENTRY DCL FOR MODULE SRCH*/                        F1E00020
      (BIT(32),BIT(*),BIT (32),FIXED BIN);                           F1E00030
/  /
 DCL RET1 ENTRY /*ENTRY DCL FOR N-ARY LOW LEVEL SUBROUTINE RET1*/    F1E00020
    (BIT(32),PTR,FIXED BIN);                                         F1E00030
 DCL DEL1 ENTRY /*ENTRY DCL FOR N-ARY LOW LEVEL SUBROUTINE DEL1*/    F1E00040
     (BIT(32), FIXED BIN);                                           F1E00050
 DCL REP1 ENTRY /*ENTRY DCL FOR N-ARY LOW LEVEL SUBROUTINE REP1*/    F1E00060
     (BIT(32),PTR, FIXED BIN);                                       F1E00070
 DCL CRT1 ENTRY /*ENTRY DCL FOR N-ARY LOW LEVEL SUBROUTINE CRT1*/    F1E00080
     (BIT (32), PTR, FIXED BIN);                                     F1E00090
/  /
 DCL UPE1 ENTRY /*ENTRY DCL FOR MODULE UPE1*/                        F1E00020
      (CHAR(1),1,2 CHAR(8),2 FIXED BIN,2 (25),3 FIXED BIN,3 CHAR(8), F1E00030
         (25) CHAR(1), FIXED BIN);                                   F1E00040
                                                                     F1E00050
/  /
 /*DCL OF ENTRY FOR USER */                                          EUS00020
 DCL USER  ENTRY;                                                    EUS00030
/  /
 DCL VNM1 ENTRY /*ENTRY DCL FOR MODULE VNM1*/                        F1E00020
```

```
      (1,2 CHAR(8),2 FIXED BIN,2 (25),3 FIXED BIN,3 CHAR(8),          F1E00030
          FIXED BIN, FIXED BIN);                                      F1E00040
                                                                      F1E00050
/  /
 /*SERVICE ROUTINES DCLS****/                                        SER00010
 DCL TCALL ENTRY (CHAR(7), FIXED BIN, PTR, PTR);                     SER00020
 DCL TRTN ENTRY (1, 2 FIXED BIN, 2 FIXED BIN, 2 FIXED BIN, PTR);     SER00030
 DCL TBEG ENTRY (1, 2 FIXED BIN, 2 FIXED BIN, 2 FIXED BIN, PTR);     SER00040
 DCL SVC3 ENTRY (PTR);                                               SER00050
 DCL PROC_NAME CHAR(7);                                              SER00060
 DCL 1 PROC_ADDR,                                                    SER00070
      (2 LEVEL,                                                      SER00080
       2 VPID,                                                       SER00090
       2 BOXID) FIXED BIN;                                           SER00100
                                                                     SER00110
/  /
 /*DCLS OF SVC SEND SERVICES*/                                       F1E00020
 DCL SVCS1 ENTRY (CHAR(7),PTR,FIXED BIN);                            F1E00030
 DCL SVCS2 ENTRY (1,2 FIXED BIN,2 FIXED BIN,2 FIXED BIN,PTR,FIXED BIN); F1E00040
 DCL 1 MSG BASED (P),                                                F1E00050
      2 LEN FIXED BIN (15) INIT(12),                                 F1E00060
      2 DUM_CBTP FIXED BIN (15) INIT (0),                            F1E00070
      2 PTR PTR, /*POINT TO ARG LIST*/                               F1E00080
      2 RTN_ADDR,   /*ADDRESS OF THE PROCESS WHICH INITIATES THE MSG*/ F1E00090
       (3 LEVEL,                                                     F1E00100
        3 VPID,                                                      F1E00110
        3 BOXID) FIXED BIN (15);                                     F1E00120
 DCL PROC_NAME CHAR(7); /*USED IF SVCS1 IS CALLED*/                  F1E00130
 DCL 1 PROC_ADDR, /*USED IF SVCS2 IS CALLED*/                        F1E00140
      (2 LEVEL,                                                      F1E00150
       2 VPID,                                                       F1E00160
       2 BOXID) FIXED BIN (15);                                      F1E00170
/  /
 DCL HEX4 EXT ENTRY (PTR) RETURNS(CHAR(8));                          HEX00010
 /* ROUTINE TO CONVERT A PTR TO HEX STRING */                        HEX00020
 DCL HEX EXT ENTRY (PTR) RETURNS(CHAR(6));                           HEX00030
/  /
 DCL VERIF ENTRY /*ENTRY DCL FOR MODULE VERIF*/                      F1E00020
     ( 1, 2 FIXED BIN, 2 FIXED BIN, 2 CHAR (1), 2 BIT (8),           F1E00030
       2 BIT (32), 2 FIXED BIN, 2 FIXED BIN, 2 BIT (32),             F1E00040
      PTR, BIT(8) ALIGNED,                                           F1E00050
 BIT(*) VAR ALIGNED,                                                 F1E00060
 FIXED BIN, BIT(1) ALIGNED);                                         F1E00070
/  /
 DCL RETB ENTRY /*ENTRY DCL FOR MODULE RETB*/                        F1E00020
   (BIT(32) ALIGNED,1,2 (2) BIT (32),2 FIXED BIN,2 CHAR (1), PTR, PTR, F1E00030
    FIXED BIN);                                                      F1E00040
/  /
 DCL RETP ENTRY /*ENTRY DCL FOR MODULE RETP*/                        F1E00020
  (1,2 FIXED BIN,2 FIXED BIN,2 CHAR (1), 2 BIT (8), 2 BIT (32),      F1E00030
   2 FIXED BIN, 2 FIXED BIN, 2 BIT (32),                             F1E00040
   PTR, PTR, FIXED BIN);                                             F1E00050
/  /
 DCL BUILD ENTRY /*ENTRY DCL FOR MODULE BUILD*/                      F1E00020
   ((25) PTR, (25) BIT(32) ALIGNED,                                  F1E00030
```

```
 PTR, FIXED BIN, PTR, FIXED BIN);                                          F1E00040
/  /
 DCL PAKT ENTRY /*ENTRY DCL FOR MODULE PAKT*/                              F1E00020
     (FIXED BIN(31),CHAR(8),CHAR(1));                                      F1E00030
/  /
 DCL GMEM ENTRY /*ENTRY DCL FOR MODULE GMEM*/                              F1E00020
     (FIXED BIN,BIT(32));                                                  F1E00030
/  /
 DCL GSEU ENTRY              /* ENTRY DCL FOR GSEU SUBROUTINE */          F1E00020
     (BIT(32),PTR,FIXED BIN);                                              F1E00030
/  /
 DCL PSEU ENTRY              /* ENTRY DCL FOR PSEU SUBROUTINE */          F1E00020
     (BIT(32),PTR,FIXED BIN);                                              F1E00030
/  /
EVTPE            EALLA            EBCT             EBNV             ECRTB
ECRTN            ECRTP            EDBA             EDDB             EDDV
EDELB            EDELN            EDELP            EDFB1            EDFP1
EDMB             EGACT            EGECT            ELEX             ELEXT
EMODN            EMPST            EPCT             EPRNT            ERELV
EREPB            ESRCH           !ESUB1           )EUPE1            EUSER
EVNM1           >SERVICE          ESVCS           *HEX             %EVERI
ERETB           )ERETP            EBUIL           uEPAKT           yEGMEM
EGSEU            EPSEU            EPSEU            EPSEU            EPSEU
```

APPENDIX 3

PROGRAM LISTING -- FSTV

```
ALLA: PROC (ENAME,ANAMES,NUM_ANAMES,RTN_CODE);                          ALL00010
                                                                        ALL00020
/* A-PROC AT THE ENTITY LEVEL, INVOKED IN RESPONSE TO A CALL TO         ALL00030
RETRIEVE ALL ATTRIBUTE NAMES OF AN ENTITY SET WHOSE NAMES IS GIVEN      ALL00040
AS ENAME, AND THESE ATTRIBUTE NAMES ARE TO BE RETURNED IN THE           ALL00050
ARRAY ANAMES */                                                         ALL00060
                                                                        ALL00070
%INCLUDE SERVICE, DECAT, ARETE;                                         ALL00080
DCL (P,LP,TP,PP,RP) PTR, NULL BUILTIN, (RTN_CODE,I) FIXED BIN;          ALL00090
DCL ENAME CHAR (8), ANAMES (25) CHAR (8), NUM_ANAMES FIXED BIN;         ALL00100
                                                                        ALL00110
/* BEGIN */                                                             ALL00120
     ALLOCATE RETE_ARG; RP,LP=P;                                        ALL00130
     RETE_ARG.NAME='E*ASET';                                            ALL00140
     RETE_ARG.GET=RETEGET.ALL;                                          ALL00150
                                                                        ALL00160
     DO I = 2 TO 4; /* NODES 2 TO 4 */                                  ALL00170
        ALLOCATE RETE_ARG; LP->RETE_ARG.PTR=P; LP=P;                    ALL00180
        RETE_ARG.NODE=I;                                                ALL00190
        SELECT (I);                                                     ALL00200
          WHEN (4) DO;                                                  ALL00210
             RETE_ARG.NAME='A*ANAME';    /* ANAME NODE */               ALL00220
             RETE_ARG.GET=RETEGET.ALL;                                  ALL00230
             RETE_ARG.PARENT=1;                                         ALL00240
             END /* WHEN 4 */;                                         ALL00250
                                                                        ALL00260
          WHEN (2) DO;                                                  ALL00270
             RETE_ARG.NAME='A*ESET';                                    ALL00280
             RETE_ARG.PARENT=1;                                         ALL00290
             RETE_ARG.GET=RETEGET.NO;                                   ALL00300
             END /* WHEN 2 */;                                         ALL00310
                                                                        ALL00320
          WHEN (3) DO;                                                  ALL00330
             RETE_ARG.NAME='A*ENAME'; /*ENAME PRED NODE */              ALL00340
             RETE_ARG.GET=RETEGET.NO;                                   ALL00350
             RETE_ARG.PARENT=2;                                         ALL00360
             RETE_ARG.N=1; /* 1 PRED */                                 ALL00370
             RETE_ARG.OP(1)=RETEOP.EQ;                                  ALL00380
             RETE_ARG.DLEN(1)=8;                                        ALL00390
             RETE_ARG.CDATA=ENAME;                                      ALL00400
             END /*WHEN 3 */;                                          ALL00410
                                                                        ALL00420
          END /* SELECT */;                                            ALL00430
        END /* DO I */;                                                ALL00440
                                                                        ALL00450
     CALL TCALL ('RETE',1,RP,TP);                                       ALL00460
     CALL SVC3 (RP);                                                    ALL00470
                                                                        ALL00480
     /* EXAMINE RETURN */                                              ALL00490
     P=TP;                                                              ALL00500
     IF RETE_RTN.RTN_CODE^=0 THEN DO;                                   ALL00510
        /***** RTN_CODE 0+ IS FROM RETE CALL ******/                   ALL00520
        CALL SVC3 (P);   GO TO RTN; END;                                ALL00530
                                                                        ALL00540
     NUM_ANAMES = RETE_RTN.N;                                           ALL00550
```

```
      IF NUM_ANAMES=0 THEN DO; /* NO ATTRIBUTES DEFINED */                ALL00560
         CALL SVC3 (P); GO TO RTN; END;                                   ALL00570
                                                                          ALL00580
      /* COPY ATTRIBUTE NAMES FOUND FOR NORMAL PROCESSING */              ALL00590
      DO I = 1 TO NUM_ANAMES;                                             ALL00600
         DO P=RETE_RTN.PTR REPEAT RETE_RTN1.PTR WHILE (P^=NULL);          ALL00610
            IF RETE_RTN1.NODE = 4 /* NODE 2 IS ANAME NODE */             ALL00620
               THEN DO; ANAMES(I)=RETE_RTN1.CDATA;                        ALL00630
                        GO TO NEXT; /* GET OUT OF P LOOP */               ALL00640
                        END;                                              ALL00650
         END /* P */;                                                     ALL00660
      NEXT: END /* DO I */;                                              ALL00670
      CALL SVC3 (TP);                                                     ALL00680
      RTN: END /* ALLA */;                                               ALL00690
```

```
 BUILD: PROC  /* SUBROUTINE USED BY RETN TO BUILD RETURN DATA*/          BUI00010
   (CURSOR,PSETID,MP,K,LP,RTN_CODE);                                    BUI00020
                                                                        BUI00030
  /*MP POINTS TO THE FIRST TOKEN IN RETN TREE, K IS THE OCCUR           BUI00040
  NUMBER OF ROOT NODE, LP POINTS TO THE LAST TOKEN IN THE               BUI00050
  RETN_RTN LIST UP TO NOW***/                                           BUI00060
                                                                        BUI00070
DCL CURSOR (25) PTR CONNECTED; /*CURRENT PTR TO NODES */                BUI00080
DCL PSETID(25) BIT(32) ALIGNED CONNECTED;/*PSETID OF RETN TREE NDS*/    BUI00090
DCL (P,MP,RTP,TP,RP,LP) PTR,(NULL,UNSPEC) BUILTIN, TP1 PTR;             BUI00100
DCL (RNODE,RTN_CODE) FIXED BIN;                                         BUI00110
DCL (K,NODE) FIXED BIN;                                                 BUI00120
                                                                        BUI00130
   %INCLUDE DBEU,ARETN,ESUB1;                                           BUI00140
                                                                        BUI00150
/*BEGIN PROCESSING*/                                                    BUI00160
 RNODE=MP->RETN_ARG.NODE;/*RNODE CONTAINS ROOT NODE NUMBER*/            BUI00170
 P=MP;                                                                  BUI00180
                                                                        BUI00190
/*ALLOCATE ROOT NODE*/                                                  BUI00200
   ALLOCATE RETN_RTN1 SET (TP);                                         BUI00210
   TP->RETN_RTN1.NODE=RETN_ARG.NODE;                                    BUI00220
   TP->RETN_RTN1.INDX=K;                                                BUI00230
   TP1 = CURSOR (RNODE);                                                BUI00240
    IF TP1=NULL THEN DO;                                                BUI00250
       RTN_CODE=1; /*BAD ID IN CURSOR TREE*********/                    BUI00260
       FREE TP->RETN_RTN1;                                              BUI00270
       RETURN;                                                          BUI00280
       END;                                                             BUI00290
                                                                        BUI00300
   TP->RETN_RTN1.DLEN=TP1->BEU.DLEN/8;                                  BUI00310
   TP->RETN_RTN1.DATA=TP1->BEU.DATA;                                    BUI00320
                                                                        BUI00330
        LP->RETN_RTN1.PTR=TP;                                           BUI00340
        LP=TP;                                                          BUI00350
                                                                        BUI00360
                                                                        BUI00370
  /*ALLOCATE THE REST*/                                                BUI00380
  DO P=RETN_ARG.PTR REPEAT RETN_ARG.PTR  WHILE (P^=NULL);               BUI00390
     IF RETN_ARG.GET^=RETNGET.NO & CURSOR(RETN_ARG.NODE) ^=             BUI00400
                NULL THEN DO; /* DO ONLY IF NECESSARY */               BUI00410
        ALLOCATE RETN_RTN1 SET (TP);                                    BUI00420
        LP->RETN_RTN1.PTR=TP;                                           BUI00430
        LP=TP;                                                          BUI00440
        TP->RETN_RTN1.NODE=RETN_ARG.NODE;                               BUI00450
        TP->RETN_RTN1.INDX=1;                                           BUI00460
        TP1 = CURSOR(RETN_ARG.NODE);                                    BUI00470
          IF TP1= NULL THEN DO;                                         BUI00480
          RTN_CODE=1; /****1 IS ILLEGAL CURSOR*******/                  BUI00490
          RETURN;                                                       BUI00500
          END;                                                          BUI00510
        TP->RETN_RTN1.DLEN=TP1->BEU.DLEN/8;                             BUI00520
        TP->RETN_RTN1.DATA=TP1->BEU.DATA;                               BUI00530
        END /*IF GET*/;                                                 BUI00540
     END /*DO P=...*/;                                                  BUI00550
```

```
END /*BUILD*/;                                          BUI00560
```

```
CRT:PROC;                                                              CRT00010
                                                                       CRT00020
/* A PROCEDURE AT THE MEMORY MGMT LEVEL TO CREATE A SEU             */ CRT00030
/* FOR INTEGRATED VERSION: MAKE USE OF SUBROUTINES GMEM (GET VIRTUAL   CRT00040
   MEMORY) AND PSEU (PUT SEU).                                       */ CRT00050
                                                                       CRT00060
  %INCLUDE DSEU1,ACRT,SERVICE,FDEBUG,EPSEU,EGMEM;                      CRT00070
  DCL P PTR, (NULL,UNSPEC) BUILTIN;                                    CRT00080
  DCL (I,RTN_CODE) FIXED BIN, (ID,ID1) BIT(32);                        CRT00090
                                                                       CRT00100
 CALL TBEG(PROC_ADDR,P);                                               CRT00110
 I=CRT_ARG.N/8;                  %INCLUDE HEX;                         CRT00120
 ALLOCATE SEU;                                                         CRT00130
 UNSPEC(SEU.DATA) = CRT_ARG.DATA;                                      CRT00140
                                                                       CRT00150
 CALL GMEM (I+5,ID);          /* 5: 3 FOR INVAL & N, 2 FOR SAFETY */   CRT00160
 ID1=ID;                                                               CRT00170
 CALL PSEU (ID1,SEUPTR,RTN_CODE);     /* STORE IT */                   CRT00180
                                                                       CRT00190
 /*RETURNS*/                                                           CRT00200
 ALLOCATE CRT_RTN;  CRT_RTN.RTN_CODE=0;                                CRT00210
 CRT_RTN.ID=ID;                                                        CRT00220
 CALL TRTN (PROC_ADDR,P);                                              CRT00230
 RETURN;                                                               CRT00240
 END/*CRT*/;                                                           CRT00250
```

```
CRTB: PROC (PARENT,BSETID, ID1,DATA,MODE,RTN_CODE);                    CRT00010
                                                                       CRT00020
%INCLUDE DBCAT, DBEU;                                                  CRT00030
%INCLUDE EBCT,ESRCH,ECRTP,ESUB1;                                       CRT00040
                                                                       CRT00050
DCL (PARENT, BSETID, ID1,ID2) BIT (32), DATA BIT (*);                  CRT00060
DCL RTN_CODE FIXED BIN, MODE CHAR (1), P PTR, NULL BUILTIN;            CRT00070
                                                                       CRT00080
CALL GBCT (BSETID, BINFO, RTN_CODE);                                   CRT00090
                                                                       CRT00100
SELECT (MODE);                                                         CRT00110
                                                                       CRT00120
  WHEN ('N') DO;/* NODE MODE OF CREATION OF BINARY ASSOC*/            CRT00130
                                                                       CRT00140
    IF BINFO.FUNC='S' THEN /*CREATE IT*/                               CRT00150
        CALL CRTP (BINFO.PSETID(2), DATA, ID2,RTN_CODE);               CRT00160
    ELSE DO; /*SEE IF IT ALREADY EXISTS*/                              CRT00170
        CALL SRCH(BINFO.PSETID(2),DATA,ID2,RTN_CODE);                  CRT00180
        IF RTN_CODE=1 /*NOT FOUND*/                                    CRT00190
           THEN CALL CRTP(BINFO.PSETID(2),DATA,ID2,RTN_CODE);          CRT00200
       END;                                                            CRT00210
  END /*WHEN N*/;                                                      CRT00220
                                                                       CRT00230
  WHEN ('A') DO; /*ARC MODE OF CREATION*/                              CRT00240
                                                                       CRT00250
     CALL SRCH (BINFO.PSETID(2), DATA,ID2, RTN_CODE);                  CRT00260
     IF RTN_CODE=1 THEN RETURN;/*RTNCODE 1 IS TARGET OF ARC NOT FND*/  CRT00270
     END /*WHEN 'A'*/;                                                 CRT00280
                                                                       CRT00290
 END /*SLECT*/;                                                        CRT00300
                                                                       CRT00310
 /*ENTER ID2 INTO ID1 ID ARRAY*/                                       CRT00320
 CALL RET1 (ID1,P,RTN_CODE);                                           CRT00330
 BEU.ID_ARRAY(BINFO.POS)=ID2;                                          CRT00340
 CALL REP1 (ID1,P,RTN_CODE);                                           CRT00350
 RTN_CODE=0;                                                           CRT00360
 END /*CRTB*/;                                                         CRT00370
```

```
CRTN: PROC(MP,OFF_NODE,RTN_CODE);                                       CRT00010
                                                                        CRT00020
/*A S-PROC AT THE N-ARY LEVEL WHICH IS CALLED AS A SUBROUTINE BY THE    CRT00030
* UPDN MODULE WHEN ROOT NODE OP IS CRT*/                                CRT00040
                                                                        CRT00050
%INCLUDE AUPDN;                                                         CRT00060
%INCLUDE ECRTB,ECRTP;                                                   CRT00070
                                                                        CRT00080
DCL (MP,P)PTR,OFF_NODE(25) BIT (1) CONNECTED, (RTN_CODE,N) FIXED BIN;   CRT00090
DCL DATA BIT (320) VARYING, (SUBSTR,NULL) BUILTIN,                      CRT00100
    (PS,BS,ID) BIT (32), MODE CHAR (1);                                 CRT00110
                                                                        CRT00120
/*CREATE ROOT NODE*/                                                    CRT00130
  P=MP;                                                                 CRT00140
  N=UPDN_ARG.DLEN;                                                      CRT00150
  DATA=SUBSTR(UPDN_ARG.DATA,1,N*8);                                     CRT00160
  PS=UPDN_ARG.PSETID;                                                   CRT00170
  CALL CRTP(PS,DATA,ID,RTN_CODE);                                       CRT00180
  IF RTN_CODE^=0 THEN RETURN; /*0+ IS RTN CODE FROM CRTP*******/        CRT00190
                                                                        CRT00200
  /*FOLLOW THE TREE*/                                                   CRT00210
  DO P=UPDN_ARG.PTR REPEAT UPDN_ARG.PTR WHILE (P^=NULL);                CRT00220
   IF OFF_NODE(UPDN_ARG.NODE)='0'B THEN DO;                             CRT00230
     BS=UPDN_ARG.BSETID;                                                CRT00240
     DATA=SUBSTR(UPDN_ARG.DATA,1,UPDN_ARG.DLEN*8);                      CRT00250
     SELECT (UPDN_ARG.OP);                                              CRT00260
                                                                        CRT00270
         WHEN (UPDNOP.IST) MODE='A';                                    CRT00280
         WHEN (UPDNOP.CI) MODE='N';                                     CRT00290
         OTHERWISE DO;                                                  CRT00300
                RTN_CODE=10;/*10 IS ILLEGAL OP**************/          CRT00310
                RETURN; END;                                            CRT00320
       END /*SELECT*/;                                                  CRT00330
    CALL CRTB(PS,BS,ID,DATA,MODE,RTN_CODE);                             CRT00340
    END /*DO IF*/ ;                                                     CRT00350
   END /*DO WHILE*/;                                                    CRT00360
  IF RTN_CODE^=0 THEN                                                   CRT00370
 RTN_CODE=RTN_CODE+20; /*20+ IS FROM SUBROUTINE CALL*********/          CRT00380
END /*CRTN*/;                                                          CRT00390
```

```
CRTP: PROC(PSETID,DATA,ID,RTN_CODE);                                    CRT00010
                                                                        CRT00020
/*A S-PROC AT THE N-ARY LEVEL TO CREATE A PRIMITIVE ELEMENT WITHIN      CRT00030
* A PSET DESGINATED BY PSETID****/                                      CRT00040
                                                                        CRT00050
%INCLUDE DPCAT,DBEU;                                                    CRT00060
%INCLUDE EPCT,ESUB1;                                                    CRT00070
DCL (PSETID,ID,ID1) BIT (32), (P,P1) PTR, RTN_CODE FIXED BIN;           CRT00080
DCL (I,J) FIXED BIN, (L_POS,L_POS2) FIXED BIN;                          CRT00090
DCL (NULL,UNSPEC) BUILTIN;                                              CRT00100
DCL DATA BIT (*) ;                                                      CRT00110
                                                                        CRT00120
CALL GPCT (PSETID,PINFO,RTN_CODE); /*OBTAIN PINFO OF PSETID*/           CRT00130
ID1=PINFO.L_ID; /*ID1 POINTS TO THE FIRST ELEMENT */                    CRT00140
L_POS=PINFO.L_POS;                                                      CRT00150
L_POS2=PINFO.L_POS2;                                                    CRT00160
I=PINFO.NUMPTR;                                                         CRT00170
J=PINFO.PLEN*8; /*NO VARIABLE PLEN AVAILABLE IN THIS VERSION*/          CRT00180
ALLOCATE BEU;                                                           CRT00190
                                                                        CRT00200
BEU.ID_ARRAY(L_POS)=ID1;                                                CRT00210
BEU.ID_ARRAY(L_POS2)=UNSPEC(NULL); /*BI-DIRECTIONAL CHAINING*/          CRT00220
BEU.DATA=DATA;                                                          CRT00230
                                                                        CRT00240
CALL CRT1 (ID,P,RTN_CODE);                                              CRT00250
IF RTN_CODE^=0 THEN DO; /*1+ IS PROBLEM IN MM LEVEL*/                   CRT00260
   RTN_CODE=1+RTN_CODE;                                                 CRT00270
   RETURN;                                                              CRT00280
   END;                                                                 CRT00290
                                                                        CRT00300
/*FIX UP 'X' TYPE PTYPE*/                                               CRT00310
IF PINFO.PTYPE='X' THEN DO;                                             CRT00320
  BEU.DATA=ID;                                                          CRT00330
  CALL REP1(ID,P,RTN_CODE);                                             CRT00340
  END;                                                                  CRT00350
                                                                        CRT00360
FREE BEU;                                                               CRT00370
                                                                        CRT00380
/*UPDATE PCAT ENTRY*/                                                   CRT00390
PINFO.L_ID=ID;                                                          CRT00400
CALL UPCT (PSETID,PINFO,RTN_CODE);                                      CRT00410
                                                                        CRT00420
IF ID1^=UNSPEC (NULL) THEN DO; /*IF ORIGINAL SET NOT EMPTY*/            CRT00430
   CALL RET1(ID1,P1,RTN_CODE);                                          CRT00440
   IF P1=NULL THEN DO;                                                  CRT00450
     RTN_CODE=20+RTN_CODE; /*20+ PROBLEM IN REPLACING CHAININIG*/       CRT00460
     RETURN;                                                            CRT00470
     END;                                                               CRT00480
                                                                        CRT00490
/*UPDATE BACK POINTER*/                                                 CRT00500
   P1->BEU.ID_ARRAY(L_POS2)=ID;                                         CRT00510
   CALL REP1(ID1,P1,RTN_CODE);                                          CRT00520
   IF RTN_CODE^=0 THEN RTN_CODE=10+RTN_CODE;                            CRT00530
   FREE P1->BEU;                                                        CRT00540
   END /*IF ID1 NOT NULL*/;                                             CRT00550
```

```
                                                                    CRT00560
                                                                    CRT00570
    RETURN;                                                         CRT00580
    END /*CRTP*/;
```

```
CRT1:PROC (ID,P,RTN_CODE);                                              CRT00010
                                                                        CRT00020
 /*A S-PROC AT THE N-ARY LEVEL FOR COMMUNICATING DIRECTLY WITH THE      CRT00030
 ** CRT MODULE AT THE MM LEVEL*/                                        CRT00040
                                                                        CRT00050
%INCLUDE ACRT, DBEU, SERVICE;                                           CRT00060
DCL ID BIT (32), (P,P1,BP) PTR,(RTN_CODE,I,J,K) FIXED BIN;              CRT00070
DCL (NULL,UNSPEC,STRING,SUBSTR) BUILTIN;                                CRT00080
DCL BIT_STRING BIT(800) BASED;                                          CRT00090
                                                                        CRT00100
                                                                        CRT00110
/*FILL UP CRT_ARG*/                                                     CRT00120
 ALLOCATE CRT_ARG SET (P1);                                             CRT00130
 P1->CRT_ARG.N=32+NUMPTR*32+DLEN;                                       CRT00140
 P1->CRT_ARG.DATA=P->BIT_STRING;                                        CRT00150
/*CALL UP CRT*/                                                         CRT00160
                                                                        CRT00170
CALL TCALL ('CRT',1,P1,BP);                                             CRT00180
RTN_CODE=BP->CRT_RTN.RTN_CODE;                                          CRT00190
IF RTN_CODE=0 THEN ID=BP->CRT_RTN.ID;                                   CRT00200
CALL SVC3(P1); CALL SVC3(BP);                                           CRT00210
                                                                        CRT00220
/*************MEMORY REQUEST TRACE DUMP ***********************/       CRT00230
IF FFF THEN CALL DUMPIT('CREATED',ID,P);DCL FFF BIT(1) EXT;  /**/       CRT00240
/*************************************************************/       CRT00250
                                                                        CRT00260
RETURN;                                                                 CRT00270
END;                                                                    CRT00280
```

```
DBA:PROC;                                                                DBA00010
/*********************************************************************** DBA00020
*                                                                       * DBA00030
*                    MODULE    DESCRIPTION                              * DBA00040
***********************************************************************/ DBA00050
/*********************************************************************** DBA00060
*****   PURPOSE:   FOR DBA TO SELECT AMONG THE FOLLOWING DBA             DBA00070
*****          SUBSYSTEMS: DDB (FOR DEFINING BASE DATA), DDV (FOR        DBA00080
*****          DEFINING VIEW DATA) AND DMB (FOR MANIPULATING             DBA00090
*****          BASE DATA).                                               DBA00100
*****          PRESENTLY THE DDV SUBSYTEM IS NOT AVAILABLE.              DBA00110
*********************************************************************** DBA00120
*****   CALLS PROCEDURES:                                                DBA00130
*****    INTER-LEVEL T-PROC: NONE;                                       DBA00140
*****    INTRA-LEVEL T-PROC: NONE;                                       DBA00150
*****    INTRA-LEVEL S-PROC: DDB,DDV,DMB;  */                            DBA00160
%INCLUDE EDDB,EDMB;                                                      DBA00170
/*****                                                                   DBA00180
*****    CONTROL STRUCTURE SERVICES:  */                                 DBA00190
/*****                                                                   DBA00200
***********************************************************************/ DBA00210
                                                                         DBA00220
 DCL NAME CHAR(25),                                                      DBA00230
     TRUE BIT(1) INIT('1'B);                                             DBA00240
                                                                         DBA00250
/* BEGIN SESSION */                                                     DBA00260
PUT SKIP(2) LIST ('-- DATABASE ADMINISTRATOR (DBA) SESSION --');         DBA00270
                                                                         DBA00280
/* GET COMMAND */                                                       DBA00290
DO WHILE(TRUE);                                                          DBA00300
   PUT SKIP(2) LIST                                                      DBA00310
   ('DBA: DATA DEFINITION (DD) OR DATA MANIPULATION (DM) OR DATA DEFINI  DBA00320
TION QUERY (DDQ)?');                                                     DBA00330
   GET EDIT (NAME) (A(25));                                              DBA00340
   IF NAME = (25)' ' THEN LEAVE;   /*OUT OF DBA SUBSYSTEM*/              DBA00350
                                                                         DBA00360
   /* SELECT COMMAND */                                                 DBA00370
   SELECT(NAME);                                                         DBA00380
                                                                         DBA00390
      /* DATA DEFINITION */                                              DBA00400
      WHEN('DD','DATA DEFINITION')DO WHILE ('1'B);                       DBA00410
                                                                         DBA00420
          /*SELECT AMONG BASE DATA DEF MODULE AND VIEW DATA DEF*/        DBA00430
          PUT SKIP LIST ('DD: BASE DATA (BASE) OR VIEW DATA (VIEW)');    DBA00440
          GET EDIT (NAME) (A(25));                                       DBA00450
          IF NAME = (25)' ' THEN LEAVE; /*OUT OF DD MODE*/               DBA00460
                                                                         DBA00470
          SELECT (NAME);                                                 DBA00480
             WHEN ('BASE DATA','BASE') CALL DDB;                         DBA00490
             WHEN ('VIEW DATA','VIEW') CALL DDV;                         DBA00500
             OTHERWISE PUT SKIP LIST (NAME,' IS NOT A VALID CMD.');      DBA00510
             END;                                                        DBA00520
                                                                         DBA00530
          END /*DO WITHIN DD*/;                                          DBA00540
                                                                         DBA00550
```

```
          /* QUERIES */                                                    DBA00560
                                                                           DBA00570
          WHEN('DATA MANIPULATION','DM') CALL DMB;                         DBA00580
                                                                           DBA00590
          WHEN ('DATA DEFINITION QUERY','DDQ') CALL DDQ;                   DBA00600
                                                                           DBA00610
          /* INVALID OPERATORS */                                          DBA00620
          OTHERWISE                                                        DBA00630
          PUT SKIP(0) EDIT (NAME,' IS AN INVALID COMMAND') (A,A);          DBA00640
          END;                                                             DBA00650
        END;                                                               DBA00660
                                                                           DBA00670
/* END SESSION */                                                          DBA00680
     PUT SKIP(2) LIST ('-- END OF DBA SESSION --');                        DBA00690
                                                                           DBA00700
/*********************************************************************/   DBA00710
DDV: PROC;                                                                 DBA00720
PUT SKIP LIST ('DATABASE VIEW DEFINITION NOT AVAILABLE.');                 DBA00730
RETURN;                                                                    DBA00740
END;                                                                       DBA00750
/*********************************************************************/   DBA00760
     END /*DBA*/;                                                         DBA00770
```

```
DDB:PROC;                                                              DDB00010
/*********************************************************************  DDB00020
*                                                                    *  DDB00030
*                    MODULE    DESCRIPTION                           *  DDB00040
*********************************************************************/ DDB00050
/*********************************************************************  DDB00060
*****    PURPOSE:  THIS MODULE IS THE INTERFACE ROUTINE USED BY THE     DDB00070
*****     DBA TO DEFINE THE BASE DATA IN THE ENTITY NETWORK MODEL.      DDB00080
*********************************************************************  DDB00090
*****    METHOD:                                                        DDB00100
*****      THIS MODULE MAKES USE OF THE PANEL MANAGER AVAILABLE ON      DDB00110
*****      CMS TO INTERACT WITH THE USER BY DISPLAYING THE PREDESIGNED  DDB00120
*****      PANEL FOR USER TO ENTER DATA DEFINITION.                     DDB00130
*********************************************************************  DDB00140
*****    INPUT PARAMETERS:   AS SHOWN IN THE PANEL;                     DDB00150
*****                                                                   DDB00160
*********************************************************************  DDB00170
*****    OUTPUT PARAMETERS:  AS SHOWN IN DEFE_ARG & DEFA_ARG, WHICH     DDB00180
*****      ARE DATA STRUCTURES FOR COMMUNICATING WITH MODULE DEFE&DEFA. DDB00190
*********************************************************************  DDB00200
*****    CALLS PROCEDURES:                                              DDB00210
*****     INTER-LEVEL T-PROC:  DEFE,DEFA;                               DDB00220
*/ %INCLUDE ADEFE,ADEFA; /* ARGUMENT DCLS*****                          DDB00230
*****     INTRA-LEVEL T-PROC: NONE;                                     DDB00240
*****     INTRA-LEVEL S-PROC: NONE;                                     DDB00250
*****     CONTROL STRUCTURE,PANEL MANAGEMENT & DEBUGGIN SERVICES:       DDB00260
*/ %INCLUDE SERVICE,FDEBUG,EUDPLI;  /*******                          DDB00270
*********************************************************************/ DDB00280
DCL 1 TEMP, /*USED AS TEMPORARY STORAGE FOR USER INPUT BEFORE CONV*/    DDB00290
      2 MLEN CHAR (2),                                                 DDB00300
      2 MAX CHAR (6),                                                  DDB00310
      2 MIN CHAR (6),                                                  DDB00320
      2 FUNC CHAR (3);                                                 DDB00330
DCL 1 UNLDLIST,                                                        DDB00340
      2 UNLOAD (9) PTR,                                                DDB00350
      2 FENCE BIT (32) INIT ((32)'1'B);                                DDB00360
DCL 1 LDLIST,                                                          DDB00370
    2 LD PTR,                                                          DDB00380
    2 FENCE BIT (32) INIT ((32)'1'B);                                  DDB00390
DCL RTN_CODE FIXED BIN (15);                                           DDB00400
DCL (P,TP,RP,LP) PTR, RSTATUS BIT (8);                                 DDB00410
DCL ADDR BUILTIN;                                                      DDB00420
DCL NULL BUILTIN;                                                      DDB00430
DCL CURSOR FIXED BIN;                                                  DDB00440
DCL PNAME CHAR (8);                                                    DDB00450
DCL    1 ARG_LIST BASED (P),                                           DDB00460
        2 LEN FIXED BIN,                                               DDB00470
        2 PTR PTR;                                                     DDB00480
DCL ENAME1 CHAR (8);/*USED TO HOLD PARENT ENTITYNAME*/                 DDB00490
DCL CMD CHAR (8);                                                      DDB00500
                                                                       DDB00510
 /*BEGIN PROCESSING*/                                                  DDB00520
                                                                       DDB00530
PUT SKIP LIST ('-BASE DATA DEFINITION SESSION-');                      DDB00540
PUT SKIP LIST ('YOU MAY DEFINE NEW ENTITY SETS OR ADD ATTRIBUTES TO EXIDDB00550
```

```
 STING ENTITY SETS.');                                                    DDB00560
                                                                          DDB00570
 DO WHILE ('1'B); /*LOOP FOR ENTITY SET DEFINITIONS*/                     DDB00580
                                                                          DDB00590
 PUT SKIP LIST ('NEW ENTITY SET (NEW) OR EXISTING ENTITY SET (OLD)?');    DDB00600
 GET EDIT (CMD) (A(8));                                                   DDB00610
 IF CMD=(8)' ' THEN LEAVE; /* GET OUT OF DATA DEFINITION */              DDB00620
 SELECT (CMD);                                                            DDB00630
                                                                          DDB00640
   WHEN ('NEW') DO;                                                       DDB00650
                                                                          DDB00660
       ALLOCATE DEFE_ARG;                                                 DDB00670
       TP,RP=P; /*SET TEMP & ROOT PTRS*/                                  DDB00680
       PNAME='PANNDS';  CURSOR=1; LD = ADDR (DEFE_ARG.NAME);              DDB00690
       PDISPLAY PNAME(PNAME) RSTATUS (RSTATUS) ULDLIST(LDLIST);           DDB00700
       IF DEFE_ARG.NAME=(8) ' '                                           DDB00710
       THEN GO TO RECYCLE_DEFE;  /*NO MORE NEW ESETS */                   DDB00720
       ENAME1=DEFE_ARG.NAME;/*SAVE ENTITY NAME*/                          DDB00730
                                                                          DDB00740
       /*SET UP DEFE CALL*/                                               DDB00750
       CALL TCALL ('DEFE',1,RP,TP);                                       DDB00760
       CALL SVC3 (RP); /*FREE UP CALLING ARGUMENTS*/                      DDB00770
                                                                          DDB00780
       /*CHECK RETURN CODE*/                                              DDB00790
       IF TP->DEFE_RTN.RTN_CODE ^=0                                       DDB00800
       THEN DO;                                                           DDB00810
          CALL DEFEMSG (TP->DEFE_RTN.RTN_CODE);                           DDB00820
          CALL SVC3(TP); /*FREE UP RTN MSG*/                              DDB00830
          GO TO RECYCLE_DEFE;                                             DDB00840
          END;/* ILLEGAL ENTITY SET DEF */                               DDB00850
                                                                          DDB00860
       CALL SVC3 (TP); /*FREE UP RTN MSG*/                                DDB00870
       CALL CONFIRM (1,ENAME1);                                           DDB00880
       GO TO ATTR; /* START DEFINE ATTRIBUTE LOOP */                     DDB00890
       END /* WHEN NEW */;                                               DDB00900
                                                                          DDB00910
   WHEN ('OLD') DO;/* OLD ENTITY SET */                                   DDB00920
                                                                          DDB00930
        PNAME='PANNDS';  CURSOR=1; LD=ADDR(ENAME1);                       DDB00940
        PDISPLAY PNAME (PNAME) RSTATUS (RSTATUS) ULDLIST (LDLIST);        DDB00950
        IF ENAME1=(8)' ' THEN GO TO RECYCLE_DEFE; /* GET OUT */           DDB00960
        GO TO ATTR;                                                       DDB00970
        END /* WHEN OLD */;                                              DDB00980
                                                                          DDB00990
    OTHERWISE DO;                                                         DDB01000
       PUT SKIP LIST ('ILLEGAL, PLEASE RE-ENTER');                        DDB01010
       GO TO RECYCLE_DEFE;                                                DDB01020
       END;                                                               DDB01030
                                                                          DDB01040
    END /* SELECT */;                                                    DDB01050
                                                                          DDB01060
  ATTR:  DO WHILE ('1'B); /*ATTRIBUTE DEF LOOP */                         DDB01070
     STRING(TEMP)=REPEAT(' ',100); /*INITIALIZE UNLOAD VAR TO BLANKS*/    DDB01080
     ALLOCATE DEFA_ARG;                                                   DDB01090
     RP=P;                                                                DDB01100
```

```
UNLOAD (1)=ADDR(RP->DEFE_ARG.NAME);                                    DDB01110
UNLOAD (2) =ADDR (DEFA_ARG.NAME);                                      DDB01120
UNLOAD(3)= ADDR (     TEMP.FUNC);                                      DDB01130
UNLOAD (4) = ADDR (DEFA_ARG.TYPE);                                     DDB01140
UNLOAD(5) = ADDR (DEFA_ARG.ENAME);                                     DDB01150
UNLOAD (6) = ADDR (TEMP.MLEN);                                         DDB01160
UNLOAD(7)=ADDR(DEFA_ARG.VTYPE);                                        DDB01170
UNLOAD(8) = ADDR (TEMP.MAX);                                           DDB01180
UNLOAD(9) = ADDR (TEMP.MIN);                                           DDB01190
PNAME='PANASD'; CURSOR=2;                                              DDB01200
PDISPLAY PNAME(PNAME) RSTATUS (RSTATUS) ULDLIST (UNLDLIST)             DDB01210
         CURSOR (CURSOR);/* GET DATA FROM PANEL */                     DDB01220
                                                                       DDB01230
IF DEFA_ARG.NAME = (8) ' ' THEN LEAVE; /*NO MORE ATTRIBUTE*/           DDB01240
DEFA_ARG.ENAME1=ENAME1; /*GIVE THE PARENT ENTITY NAME*/                DDB01250
                                                                       DDB01260
/*PERFORM CONVERSION*/                                                 DDB01270
SELECT (DEFA_ARG.TYPE);                                                DDB01280
   WHEN ('V') DO;                                                      DDB01290
      IF TEMP.MLEN=(2)' ' THEN TEMP.MLEN='20';                         DDB01300
      IF VERIFY(TEMP.MLEN,'0123456789 ') ^= 0                          DDB01310
      THEN DO;                                                         DDB01320
         CALL DEFAMSG (-1);                                            DDB01330
         GO TO RECYCLE_DEFA;                                           DDB01340
         END /* ILLEGAL MLEN */;                                      DDB01350
      DEFA_ARG.MLEN=TEMP.MLEN;/* CONVERSION */                        DDB01360
                                                                       DDB01370
      IF DEFA_ARG.VTYPE=' ' THEN DEFA_ARG.VTYPE='C';                   DDB01380
      IF VERIFY(DEFA_ARG.VTYPE,'NC') ^= 0                              DDB01390
      THEN DO;                                                         DDB01400
         CALL DEFAMSG (-2); /* ILLEGA VTYPE */                        DDB01410
         GO TO RECYCLE_DEFA;                                           DDB01420
         END; /* ILLEGAL VTYPE */                                     DDB01430
                                                                       DDB01440
      IF DEFA_ARG.VTYPE='N' THEN DO;/* CHECK MAX, MIN */               DDB01450
         IF VERIFY (TEMP.MAX,'0123456789 ') ^= 0 |                     DDB01460
            VERIFY (TEMP.MIN,'0123456789 ') ^= 0                       DDB01470
         THEN DO;                                                      DDB01480
            CALL DEFAMSG (-3); /* ILLEGAL NUMERIC RANGE */             DDB01490
            GO TO RECYCLE_DEFA;                                        DDB01500
            END;                                                       DDB01510
         IF TEMP.MAX=(6)' ' THEN TEMP.MAX='99999';                     DDB01520
         IF TEMP.MIN=(6)' ' THEN TEMP.MIN='-99999';                    DDB01530
         DEFA_ARG.MAX=TEMP.MAX;/* CONVERSION */                       DDB01540
         DEFA_ARG.MIN=TEMP.MIN;                                        DDB01550
         END;                                                          DDB01560
      END /* 'V' */;                                                  DDB01570
                                                                       DDB01580
   WHEN ('E');                                                         DDB01590
                                                                       DDB01600
   OTHERWISE DO;                                                       DDB01610
      CALL DEFAMSG (-4); /* ILLEGAL ATTRIBUTE TYPE */                 DDB01620
      GO TO RECYCLE_DEFA;                                              DDB01630
      END /* OTHERWISE */;                                            DDB01640
                                                                       DDB01650
```

```
        END /* SELECT */;                                                    DDB01660
                                                                            DDB01670
     SELECT (TEMP.FUNC);/* CHECK FUNCTION TYPE */                          DDB01680
        WHEN ('1:1') DEFA_ARG.FUNC='S';                                     DDB01690
        WHEN ('M:1') DEFA_ARG.FUNC='M';                                     DDB01700
        WHEN ('KEY') DEFA_ARG.FUNC='K';                                     DDB01710
        WHEN (' ') DEFA_ARG.FUNC = 'M'; /* DEFAULT TO M */                  DDB01720
        OTHERWISE DO;                                                       DDB01730
           CALL DEFAMSG (-5); /* ILLEGAL FUNCTION TYPE */                  DDB01740
           GO TO RECYCLE_DEFA;                                              DDB01750
           END;                                                             DDB01760
                                                                            DDB01770
        END; /* SELECT */                                                   DDB01780
                                                                            DDB01790
       /*PASS MSG TO DEFA*/                                                 DDB01800
    CALL TCALL ('DEFA',1,RP,TP);                                            DDB01810
    CALL SVC3 (RP); /*FREE CALL ARG LIST*/                                  DDB01820
                                                                            DDB01830
    RTN_CODE=TP->DEFA_RTN.RTN_CODE;                                         DDB01840
    CALL SVC3 (TP);                                                         DDB01850
                                                                            DDB01860
    IF RTN_CODE ^= 0 THEN DO;                                               DDB01870
       CALL DEFAMSG (RTN_CODE);                                             DDB01880
       GO TO RECYCLE_DEFA;                                                  DDB01890
       END; /*RTN_CODE IF*/                                                 DDB01900
                                                                            DDB01910
    ELSE CALL CONFIRM (2,DEFA_ARG.NAME);                                    DDB01920
                                                                            DDB01930
RECYCLE_DEFA:                                                               DDB01940
   END /*ATTR DEF LOOP DO WHILE '1'B*/;                                     DDB01950
                                                                            DDB01960
RECYCLE_DEFE:                                                               DDB01970
END /*ENTITY SET DEF LOOP*/;                                                DDB01980
                                                                            DDB01990
                                                                            DDB02000
/******** SUBROUTINE FOR CONFIRM MESSAGE AND ERROR MESSAGE *********/      DDB02010
CONFIRM: PROC (CODE,NAME);                                                  DDB02020
DCL CODE FIXED BIN (31);                                                    DDB02030
DCL NAME CHAR(8);                                                           DDB02040
SELECT (CODE);                                                              DDB02050
   WHEN (1) PUT SKIP EDIT ('ENTITY SET ',NAME,' DEFINED') (A,A,A);          DDB02060
   WHEN (2) PUT SKIP EDIT ('ATTRIBUTE ',NAME,' DEFINED') (A,A,A);           DDB02070
   OTHERWISE;                                                               DDB02080
   END /* SELECT */;                                                       DDB02090
CALL PROMPT;                                                                DDB02100
END /*CONFIRM */;                                                           DDB02110
                                                                            DDB02120
/*********************************************************************/     DDB02130
DEFEMSG: PROC (CODE);                                                       DDB02140
DCL CODE FIXED BIN (31);                                                    DDB02150
PUT SKIP LIST ('ILLEGAL ENTITY SET NAME: DEFINITION IGNORED.');             DDB02160
CALL PROMPT;                                                                DDB02170
END /* DEFEMSG */;                                                         DDB02180
                                                                            DDB02190
/*********************************************************************/     DDB02200
```

```
DEFAMSG: PROC (CODE);                                                    DDB02210
DCL CODE FIXED BIN (31);                                                 DDB02220
SELECT (CODE);                                                           DDB02230
   WHEN (-1) PUT SKIP LIST ('ILLEGAL MAXIMUM LENGTH.');                  DDB02240
   WHEN (-2) PUT SKIP LIST ('ILLEGAL VALUE TYPE.');                      DDB02250
   WHEN (-3) PUT SKIP LIST ('ILLEGAL NUMERIC RANGE.');                   DDB02260
   WHEN (-4) PUT SKIP LIST ('ILLEGAL ATTRIBUTE TYPE.');                  DDB02270
   WHEN (-5) PUT SKIP LIST ('ILLEGAL FUNCTION TYPE.');                   DDB02280
   WHEN (5) PUT SKIP LIST ('ILLEGAL ENTITY SET NAME.');                  DDB02290
   WHEN (1) PUT SKIP LIST ('DUPLICATE ATTRIBUTE NAME.');                 DDB02300
   WHEN (7) PUT SKIP LIST ('UNKNOWN DOMAIN ENTITY SET NAME');            DDB02310
   OTHERWISE;                                                            DDB02320
   END /* SELECT */;                                                    DDB02330
PUT SKIP LIST ('ILLEGAL ATTRIBUTE DEFINITION: DEFINITION IGNORED.');     DDB02340
CALL PROMPT;                                                             DDB02350
END /*DEFAMSG */;                                                       DDB02360
                                                                         DDB02370
/*************************************************************************/ DDB02380
PROMPT: PROC;                                                            DDB02390
PUT SKIP LIST ('PRESS ENTER TO CONTINUE');                               DDB02400
GET LIST (CMD);                                                          DDB02410
END;                                                                     DDB02420
/*************************************************************************/ DDB02430
                                                                         DDB02440
END /*DDB*/;                                                            DDB02450
```

8711

```
DDQ: PROC;                                                              DDQ00010
/*************************************************************************  DDQ00020
*                                                                       *  DDQ00030
*                    MODULE    DESCRIPTION                            *  DDQ00040
*************************************************************************/ DDQ00050
/*************************************************************************  DDQ00060
*****    PURPOSE:                                                         DDQ00070
** S-PROC AT THE USER INTERFACE LEVEL, CALLED AS A SUBROUTINE BY THE     DDQ00080
* DBA MODULE, TO PROCESS DBA'S REQUEST TO QUERY DATA DEFINITIONS.        DDQ00090
* THIS MODULE MAKES USE OF THE T-PROC AT THE NEXT LEVEL, SHWE, TO        DDQ00100
* OBTAIN INFORMATION ABOUT ENTITY SETS AND ATTRIBUTES AND DISPLAYS IT.   DDQ00110
*****                                                                    DDQ00120
*************************************************************************/ DDQ00130
                                                                        DDQ00140
%INCLUDE ASHWE.SERVICE.ELEX;                                            DDQ00150
DCL ENAME CHAR (8);                                                     DDQ00160
DCL(NUM_TOKEN,I) FIXED BIN;                                             DDQ00170
DCL RTN_CODE FIXED BIN;                                                 DDQ00180
DCL 1 TOKEN(25), /* USED TO COMM. WITH LEX */                           DDQ00190
     2 D CHAR (40),                                                     DDQ00200
     2 L FIXED BIN;                                                     DDQ00210
DCL NO_BLANKS FIXED BIN INIT (1);                                       DDQ00220
DCL FUNC1 CHAR (3);                                                     DDQ00230
DCL NULL BUILTIN, P PTR;                                                DDQ00240
                                                                        DDQ00250
/* BEGIN */                                                             DDQ00260
PUT SKIP LIST ('-- DATA DEFINITION QUERY SESSION --');                  DDQ00270
                                                                        DDQ00280
DO WHILE ('1'B);                                                        DDQ00290
   PUT SKIP LIST ('ENTITY SET NAME? USE * IF ALL ENTITY SET NAMES ARE DDDQ00300
ESIRED.');                                                              DDQ00310
   CALL LEX (NUM_TOKEN,TOKEN,NO_BLANKS,RTN_CODE);                       DDQ00320
   IF NUM_TOKEN=0  THEN LEAVE; /* GET OUT */                           DDQ00330
   ENAME = TOKEN.D(1);                                                  DDQ00340
   ALLOCATE SHWE_ARG;                                                   DDQ00350
   SHWE_ARG.NAME1=ENAME;                                                DDQ00360
   IF ENAME = '*' THEN GO TO SEND; /* WANT ALL */                       DDQ00370
                                                                        DDQ00380
   LOOP:                                                                DDQ00390
   PUT SKIP LIST ('ATTRIBUTE NAMES? SEPERATE BY COMMAS. USE * IF ALL ATDDQ00400
TRIBUTES ARE DESIRED');                                                 DDQ00410
   CALL LEX (NUM_TOKEN,TOKEN,NO_BLANKS,RTN_CODE); /* CALL LEX TO GET   DDQ00420
                                                     LINE */            DDQ00430
   IF RTN_CODE ^= 0 THEN DO;                                            DDQ00440
      PUT SKIP LIST ('ILLEGAL SYNTAX');                                 DDQ00450
      GO TO LOOP; /* TRY AGAINS */                                      DDQ00460
      END;                                                              DDQ00470
                                                                        DDQ00480
   IF NUM_TOKEN=0 THEN LEAVE;          /*GET OUT*/                     DDQ00490
                                                                        DDQ00500
   IF SUBSTR(TOKEN.D(1),1,1)='*'       /* ALL */                       DDQ00510
   THEN SHWE_ARG.NUM_NAME2=999;        /* MEANING ALL */               DDQ00520
   ELSE DO;                            /* SELECTIVE */                 DDQ00530
      SHWE_ARG.NUM_NAME2=NUM_TOKEN;                                     DDQ00540
      DO I = 1 TO NUM_TOKEN;                                            DDQ00550
```

```
         SHWE_ARG.NAME2(I) = TOKEN.D(I);                                    DDQ00560
         END;                                                               DDQ00570
      END;                                                                  DDQ00580
                                                                            DDQ00590
   SEND:                                                                    DDQ00600
   PUT  LIST ('*** DDQ TO CALL TCALL SHWE');                                DDQ00610
   CALL TCALL ('SHWE', 1, SHWE_ARG_P, SHWE_RTN_P);                          DDQ00620
   CALL SVC3 (SHWE_ARG_P);                                                  DDQ00630
                                                                            DDQ00640
 /* EXAMINE RETURN CODE */                                                 DDQ00650
   I = SHWE_RTN.RTN_CODE;                                                   DDQ00660
   IF I ^=0 THEN DO;                  /*ERROR RETURN FROM SHWE       */    DDQ00670
      CALL SHWEMSG (I);                                                     DDQ00680
      CALL SVC3 (SHWE_RTN_P);                                               DDQ00690
      GO TO END; /* TRY AGAIN */                                            DDQ00700
      END;                                                                  DDQ00710
                                                                            DDQ00720
   IF ENAME='*' THEN DO;                                                    DDQ00730
      P=SHWE_RTN_P; /* SAVE IT */                                          DDQ00740
      SHWE_RTN_P=SHWE_RTN.PTR; /* ->SHWE_RTN2 */                           DDQ00750
      PUT SKIP LIST ('NUMBER OF ENTITY SETS DEFINED', NUM_ENAME);           DDQ00760
      DO I = 1 TO NUM_ENAME;                                                DDQ00770
         IF MOD(I,8)= 1 THEN PUT SKIP EDIT (SHWE_RTN2.ENAME(I))(A(8));      DDQ00780
         ELSE PUT EDIT (SHWE_RTN2.ENAME(I)) (COL((I-1)*9+1),A(8));          DDQ00790
        END;                                                                DDQ00800
      CALL SVC3 (P);                                                        DDQ00810
      END /* IF ENAME = * */;                                              DDQ00820
                                                                            DDQ00830
   ELSE DO;                                                                 DDQ00840
      PUT SKIP LIST ('ENTITY SET NAME', ENAME);                             DDQ00850
      PUT SKIP EDIT ('ATTRIBUTE NAME  FUNCTION  TYPE  ENAME     VTYPE       DDQ00860
MAX LEN  MAX VALUE  MIN VALUE') (A);                                        DDQ00870
      P=SHWE_RTN_P; /* SAVE IT */                                          DDQ00880
      DO SHWE_RTN_P = SHWE_RTN.PTR REPEAT SHWE_RTN1.PTR WHILE               DDQ00890
      (SHWE_RTN_P ^= NULL);                                                 DDQ00900
          IF TYPE ^= 'E' THEN ENAME = (8)' '; ELSE VTYPE = '';              DDQ00910
          SELECT (FUNC);                                                    DDQ00920
             WHEN ('S') FUNC1='1:1';                                        DDQ00930
             WHEN ('K') FUNC1='KEY';                                        DDQ00940
             WHEN ('M') FUNC1='M:1';                                        DDQ00950
             OTHERWISE;                                                     DDQ00960
             END /* SELECT */ ;                                            DDQ00970
          PUT SKIP EDIT (ANAME,FUNC1,TYPE,SHWE_RTN1.DATA.ENAME,             DDQ00980
VTYPE) (X(6),A(8),X(7),A(3),                                                DDQ00990
           X(5),A(1),X(2),A(8),X(6),A(1));                                  DDQ01000
          IF TYPE = 'V' THEN DO;                                            DDQ01010
             PUT EDIT (MLEN) (COL(52),F(5));                                DDQ01020
             IF VTYPE = 'N'                                                 DDQ01030
             THEN PUT EDIT (MAX,MIN)(COL(62),F(6),X(5),F(6));               DDQ01040
             END;                                                           DDQ01050
                                                                            DDQ01060
          END /* DO P */;                                                   DDQ01070
       CALL SVC3 (P);                                                       DDQ01080
       END /* ELSE */;                                                     DDQ01090
                                                                            DDQ01100
```

```
     END;                                                                DDQO1110
     END; /* DO WHILE */                                                 DDQO1120
                                                                         DDQO1130
 /*******************************************************************/  DDQO1140
 SHWEMSG: PROC(CODE);     /* INTERNAL MSG SUBROUTINE          */        DDQO1150
 DCL CODE FIXED BIN;                                                     DDQO1160
                                                                         DDQO1170
 IF CODE < 0                                                             DDQO1180
 THEN DO;                                                                DDQO1190
    SELECT (CODE);                                                       DDQO1200
       WHEN (-1) PUT SKIP LIST ('ILLEGAL ENTITY NAME');                  DDQO1210
       WHEN (-2) PUT SKIP LIST ('NO ENTITY SETS DEFINED.');              DDQO1220
       WHEN (-3) PUT SKIP EDIT ('NO ATTRIBUTES DEFINED FOR ENTITY SET',  DDQO1230
           ENAME)(A,A);                                                  DDQO1240
       OTHERWISE;                                                        DDQO1250
       END /*SELECT */;                                                  DDQO1260
    END /* CODE < 0 */;                                                 DDQO1270
                                                                         DDQO1280
 ELSE IF CODE <= NUM_TOKEN                                               DDQO1290
 THEN PUT SKIP LIST                                                      DDQO1300
           ('ILLEGAL ATTRIBUTE NAME', TOKEN.D(I));                       DDQO1310
                                                                         DDQO1320
 PUT SKIP LIST ('ILLEGAL DD QUERY STATEMENT');                           DDQO1330
 PUT SKIP LIST ('PLEASE RE-ENTER');                                      DDQO1340
 END /* SHWEMSG */;                                                     DDQO1350
 /*******************************************************************/  DDQO1360
                                                                         DDQO1370
 END /* DDQ */;                                                         DDQO1380
```

```
DEFA:PROC;                                                              DEF00010
                                                                        DEF00020
/* A T-PROC AT THE ENTITY LEVEL FOR DEFINING ATTRIBUTES*/               DEF00030
                                                                        DEF00040
%INCLUDE ADEFA, ARETE, AUPDE, SERVICE, FDEBUG,ADEFE;                    DEF00050
%INCLUDE DECAT; /*CATALOGUE TEMPLATES*/                                 DEF00060
%INCLUDE EGACT,EGECT,EDFP1,EDFB1;                                       DEF00070
                                                                        DEF00080
DCL (P,MP,TP,RP,RTP,LP) PTR, NULL BUILTIN;                              DEF00090
DCL (ENAME,ANAME) CHAR (8), RTN_CODE FIXED BIN;                         DEF00100
DCL (PSETID1,PSETID2,BSETID) BIT (32);                                  DEF00110
DCL BIT_STRING BIT (232) BASED; /*AINFO IS 19 BYTES LONG                DEF00120
                  ENCODED STRING*/                                      DEF00130
DCL (STRING, UNSPEC) BUILTIN;                                           DEF00140
                                                                        DEF00150
 /*BEGIN*/                                                             DEF00160
CALL TBEG (PROC_ADDR, MP);                                              DEF00170
ALLOCATE DEFA_RTN SET (RTP);                                            DEF00180
                                                                        DEF00190
AINFO=MP->DEFA_ARG, BY NAME;                                            DEF00200
ENAME=MP->DEFA_ARG.ENAME1; /*PARENT ENTITY NAME*/                       DEF00210
ANAME=MP->DEFA_ARG.NAME; /*ATTRIBUTE NAME*/                             DEF00220
                                                                        DEF00230
/*CHECK PARENT ENTITY SET*/                                             DEF00240
CALL GECT (ENAME, EINFO, RTN_CODE);                                     DEF00250
 IF RTN_CODE^=0 THEN DO;                                                DEF00260
   RTP->DEFA_RTN.RTN_CODE=5; /***5 IS PROBLEM WITH PARENT ENTITY***/    DEF00270
   CALL TRTN(PROC_ADDR, RTP);                                           DEF00280
   RETURN;  /*RETURNS*/                                                DEF00290
   END;                                                                 DEF00300
 PSETID1=EINFO;                                                         DEF00310
                                                                        DEF00320
/*CHECK IF ATTRIBUTE NAME DUPLICATED*/                                  DEF00330
CALL GACT (ANAME, ENAME, AINFO, RTN_CODE);                              DEF00340
  IF RTN_CODE^=2 THEN DO;   /*2 FROM GACT IS NO SUCH ATTRIBUTE****/     DEF00350
      RTP->DEFA_RTN.RTN_CODE=1; /***1 IS DUPLICATE ATTRIBUTE NAME***/    DEF00360
      CALL TRTN (PROC_ADDR, RTP); /*RETURNS*/                          DEF00370
      RETURN;                                                           DEF00380
      END;                                                              DEF00390
                                                                        DEF00400
/*HANDLING TARGET PSET**/                                               DEF00410
IF AINFO.TYPE='V' THEN DO; /*MUST CREATE TARGET PSET*/                  DEF00420
   IF AINFO.VTYPE='N' THEN DO;/*NUMERIC*/                               DEF00430
 /*IF FDEBUG.LEV1 THEN PUT SKIP LIST ('DEFA: CREATING V PSET');  */     DEF00440
      CALL DFP1 ('N',4,'V',RTN_CODE, PSETID2);                          DEF00450
      END;                                                              DEF00460
   ELSE CALL DFP1 (AINFO.VTYPE, AINFO.MLEN, 'V', RTN_CODE, PSETID2);    DEF00470
   END;                                                                 DEF00480
                                                                        DEF00490
ELSE DO; /*ENTITY TYPE, CHECK TO SEE IF EXISTS ALREADY**/               DEF00500
A: CALL GECT (AINFO.ENAME, EINFO, RTN_CODE);                            DEF00510
   SELECT (RTN_CODE);                                                   DEF00520
                                                                        DEF00530
     WHEN (2) DO ; /*MEANS TARGET ENTITY SET DOES NOT EXIST YET*/       DEF00540
        /* IN THIS VERSION THIS MEANS AN ERROR; FUTURE VERSIONS         DEF00550
```

```
          WILL OPTIONALLY DEFINES THE TARGET ENTITY SET */          DEF00560
        RTP->DEFA_RTN.RTN_CODE=7; /*** 7 IS UNKNOWN TARGET ESET *****/ DEF00570
        CALL TRTN (PROC_ADDR, RTP);                                 DEF00580
        RETURN;                                                     DEF00590
        END /* WHEN 2*/;                                           DEF00600
                                                                    DEF00610
      WHEN (0)  /*GOT IT*/                                          DEF00620
        PSETID2=EINFO;                                              DEF00630
                                                                    DEF00640
      OTHERWISE DO; /*ILLEGA TARGET ESET*/                          DEF00650
        RTP->DEFA_RTN.RTN_CODE=3; /*3 IS PROBLEM IN TARGET ESET*/  DEF00660
        CALL TRTN(PROC_ADDR, RTP);                                  DEF00670
         RETURN;                                                    DEF00680
        END;                                                        DEF00690
                                                                    DEF00700
    END /*SELECT*/;                                                 DEF00710
   END /*ELSE DO ENTITY TYPE*/;                                     DEF00720
                                                                    DEF00730
 /*CREATE BSET*/                                                   DEF00740
 /*IF FDEBUG.LEV1 THEN PUT SKIP LIST ('DEFA: CREATING BSET'); */   DEF00750
  CALL DFB1 (AINFO.FUNC, PSETID1,PSETID2, RTN_CODE,BSETID);         DEF00760
                                                                    DEF00770
  AINFO.BSETID=BSETID;                                              DEF00780
  AINFO.PSETID=PSETID2;                                             DEF00790
                                                                    DEF00800
  /*GIVE IT TO UPDE**/                                            DEF00810
  ALLOCATE UPDE_ARG; RP,LP=P;    /*ASET NODE*/                    DEF00820
  UPDE_ARG.NAME='E*ASET  ';                                         DEF00830
  UPDE_ARG.OP=UPDEOP.CRT;                                           DEF00840
                                                                    DEF00850
  ALLOCATE UPDE_ARG; /*ESET NODE*/                                  DEF00860
  LP->UPDE_ARG.PTR=P; LP=P;                                         DEF00870
  UPDE_ARG.NAME='A*ESET  ';                                         DEF00880
  UPDE_ARG.OP=UPDEOP.IST;                                           DEF00890
  UPDE_ARG.PARENT=1;                                                DEF00900
  UPDE_ARG.NODE=2;                                                  DEF00910
                                                                    DEF00920
  ALLOCATE UPDE_ARG; /*ENAME NODE*/                                 DEF00930
  LP->UPDE_ARG.PTR=P; LP=P;                                         DEF00940
  UPDE_ARG.NAME='A*ENAME ';                                         DEF00950
  UPDE_ARG.OP=UPDEOP.ID;                                            DEF00960
  UPDE_ARG.PARENT=2;                                                DEF00970
  UPDE_ARG.DLEN=8;                                                  DEF00980
  UPDE_ARG.CDATA=ENAME;                                             DEF00990
  UPDE_ARG.NODE = 3;                                                DEF01000
                                                                    DEF01010
  ALLOCATE UPDE_ARG; /*ANAME NODE*/                                 DEF01020
  LP->UPDE_ARG.PTR=P; LP=P;                                         DEF01030
  UPDE_ARG.NAME='A*ANAME ';                                         DEF01040
  UPDE_ARG.OP=UPDEOP.IST;                                           DEF01050
  UPDE_ARG.PARENT=1;                                                DEF01060
  UPDE_ARG.DLEN=8;                                                  DEF01070
  UPDE_ARG.CDATA=ANAME;                                             DEF01080
  UPDE_ARG.NODE = 4;                                                DEF01090
                                                                    DEF01100
```

```
 ALLOCATE UPDE_ARG; /*AINFO NODE*/                                    DEFO1110
 LP->UPDE_ARG.PTR=P; LP=P;                                            DEFO1120
 UPDE_ARG.NAME='A*AINFO ';                                            DEFO1130
 UPDE_ARG.OP=UPDEOP.IST;                                              DEFO1140
 UPDE_ARG.NODE = 5;                                                   DEFO1150
 UPDE_ARG.PARENT=1;                                                   DEFO1160
 UPDE_ARG.DLEN=29;                                                    DEFO1170
 UNSPEC(UPDE_ARG.CDATA)=ADDR(AINFO)->BIT_STRING;                      DEFO1180
                                                                      DEFO1190
/*CALL UP UPDE*/                                                     DEFO1200
CALL TCALL ('UPDE',1,RP,TP);                                         DEFO1210
                                                                      DEFO1220
/*CHECK RETURN*/                                                     DEFO1230
IF TP->UPDE_RTN.RTN_CODE^=0 THEN                                     DEFO1240
   RTP->DEFA_RTN.RTN_CODE=6; /**6 IS PROBLEM INUPDATING ENTITY CAT.*/ DEFO1250
                                                                      DEFO1260
/*RETURNS*/                                                          DEFO1270
 CALL SVC3 (RP);                                                      DEFO1280
 CALL SVC3 (TP);                                                      DEFO1290
 CALL TRTN( PROC_ADDR, RTP);                                          DEFO1300
 RETURN;                                                              DEFO1310
 END /*DEFA*/ ;                                                       DEFO1320
```

```
DEFB:PROC;                                                              DEF00010
                                                                        DEF00020
/** A T-PROC AT THE N-ARY LEVEL FOR DEFINING BINARY ASSOCIATION SET*/   DEF00030
                                                                        DEF00040
%INCLUDE SERVICE,ADEFB,DPCAT,DBEU,DNKEY,DBCAT;                          DEF00050
%INCLUDE EPCT,EMPST,ECRTP;                                              DEF00060
                                                                        DEF00070
DCL MAP BIT (32),(P,MP,RTP,TP)PTR,(NULL,UNSPEC,STRING,ADDR) BUILTIN;    DEF00080
DCL (POS,RTN_CODE) FIXED BIN;                                           DEF00090
DCL 1 BIT_STRING BIT (88) BASED; /*BIT TEMPLATE OF BINFO*/              DEF00100
DCL ID BIT (32);                                                        DEF00110
                                                                        DEF00120
/*BEGIN PROCESSING*/                                                    DEF00130
CALL TBEG (PROC_ADDR, MP);                                              DEF00140
ALLOCATE DEFB_RTN SET (RTP); RTP->DEFB_RTN.RTN_CODE=0;                  DEF00150
                                                                        DEF00160
BINFO=MP->DEFB_ARG, BY NAME;                                            DEF00170
                                                                        DEF00180
/*ASSIGN ID ARRAY IN PSETID(1)*/                                        DEF00190
CALL GPCT(BINFO.PSETID(1), PINFO,RTN_CODE); /*OBTAIN CAT ENTRY*/        DEF00200
MAP=PINFO.MAP;                                                          DEF00210
CALL MPST (MAP, POS);                                                   DEF00220
IF POS=0 THEN DO;                                                       DEF00230
        RTP->DEFB_RTN.RTN_CODE=1; /**1 IS ID ARRAY OVERFLOW****/        DEF00240
        CALL TRTN(PROC_ADDR,RTP);                                       DEF00250
        RETURN;                                                         DEF00260
        END;                                                            DEF00270
                                                                        DEF00280
BINFO.POS=POS;                                                          DEF00290
                                                                        DEF00300
 /*UPDATE PSETCAT ENTRY TO REFLECT USAGE OF PTR IN ID_ARRAY*/           DEF00310
 PINFO.MAP=MAP;                                                         DEF00320
 CALL UPCT(BINFO.PSETID(1), PINFO,RTN_CODE);                            DEF00330
                                                                        DEF00340
CALL CRTP(BCATID,ADDR(BINFO)->BIT_STRING, ID, RTN_CODE);                DEF00350
  IF RTN_CODE^=0 THEN                                                   DEF00360
      RTP->DEFB_RTN.RTN_CODE=1+RTN_CODE; /*1+ IS FROM CRTP*****/        DEF00370
      ELSE RTP->DEFB_RTN.BSETID=ID;                                     DEF00380
                                                                        DEF00390
 CALL TRTN (PROC_ADDR, RTP);                                            DEF00400
 RETURN;                                                                DEF00410
 END;                                                                   DEF00420
                                                                        DEF00430
```

```
DEFE:PROC;                                                              DEF00010
/*************************************************************************DEF00020
*                                                                       *DEF00030
*                    MODULE    DESCRIPTION                            *DEF00040
*************************************************************************/DEF00050
/*************************************************************************DEF00060
*****   PURPOSE: THIS ROUTINE ENABLES DEFINITION OF AN ENTITY SET;      DEF00070
*****        IT IS PAIRED WITH MODULE DEFA TO COMPLETE DD OF THE        DEF00080
*****     ENTITY NETWORK IN THE BASE DATABASE.                          DEF00090
*************************************************************************DEF00100
*****   METHOD:   1. CALLS UPDE  TO INSERT THE                          DEF00110
*****           ENTITY NAME INTO THE 'ESET' ENTITY; IF ERROR RETURN,    DEF00120
*****           THEN DUPLICATED ESET NAME IS DETECTED AND THE CALLER    DEF00130
*****           SO ADVISED;                                             DEF00140
*****        2. CALLS DEFP (THROUGH SBR DFP1) TO DEFINE A PSET          DEF00150
*****           CORRESPONDING TO THIS ENTITY SET;                       DEF00160
*****        3. CALLS UPDE AGAIN TO INSERT THE PSETID                   DEF00170
*****           INTO 'ESET' ENTITY SET;                                 DEF00180
*****        4. SET UP RETURN MSG FOR CALLER;                           DEF00190
*************************************************************************DEF00200
*****   INPUT PARAMETERS:  AS INDICATED BY DEFE_ARG;                    DEF00210
*****                                                                   DEF00220
*************************************************************************DEF00230
*****   OUTPUT PARAMETERS: AS INDICATED BY DEFE_RTN;                    DEF00240
*****                                                                   DEF00250
*************************************************************************DEF00260
***** CALLING ARG LIST :                                               */DEF00270
%INCLUDE ADEFE;                                                         DEF00280
/****   CALLS PROCEDURES:                                             *  DEF00290
*****    INTER-LEVEL T-PROC:   DEFP (THROUGH SBR DFP1)                *  DEF00300
*****    INTRA-LEVEL T-PROC: UPDE                                     */ DEF00310
%INCLUDE AUPDE;                                                         DEF00320
/****    INTRA-LEVEL S-PROC: DFP1,GECT                                */ DEF00330
%INCLUDE EDFP1,EGECT;                                                   DEF00340
/****    CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES:   */ DEF00350
%INCLUDE SERVICE,FDEBUG;                                                DEF00360
/*************************************************************************/DEF00370
DCL (P,TP,RP,TP1,RTP)PTR;                                               DEF00380
DCL RTN_CODE FIXED BIN;                                                 DEF00390
DCL NAME CHAR (8), PSETID BIT(32);                                      DEF00400
DCL NULL BUILTIN;                                                       DEF00410
DCL EINFO BIT (32);                                                     DEF00420
/*BEGIN PROCESSING*/                                                    DEF00430
                                                                        DEF00440
CALL TBEG (PROC_ADDR,P);                                                DEF00450
NAME=P->DEFE_ARG.NAME;                                                  DEF00460
                                                                        DEF00470
/*ALLOCAT DEFE_RTN*/                                                    DEF00480
ALLOCATE DEFE_RTN SET (RTP);                                            DEF00490
RTP->DEFE_RTN.RTN_CODE=0; /*INITIALIZE RTN CODE TO O.K.*/               DEF00500
                                                                        DEF00510
/*CHECK IF NAME DUPLICATED BY CALLING GECT*/                            DEF00520
   CALL GECT (NAME,EINFO,RTN_CODE);                                     DEF00530
  IF RTN_CODE ^= 2 THEN DO;/*NAME ALREAY EXISTS*****/                   DEF00540
   RTP-> DEFE_RTN.RTN_CODE = 1; /*ILLEGAL DEFE*/                        DEF00550
```

```
    GO TO RTN;                                                          DEF00560
    END;                                                                DEF00570
                                                                        DEF00580
    /*CALL SUBROUTINE TO CALL DEFP*/                                    DEF00590
    CALL DFP1 ('X', 0, 'E',RTN_CODE,PSETID);                            DEF00600
     IF RTN_CODE ^=0 THEN DO;                                           DEF00610
        IF FDEBUG.LEV1 THEN PUT SKIP LIST ('JUST RETURNED FROM          DEF00620
          DFP1 CALL, RTN_CODE SET TO',RTN_CODE);                        DEF00630
        RTP->DEFE_RTN.RTN_CODE=2; /*ILLEGAL DEFE*/                      DEF00640
        GO TO RTN;                                                      DEF00650
        END;                                                            DEF00660
                                                                        DEF00670
      /*CALL UPDE TO CREATE ECAT ENTRY FOR THIS ENTITY SET */           DEF00680
     /* BY CREATING AN ENTITY IN THE E*ESET ENTITY SET */               DEF00690
                                                                        DEF00700
    ALLOCATE UPDE_ARG;   RP=P; /*ROOT PTR*/                            DEF00710
    UPDE_ARG.NAME='E*ESET  ';                                           DEF00720
    UPDE_ARG.OP=UPDEOP.CRT;                                             DEF00730
    ALLOCATE UPDE_ARG;   /*2ND TOKEN*/                                 DEF00740
    RP->UPDE_ARG.PTR=P;   TP=P;                                         DEF00750
    UPDE_ARG.NAME='A*EINFO ';                                           DEF00760
    UPDE_ARG.PARENT=1;                                                  DEF00770
    UPDE_ARG.NODE=2;                                                    DEF00780
    UPDE_ARG.DLEN=4;                                                    DEF00790
    UNSPEC(UPDE_ARG.CDATA)=PSETID;     UPDE_ARG.OP=UPDEOP.IST;          DEF00800
    ALLOCATE UPDE_ARG; /*3RD TOKEN*/                                    DEF00810
    TP->UPDE_ARG.PTR=P;                                                 DEF00820
    UPDE_ARG.NAME='A*ENAME ';                                           DEF00830
    UPDE_ARG.PARENT=1;                                                  DEF00840
    UPDE_ARG.NODE=3;                                                    DEF00850
    UPDE_ARG.DLEN=8;                                                    DEF00860
    UPDE_ARG.CDATA=NAME;                                                DEF00870
    UPDE_ARG.OP=UPDEOP.IST;                                             DEF00880
                                                                        DEF00890
    CALL TCALL ('UPDE',1,RP,TP);                                        DEF00900
                                                                        DEF00910
    CALL SVC3 (RP); /*FREE CALLING ARG LIST*/                           DEF00920
      /*CHECK RTN_CODE*/                                                DEF00930
      IF TP->UPDE_RTN.RTN_CODE ^=0 THEN DO;                             DEF00940
       IF FDEBUG.LEV1 THEN PUT SKIP LIST ('2ND TIME ROUND FROM UPE1,    DEF00950
         RTN_CODE SET TO',TP->UPDE_RTN.RTN_CODE);                       DEF00960
       RTP->DEFE_RTN.RTN_CODE =3; /*ILLEGAL DEFE*/                      DEF00970
       END;                                                             DEF00980
      CALL SVC3 (TP); /*FREE UP RTN MSG*/                               DEF00990
                                                                        DEF01000
                                                                        DEF01010
   /*SET UP DEFE_RTN*/                                                 DEF01020
 RTN:CALL TRTN(PROC_ADDR,RTP);                                          DEF01030
 /*IF FDEBUG.LEV1 THEN  PUT SKIP LIST ('DEFE RETURNS'); */             DEF01040
   END /*DEFE*/;                                                        DEF01050
```

```
DEFP  PROC;                                                              DEF00010
                                                                         DEF00020
/*T-PROC AT THE N-ARY LEVEL FOR DEFINING PRIMITIVE SETS*/                DEF00030
                                                                         DEF00040
%INCLUDE SERVICE,ADEFP,DPCAT,DNKEY;                                      DEF00050
%INCLUDE EMPST,ECRTP;                                                    DEF00060
DCL (P,AP,RTP) PTR, (RTN_CODE,I) FIXED BIN, POS FIXED BIN;               DEF00070
DCL MAP BIT(32) INIT ((32)'0'B),(NULL,UNSPEC,STRING,ADDR)BUILTIN;        DEF00080
DCL ID BIT (32);                                                         DEF00090
DCL BIT_STRING BIT (144) BASED; /*A PSET CAT ENTRY IS 144 BYTES*/        DEF00100
                                                                         DEF00110
/**BEGIN PROCESSING***/                                                  DEF00120
CALL TBEG (PROC_ADDR,AP);                                                DEF00130
ALLOCATE DEFP_RTN SET (RTP);                                             DEF00140
RTP->DEFP_RTN.RTN_CODE=0;                                                DEF00150
                                                                         DEF00160
                                                                         DEF00170
PINFO=AP->DEFP_ARG, BY NAME; /*COPY INFO FROM DEFP_ARG TO PINFO*/        DEF00180
IF PINFO.PTYPE='X' THEN PINFO.PLEN=4; /*FIX UP 'X' TYPE*/                DEF00190
PINFO.NUMPTR=AP->DEFP_ARG.IMP.NUMPTR;                                    DEF00200
PINFO.LTYPE=PINFOLTYPE.LL;/*ONLY LINK LIST IS AVAILABLE IN THIS VER*/    DEF00210
PINFO.L_ID=UNSPEC(NULL());                                               DEF00220
                                                                         DEF00230
DO I=PINFO.NUMPTR+1 TO 32; /*INIT MAP*/                                  DEF00240
  CALL MPST(MAP,POS);                                                    DEF00250
END;                                                                     DEF00260
                                                                         DEF00270
CALL MPST(MAP,POS); /*FIND FORWARD LINK POS*/                            DEF00280
PINFO.L_POS=POS;                                                         DEF00290
CALL MPST(MAP,POS); /*FIND BACKWARD LINK POS*/                           DEF00300
PINFO.L_POS2=POS;                                                        DEF00310
PINFO.MAP=MAP;                                                           DEF00320
                                                                         DEF00330
  CALL CRTP(PCATID,ADDR(PINFO)->BIT_STRING,ID,RTN_CODE);                 DEF00340
  IF RTN_CODE^=0 THEN RTP->DEFP_RTN.RTN_CODE=RTN_CODE;                   DEF00350
   ELSE RTP->DEFP_RTN.PSETID=ID;                                         DEF00360
                                                                         DEF00370
 CALL TRTN(PROC_ADDR, RTP);                                              DEF00380
 RETURN;                                                                 DEF00390
 END /*DEFP*/;                                                           DEF00400
```

158

```
DEL:PROC;                                                              DEL00010
                                                                       DEL00020
/*T-PROC AT THE MM LEVEL FOR DELETING A SEU FROM SH*****/              DEL00030
/* WILL SET INVAL AT ID*/                                             DEL00040
/* FOR INTEGRATED VERSION: MAKE USE OF SUBROUTINES PSEU, GSEU */       DEL00050
                                                                       DEL00060
%INCLUDE ADEL,DSEU1,SERVICE,EGSEU,EPSEU;                               DEL00070
DCL (P,RTP) PTR, (NULL,UNSPEC) BUILTIN, RTN_CODE FIXED BIN;            DEL00080
                                                                       DEL00090
CALL TBEG (PROC_ADDR,P);                                               DEL00100
ALLOCATE DEL_RTN SET (RTP);   RTP->DEL_RTN.RTN_CODE=0;                 DEL00110
CALL GSEU(DEL_ARG.ID, SEUPTR,RTN_CODE);   /* GET SEU TO BE DELETED */  DEL00120
IF RTN_CODE = 1                           /* INVALID */                DEL00130
THEN RTP->DEL_RTN.RTN_CODE=1;                                          DEL00140
ELSE DO;                                                               DEL00150
   SEU.INVAL = 'Y';                                                    DEL00160
   CALL PSEU (DEL_ARG.ID, SEUPTR, RTN_CODE); /* SET SEU INVAL FLAG */  DEL00170
   END;                                                                DEL00180
                                                                       DEL00190
/*RETURNS*/                                                            DEL00200
CALL TRTN(PROC_ADDR, RTP);                                             DEL00210
RETURN;                                                                DEL00220
END /*DEL*/;                                                           DEL00230
```

```
                                                                        DEL00010
DELB: PROC (PARENT, BSETID,ID1,MODE,RTN_CODE);                          DEL00020
                                                                        DEL00030
/*A S-PROC AT THE N-ARY LEVEL TO DELETE A BINARY ASSOCIATION*/          DEL00040
                                                                        DEL00050
%INCLUDE DBEU,DBCAT;                                                    DEL00060
%INCLUDE EBCT,ESUB1,EDELP;                                              DEL00070
                                                                        DEL00080
DCL (PARENT,BSETID,ID1,ID2) BIT (32), P PTR, MODE CHAR (1);             DEL00090
DCL (NULL,UNSPEC) BUILTIN;                                              DEL00100
DCL RTN_CODE FIXED BIN;                                                 DEL00110
                                                                        DEL00120
CALL GBCT (BSETID,BINFO,RTN_CODE);                                      DEL00130
IF RTN_CODE ^= 0 THEN RETURN; /* 0+ IS FROM GBCT */                     DEL00140
CALL RET1 (ID1,P,RTN_CODE);                                             DEL00150
IF P=NULL THEN DO;                                                      DEL00160
  RTN_CODE=10; /***10 IS BAD SOURCE ELEM ********/                      DEL00170
  RETURN; END;                                                          DEL00180
                                                                        DEL00190
ID2=BEU.ID_ARRAY(BINFO.POS);                                            DEL00200
IF ID2 ^= UNSPEC (NULL) THEN DO; /* IF THERE EXISTS A BIN ASSOC ELEM */ DEL00210
                                                                        DEL00220
BEU.ID_ARRAY(BINFO.POS)=UNSPEC (NULL); /*DELETE ARC*/                   DEL00230
                                                                        DEL00240
SELECT (MODE);                                                          DEL00250
                                                                        DEL00260
 WHEN ('N')  /*NODE MODE DELETES TARGET NODE TOO*/                      DEL00270
   IF BINFO.FUNC='S' THEN CALL DELP(BINFO.PSETID(2),ID2,RTN_CODE);      DEL00280
                                                                        DEL00290
 OTHERWISE;                                                             DEL00300
 END /*SELECT*/;                                                        DEL00310
 IF RTN_CODE ^= 0 THEN RTN_CODE = 20 + RTN_CODE; /* 20 + IS RTNCODE    DEL00320
                    FROM DELP ************************/                DEL00330
END /* IF THERE EXISTS... */;                                          DEL00340
RETURN;                                                                 DEL00350
END /*DELB*/;                                                           DEL00360
```

```
DELN: PROC(MP,OFF_NODE,RTN_CODE);                                          DEL00010
                                                                           DEL00020
/*A S-PROC AT THE N-ARY LEVEL USED AS A SUBROUTINE CALLED BY THE           DEL00030
*UPDN PROGRAM TO PERFORM DEE OP AT ROOT NODE*/                             DEL00040
                                                                           DEL00050
%INCLUDE AUPDN;                                                            DEL00060
%INCLUDE EDELB,ESRCH,EDELP;                                                DEL00070
                                                                           DEL00080
DCL (MP,P) PTR, (N,RTN_CODE) FIXED BIN,                                    DEL00090
  OFF_NODE(25) BIT (1) CONNECTED,                                          DEL00100
  DATA BIT (320) VARYING, (PS,ID,BS) BIT (32), (SUBSTR,NULL)BUILTIN;       DEL00110
                                                                           DEL00120
/*DO P=MP REPEAT UPDN_ARG.PTR WHILE(P^=NULL);                              DEL00130
   PUT SKIP LIST('DELN ARG LIST',UPDN_ARG); END;  */                       DEL00140
                                                                           DEL00150
P=MP;                                                                      DEL00160
/*LOCATE ROOT NODE ID*/                                                    DEL00170
PS=UPDN_ARG.PSETID;                                                        DEL00180
DATA=SUBSTR(UPDN_ARG.DATA,1,UPDN_ARG.DLEN * 8);                            DEL00190
CALL SRCH(PS,DATA,ID,RTN_CODE);                                            DEL00200
                                                                           DEL00210
/*FOLLOW THE TREE*/                                                        DEL00220
DO P=UPDN_ARG.PTR REPEAT UPDN_ARG.PTR WHILE (P^=NULL);                     DEL00230
   IF OFF_NODE(UPDN_ARG.NODE)='0'B THEN DO;                                DEL00240
      BS=UPDN_ARG.BSETID;                                                  DEL00250
      SELECT (UPDN_ARG.OP);                                                DEL00260
                                                                           DEL00270
        WHEN (UPDNOP.DEL) CALL DELB (PS,BS,ID,'A',RTN_CODE);               DEL00280
        WHEN (UPDNOP.DD) CALL DELB (PS,BS,ID,'N',RTN_CODE);                DEL00290
                                                                           DEL00300
       END /*SELECT*/;                                                     DEL00310
       END /*DO IF*/;                                                      DEL00320
    END /*DO P LOOP */;                                                    DEL00330
                                                                           DEL00340
 IF RTN_CODE ^= 0 THEN RETURN;/* RTN_CODE SET TO RTN FROM SBR CALLS */     DEL00350
                                                                           DEL00360
 /* NOW DELETE THE ROOT NODE */                                            DEL00370
 CALL DELP (PS,ID,RTN_CODE);                                               DEL00380
 IF RTN_CODE ^= 0 THEN RTN_CODE = 50 + RTN_CODE; /* 50+ FROM DELP ***/    DEL00390
                                                                           DEL00400
  END /*DELN*/;                                                            DEL00410
```

```
DELP: PROC (PSETID,ID,RTN_CODE);                                        DEL00010
                                                                        DEL00020
/* A S-PROC AT THE N-ARY LEVEL WHICH DELETES A P ELEMENT GIVEN ITS      DEL00030
* ID WITHIN A PSETID*/                                                  DEL00040
                                                                        DEL00050
%INCLUDE DPCAT,DBEU;                                                    DEL00060
%INCLUDE EPCT,ESUB1;                                                    DEL00070
                                                                        DEL00080
DCL (PSETID,ID) BIT (32), RTN_CODE FIXED BIN, P PTR;                    DEL00090
DCL (NULL,UNSPEC) BUILTIN, (PREV_ID,POST_ID) BIT (32);                  DEL00100
                                                                        DEL00110
CALL GPCT (PSETID,PINFO,RTN_CODE); /*FIRST OBTAINS PSET CAT*/           DEL00120
                                                                        DEL00130
CALL RET1 (ID,P,RTN_CODE); /*OBTAIN ELEM TO BE DELETED*/                DEL00140
IF P=NULL THEN DO;                                                      DEL00150
  RTN_CODE=1; /*BAD ELEM, MIGHT ALREADY BEEN DELETED*/                  DEL00160
 RETURN; END;                                                           DEL00170
                                                                        DEL00180
PREV_ID=BEU.ID_ARRAY(PINFO.L_POS2); /*REMEMBER NEIGHBORS IN CHAIN*/     DEL00190
POST_ID=BEU.ID_ARRAY(PINFO.L_POS);                                      DEL00200
                                                                        DEL00210
CALL DEL1(ID,RTN_CODE); /*DELETE IT*/                                   DEL00220
FREE BEU;                                                               DEL00230
                                                                        DEL00240
/*MODIFY PTR CHAIN WITHIN THE PSET*/                                    DEL00250
IF PREV_ID ^= UNSPEC(NULL) THEN DO;                                     DEL00260
CALL RET1 (PREV_ID,P,RTN_CODE);                                         DEL00270
BEU.ID_ARRAY(PINFO.L_POS)=POST_ID;                                      DEL00280
CALL REP1 (PREV_ID,P,RTN_CODE);                                         DEL00290
FREE BEU;                                                               DEL00300
END;                                                                    DEL00310
ELSE DO; /* IF PREV_ID IS NULL THEN ELEM BEING DELETED IS FIRST IN      DEL00320
      PSET CHAIN, THEREFORE NEEDS TO UPDATE PCAT ENTRY */               DEL00330
PINFO.L_ID = POST_ID;                                                   DEL00340
CALL UPCT (PSETID,PINFO,RTN_CODE);                                      DEL00350
END;                                                                    DEL00360
                                                                        DEL00370
IF POST_ID ^= UNSPEC(NULL) THEN DO;                                     DEL00380
CALL RET1(POST_ID,P,RTN_CODE);                                          DEL00390
BEU.ID_ARRAY(PINFO.L_POS2)=PREV_ID;                                     DEL00400
CALL REP1 (POST_ID,P,RTN_CODE);                                         DEL00410
FREE BEU;                                                               DEL00420
END;                                                                    DEL00430
                                                                        DEL00440
RTN_CODE=0; RETURN;                                                     DEL00450
END /*DELP*/;                                                           DEL00460
```

```
DEL1: PROC (ID,RTN_CODE);                                                DUM00010
                                                                         DUM00020
/*A S-PROC AT THE LOWEST LAYER OF N-ARY LEVEL TO SET UP CALL TO          DUM00030
* DEL MODULE AT THE MM LEVEL TO DELETE BEU DESIGNATED BY ID*/            DUM00040
                                                                         DUM00050
%INCLUDE SERVICE,ADEL,FDEBUG;                                            DUM00060
DCL (P,TP) PTR, NULL BUILTIN, RTN_CODE FIXED BIN, ID BIT (32);           DUM00070
                                                                         DUM00080
ALLOCATE DEL_ARG;                                                        DUM00090
DEL_ARG.ID=ID;                                                           DUM00100
CALL TCALL ('DEL',1,P,TP);                                               DUM00110
                                                                         DUM00120
/***************************** MEMORY REQUEST TRACE DUMP **********/     DUM00130
IF FFF THEN CALL DUMPIT ('DELETED',ID,NULL);                  /****/     DUM00140
DCL FFF BIT (1) EXT;                                          /****/     DUM00150
/*****************************************************************/     DUM00160
                                                                         DUM00170
RTN_CODE=TP->DEL_RTN.RTN_CODE;                                           DUM00180
CALL SVC3(P);                                                            DUM00190
CALL SVC3(TP);                                                           DUM00200
RETURN;                                                                  DUM00210
END /*DEL1*/;                                                           DUM00220
```

```
DFB1:PROC(FUNC, PSETID1,PSETID2,RTN_CODE,BSETID);                    DFB00010
   /*A S-PROC AT THE ENTITY LEVEL FOR SETTING UP CALLS TO DEFB*/     DFB00020
                                                                     DFB00030
   %INCLUDE SERVICE,ADEFB,FDEBUG;                                    DFB00040
                                                                     DFB00050
   DCL NULL BUILTIN, (P,TP)PTR;                                      DFB00060
   DCL FUNC CHAR (1),(PSETID1,PSETID2)BIT(32);                       DFB00070
   DCL RTN_CODE FIXED BIN, BSETID BIT(32);                           DFB00080
                                                                     DFB00090
   ALLOCATE DEFB_ARG;                                                DFB00100
   DEFB_ARG.PSETID(1)=PSETID1;                                       DFB00110
   DEFB_ARG.PSETID(2)=PSETID2;                                       DFB00120
   DEFB_ARG.FUNC=FUNC;                                               DFB00130
   IF FUNC='K' THEN DEFB_ARG.FUNC='S';                               DFB00140
                                                                     DFB00150
   CALL TCALL ('DEFB',1,P,TP);                                       DFB00160
   RTN_CODE=TP->DEFB_RTN.RTN_CODE;                                   DFB00170
   IF RTN_CODE=0 THEN BSETID=TP->DEFB_RTN.BSETID;                    DFB00180
                                                                     DFB00190
   CALL SVC3 (P);                                                    DFB00200
  CALL SVC3 (TP);                                                    DFB00210
  RETURN;                                                            DFB00220
  END /*DFB1*/;                                                      DFB00230
```

```
DFP1:PROC ( PTYPE,PLEN,TYPE,RTN_CODE,PSETID);                          DFP00010
/*********************************************************************  DFP00020
*                                                                    *  DFP00030
*                    MODULE    DESCRIPTION                           *  DFP00040
*********************************************************************/ DFP00050
/*********************************************************************  DFP00060
*****    PURPOSE: THIS IS THE SUBROUTINE USED AT THE ENTITY LEVEL      DFP00070
*****     TO SET UP CALL TO DEFINE ONE PSET.                           DFP00080
*********************************************************************  DFP00090
*****    INPUT PARAMETERS: PTYPE, CHAR(1), IS THE DATA TYPE OF THE     DFP00100
*****    PSET TO BE DEFINED;  PLEN, FIXED BIN, IS THE LENGTH;          DFP00110
*****     TYPE, CHAR(1), INDICATES 'E' OR'V', TO BE USED BY THE        DFP00120
*****     INTERNAL ROUTINE IMPP FOR DETERMINING IMP PORTION OF         DFP00130
*****     DEFP_ARG;                                                    DFP00140
*********************************************************************  DFP00150
*****    OUTPUT PARAMETERS:                                            DFP00160
*****     RTN_CODE AND PSETID AS OBTAINED FROM DEFP CALL;              DFP00170
*********************************************************************  DFP00180
*****    CALLS PROCEDURES:                                             DFP00190
*****     INTER-LEVEL T-PROC: DEFP;                                  */ DFP00200
%INCLUDE ADEFP;                                                        DFP00210
/****     INTRA-LEVEL T-PROC:   NONE;                                   DFP00220
*****     INTRA-LEVEL S-PROC:   NONE;                                  DFP00230
*****     CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES: */ DFP00240
%INCLUDE SERVICE,FDEBUG;                                               DFP00250
/*********************************************************************/ DFP00260
                                                                       DFP00270
DCL (P,TP,RP)PTR, NULL BUILTIN;                                        DFP00280
DCL RTN_CODE FIXED BIN;                                                DFP00290
DCL PTYPE CHAR(1), PLEN FIXED BIN, TYPE CHAR(1);                       DFP00300
DCL PSETID BIT(32);                                                    DFP00310
/*BEGIN PROCESSING*/                                                   DFP00320
                                                                       DFP00330
   ALLOCATE DEFP_ARG;   RP=P; /*ROOT PTR*/                             DFP00340
   DEFP_ARG.PTYPE=PTYPE;                                               DFP00350
   DEFP_ARG.PLEN=PLEN;                                                 DFP00360
   CALL IMPP; /*IMPP COMPLETES THE IMP PORTION OF DEFP_ARG*/           DFP00370
                                                                       DFP00380
   /*SET UP MSG TO CALL DEFP*/                                         DFP00390
  CALL TCALL ('DEFP',1,RP,TP);                                         DFP00400
                                                                       DFP00410
   CALL SVC3(RP); /*FREE CALLING ARG LIST*/                            DFP00420
   /*EXTRACT RTN_CODE AND PSETID*/                                     DFP00430
   P=TP;                                                               DFP00440
   RTN_CODE=TP->DEFP_RTN.RTN_CODE;                                     DFP00450
   IF FDEBUG.LEV1 THEN DO;                                             DFP00460
     IF DEFP_RTN.RTN_CODE^=0 THEN DO;                                  DFP00470
    PUT SKIP LIST ('DFP1: DEFP RTN CODE NOT 0, IS',DEFP_RTN.RTN_CODE); DFP00480
    RTN_CODE=0;                                                        DFP00490
    END;                                                               DFP00500
    END;                                                               DFP00510
   PSETID=TP->DEFP_RTN.PSETID;                                         DFP00520
   CALL SVC3 (TP); /*FREE UP RTN MSG LIST*/                            DFP00530
     RETURN;                                                           DFP00540
IMPP:PROC; /*INTERNAL SUBROUTINE FOR ASSIGNING IMP PORTION OF DEFP*/   DFP00550
```

```
      /*NOTE THAT THIS ROUTINE INTERFACES WITH IMPLEMENTAION          DFP00560
        OF THE N-ARY LEVEL, THEREFORE SUBJECT TO CHANGE IF           DFP00570
        DESIGN OF N-ARY LEVEL CHANGES*/                              DFP00580
      IF TYPE='E' THEN DO;                                           DFP00590
        DEFP_ARG.IMP.HOW=DEFPHOW.SA; /*STAND ALONE*/                 DFP00600
        DEFP_ARG.IMP.NUMPTR=12;/*DEFAULT TO12*/                      DFP00610
        END;                                                         DFP00620
     ELSE DO;                                                        DFP00630
        DEFP_ARG.IMP.HOW=DEFPHOW.SA; /*ALSO STAND ALONE*/            DFP00640
        DEFP_ARG.IMP.NUMPTR=4; /*DEFAULT TO 4*/                      DFP00650
        END;                                                         DFP00660
    END /*IMPP*/;                                                    DFP00670
  END /*DFP1*/;                                                      DFP00680
```

```
DMB: PROC;                                                              DMB00010
DCL CMD CHAR(7);                                                        DMB00020
                                                                        DMB00030
PUT SKIP LIST('-- DATA MANIPULATION SESSION --');                       DMB00040
DO WHILE ('1'B); /*DO FOREVER*/                                         DMB00050
  PUT SKIP LIST ('DM: ENTER MANIPULATION COMMAND:');                    DMB00060
  PUT SKIP LIST('CREATE(CRT), MODIFY(MOD), DELETE(DEL), QUERY(QUE)');   DMB00070
  GET EDIT (CMD) (A(8));                                                DMB00080
  SELECT (CMD);                                                         DMB00090
    WHEN ('CREATE','CRT','C') CALL DMC;                                 DMB00100
    WHEN ('MODIFY','MOD','M') CALL DMM;                                 DMB00110
    WHEN ('DELETE','DEL','D') CALL DMD;                                 DMB00120
    WHEN ('QUERY','QUE','Q') CALL DMQ;                                  DMB00130
    WHEN ((8)' ')  LEAVE;                                               DMB00140
  OTHERWISE PUT SKIP LIST('ILLEGAL COMMAND,(PLEASE USE UPPER CASE)');   DMB00150
  END /*SELECT*/;                                                       DMB00160
 END /*DO WHILE*/;                                                      DMB00170
                                                                        DMB00180
 END /*DM*/;                                                            DMB00190
```

L91

```
DMC: PROC; /*PROCESSING CREATE COMMANDS*/                                DMC00010
%INCLUDE ELEX,ELEXT,EVNM1,EUPE1;                                         DMC00020
DCL 1 T,   /*DATA STRUCTURE OF NAMES */                                  DMC00030
      2 ENAME CHAR(8),                                                   DMC00040
      2 TN FIXED BIN, /*NUMBER OF ATTR*/                                 DMC00050
      2 T1 (25),                                                         DMC00060
        3 PARENT FIXED BIN,                                              DMC00070
        3 ANAME CHAR (8);                                                DMC00080
DCL RTN_CODE FIXED BIN;                                                  DMC00090
DCL (OP(25), OPE) CHAR(1);                                               DMC00100
DCL  N FIXED BIN;  /*THE FOLLOWING USED TO CALL LEX*/                    DMC00110
DCL 1 TOKEN(25),                                                         DMC00120
      2 D CHAR(40),                                                      DMC00130
      2 L FIXED BIN;                                                     DMC00140
DCL MODE FIXED BIN;                                                      DMC00150
DCL N1 FIXED BIN;    /*USED TO COMM. WITH VNM1*/                         DMC00160
DCL 1 PRED(25),/* STRUCTURE TO COMMUNICATE WITH LEXT*/                   DMC00170
      2 OP CHAR (1), /* =,>,<, */                                        DMC00180
      2 VALUE CHAR (40),                                                 DMC00190
      2 VLEN FIXED BIN;                                                  DMC00200
                                                                         DMC00210
DO WHILE ('1'B); /*DO FOREVER*/                                          DMC00220
A1:PUT SKIP LIST ('CREATE: ENTER ENTITY SET NAME');                      DMC00230
   CALL LEX (N,TOKEN,1,RTN_CODE);                                        DMC00240
   IF N = 0  THEN LEAVE; /*GET OUT OF DMC*/                              DMC00250
   T.ENAME = TOKEN.D(1);                                                 DMC00260
                                                                         DMC00270
 A:PUT SKIP LIST ('ENTER NAMES OF ATTRIBUTES SEPARATED BY COMMA');       DMC00280
   MODE=1; /*SUPPRESS BLANKS*/                                           DMC00290
   CALL LEX (N, TOKEN, MODE,RTN_CODE);/*GET INPUT LINE THRU LEX*/        DMC00300
   IF RTN_CODE^=0 THEN DO;                                               DMC00310
      PUT SKIP LIST ('IMPROPER SYNTAX (LEX), PLEASE RE-ENTER');          DMC00320
      GO TO A;                                                           DMC00330
      END;                                                               DMC00340
                                                                         DMC00350
   IF N=0 THEN LEAVE; /*GET OUT OF DMC*/                                 DMC00360
                                                                         DMC00370
   CALL LEXT (N,TOKEN,T1,T.TN,PRED,OP,RTN_CODE);/*BUILD T STRUCTURE*/    DMC00380
      IF RTN_CODE^=0 THEN DO;                                            DMC00390
         PUT SKIP LIST ('IMPROPER SYNTAX, PLEASE RE-ENTER');             DMC00400
         GO TO A;                                                        DMC00410
         END;                                                            DMC00420
                                                                         DMC00430
   /*CALL UP VNME TO CHECK FOR LEGALITY OF NAMES*/                       DMC00440
   CALL VNM1(T,RTN_CODE,N1);       /* VNM1 WILL PRINT ERR MSG IF ANY*/  DMC00450
   IF N1=-1 THEN GO TO A1;         /* ESET NAME ERROR               */  DMC00460
   ELSE IF N1 ^= 0 THEN GO TO A;  /* ATTR NAME ERROR               */   DMC00470
                                                                         DMC00480
   /*START ASKING FOR DATA THROUGH SUBROUTINE UPE1*/                     DMC00490
   OPE='C';                                                              DMC00500
   DO I=1 TO 25; OP(I)='N'; /*INITIALIZE OPS TO BE 'IST' */ END;         DMC00510
   PUT SKIP LIST ('ENTER DATA:');                                        DMC00520
                                                                         DMC00530
   CALL UPE1 (OPE,T.OP,RTN_CODE);  /*UPE1 WILL PRINT ERR MSG IF ANY */  DMC00540
   END /*DO WHILE*/;                                                     DMC00550
```

```
RETURN;                                                    DMC00560
END /*DMC*/;                                               DMC00570
```

```
DMD: PROC; /*PROCESSING DELETE COMMANDS*/                              DMD00010
%INCLUDE ELEX,ELEXT,EVNM1,EUPE1;                                       DMD00020
DCL 1 T,   /*DATA STRUCTURE USED TO COMMUNICATE WITH LEXT*/            DMD00030
      2 ENAME CHAR(8),                                                 DMD00040
      2 TN FIXED BIN, /*NUMBER OF ATTR*/                               DMD00050
      2 T1 (25),                                                       DMD00060
        3 PARENT FIXED BIN,                                            DMD00070
        3 ANAME CHAR (8);                                              DMD00080
DCL RTN_CODE FIXED BIN;                                                DMD00090
DCL (OP(25), OPE) CHAR(1);                                             DMD00100
DCL  N FIXED BIN;  /*THE FOLLOWING USED TO CALL LEX*/                  DMD00110
DCL 1 TOKEN(25),                                                       DMD00120
      2 D CHAR(40),                                                    DMD00130
      2 L FIXED BIN;                                                   DMD00140
DCL MODE FIXED BIN;                                                    DMD00150
DCL N1 FIXED BIN;    /*USED TO COMM. WITH VNM1*/                       DMD00160
DCL 1 PRED(25),/* STRUCTURE TO COMMUNICATE WITH LEXT*/                 DMD00170
      2 OP CHAR (1), /* =,>,<, */                                      DMD00180
      2 VALUE CHAR (40),                                               DMD00190
      2 VLEN FIXED BIN;                                                DMD00200
DCL CMD CHAR (8);                                                      DMD00210
                                                                       DMD00220
PUT SKIP LIST ('TYPE YES IF NEED HELP');                               DMD00230
GET EDIT (CMD) (A(8)); IF CMD='YES' THEN CALL HELP;                    DMD00240
                                                                       DMD00250
DO WHILE ('1'B); /*DO FOREVER*/                                        DMD00260
A1:PUT SKIP LIST ('DELETE: ENTER ENTITY SET NAME');                    DMD00270
   CALL LEX (N, TOKEN, 1, RTN_CODE);                                   DMD00280
    IF N = 0  THEN LEAVE; /*GET OUT OF DMD*/                           DMD00290
   T.ENAME = TOKEN.D (1);                                              DMD00300
                                                                       DMD00310
 A:PUT SKIP LIST ('ENTER IDENTIFIER ATTRIBUTE NAMES. SEPARATE BY COMMA'DMD00320
);                                                                     DMD00330
   MODE=1; /*SUPPRESS BLANKS*/                                         DMD00340
   CALL LEX (N, TOKEN, MODE,RTN_CODE);/*GET INPUT LINE THRU LEX*/      DMD00350
   IF RTN_CODE^=0 THEN DO;                                             DMD00360
      PUT SKIP LIST ('IMPROPER SYNTAX (LEX), PLEASE RE-ENTER');        DMD00370
      GO TO A;                                                         DMD00380
       END;                                                            DMD00390
                                                                       DMD00400
   IF N=0 THEN LEAVE; /*GET OUT OF DMD*/                               DMD00410
                                                                       DMD00420
   CALL LEXT (N,TOKEN,T1,T.TN,PRED,OP,RTN_CODE);/*BUILD T STRUCTURE*/  DMD00430
   IF RTN_CODE^=0 THEN DO;                                             DMD00440
      PUT SKIP LIST ('IMPROPER SYNTAX, PLEASE RE-ENTER');              DMD00450
      GO TO A;                                                         DMD00460
      END;                                                             DMD00470
                                                                       DMD00480
   /*CALL UP VNME TO CHECK FOR LEGALITY OF NAMES*/                     DMD00490
   CALL VNM1(T,RTN_CODE,N1);                                           DMD00500
   IF N1=-1 THEN GO TO A1;      /* ESET NAME ERROR       */           DMD00510
   ELSE IF N1 ^= 0 THEN GO TO A; /* ATTR NAME ERROR      */           DMD00520
                                                                       DMD00530
   /*START ASKING FOR DATA THROUGH SUBROUTINE UPE1*/                   DMD00540
   OPE='D';                                                            DMD00550
```

```
   PUT SKIP LIST ('ENTER DATA:');                                          DMD00560
                                                                           DMD00570
   DO I = 1 TO 25; OP(I) = 'I'; END; /*INITIALIZE ALL OPS TO ID */         DMD00580
                                                                           DMD00590
   CALL UPE1 (OPE,T,OP,RTN_CODE);  /*UPE1 WILL PRINT ERR MSG IF ANY */    DMD00600
                                                                           DMD00610
   END /*DO WHILE*/;                                                       DMD00620
RETURN;                                                                    DMD00630
                                                                           DMD00640
/*********************************************************************/   DMD00650
HELP: PROC; /**** INTERNAL SUBROUTINE *********/                          DMD00660
PUT SKIP LIST ('********** BRIEF EXPLANATION ON HOW TO DO DELETE ***');   DMD00670
PUT SKIP LIST ('AT THE PROMPT FOR IDENTIFIER ATTRIBUTE NAMES PLEASE');     DMD00680
PUT SKIP LIST ('ENTER ATTRIBUTE NAMES WHOSE VALUE WILL BE GIVEN TO');      DMD00690
PUT SKIP LIST ('IDENTIFY THE ENTITY TO BE DELETE.');                       DMD00700
PUT SKIP LIST ('AN EXAMPLE FOLLOWS:');                                     DMD00710
PUT SKIP LIST('TO DELETE THE EMPLOYEE WITH A CERTAIN EMPLOYEE NUMBER');    DMD00720
PUT SKIP LIST ('FIRST ISSUE THE FOLLOWING AT THE PROMPT FOR');             DMD00730
PUT SKIP LIST ('IDENTIFIER ATTRIBUTES:');                                  DMD00740
PUT SKIP LIST ('EMPNUM');                                                  DMD00750
PUT SKIP LIST('AND LATER ENTER EMPNUM AT THE PROMT FOR DATA');             DMD00760
PUT SKIP LIST ('********** NOW TRY IT ***************************');       DMD00770
END /*HELP*/;                                                              DMD00780
/*********************************************************************/   DMD00790
                                                                           DMD00800
END /*DMD*/;                                                              DMD00810
```

```
DMM: PROC; /*PROCESSING CREATE COMMANDS*/                                DUM00010
/*************************************************************************DUM00020
*                                                                       *DUM00030
*                    MODULE    DESCRIPTION                            *DUM00040
*************************************************************************/DUM00050
/*************************************************************************DUM00060
*****   PURPOSE:                                                        DUM00070
*****    THIS IS A S-PROC AT THE USER INTERFACE LEVEL, CALLED BY        DUM00080
*****    MODULE DMB, TO PROCESS USER DATA MODIFICATION COMMAND          DUM00090
*****    WITHIN A USER DATA MANIPULATION - MODIFICATION SESSION.        DUM00100
*************************************************************************DUM00110
*****   METHOD:                                                         DUM00120
*****     1. INPUT ENTITY SET NAME IN T.ENAME.                          DUM00130
*****     2. CALL LEX TO INPUT AND PARSE ATTRIBUTE LIST INTO TOKENS.    DUM00140
*****     3. CALL LEXT TO BUILD NAME TREE AND OP TABLE.                 DUM00150
*****     4. CALL VNM1 TO SET UP CALL TO VNME AT THE NEXT LEVEL         DUM00160
*****        TO VERIFY NAMES.                                           DUM00170
*****     5. CALL UPE1 TO GET DATA FROM USER AND SET UP CALL TO UPDE.   DUM00180
*************************************************************************DUM00190
*****   CALLS PROCEDURES:                                               DUM00200
*****    INTER-LEVEL T-PROC: NONE                                       DUM00210
*****    INTRA-LEVEL T-PROC: NONE                                       DUM00220
*****    INTRA-LEVEL S-PROC: LEX,LEXT,VNM1,UPE1.                        DUM00230
*****    CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES:     DUM00240
*****                        NONE.                                      DUM00250
*************************************************************************/DUM00260
%INCLUDE ELEX,ELEXT,EVNM1,EUPE1;                                         DUM00270
DCL 1 T,   /*DATA STRUCTURE USED TO COMMUNICATE WITH LEXT*/             DUM00280
     2 ENAME CHAR(8),                                                    DUM00290
     2 TN FIXED BIN,             /*NUMBER OF ATTR*/                     DUM00300
     2 T1 (25),                                                          DUM00310
       3 PARENT FIXED BIN,                                               DUM00320
       3 ANAME CHAR (8);                                                 DUM00330
DCL RTN_CODE FIXED BIN;                                                  DUM00340
DCL (OP(25), OPE) CHAR(1);                                               DUM00350
DCL  N FIXED BIN;    /*THE FOLLOWING USED TO CALL LEX*/                  DUM00360
DCL 1 TOKEN(25),                                                         DUM00370
     2 D CHAR(40),                                                       DUM00380
     2 L FIXED BIN;                                                      DUM00390
DCL MODE FIXED BIN;                                                      DUM00400
DCL N1 FIXED BIN;    /*USED TO COMM. WITH VNM1*/                         DUM00410
DCL 1 PRED(25),/* STRUCTURE TO COMMUNICATE WITH LEXT*/                   DUM00420
     2 OP CHAR (1), /* =,>,<, */                                         DUM00430
     2 VALUE CHAR (40),                                                  DUM00440
     2 VLEN FIXED BIN;                                                   DUM00450
 DCL CMD CHAR (8);                                                       DUM00460
                                                                         DUM00470
 PUT SKIP LIST ('TYPE YES IF NEED HELP');                                DUM00480
 GET EDIT (CMD) (A(8)); IF CMD='YES' THEN CALL HELP;                     DUM00490
                                                                         DUM00500
 DO WHILE ('1'B); /*DO FOREVER*/                                         DUM00510
 A1:PUT SKIP LIST ('MODIFY: ENTER ENTITY SET NAME');                     DUM00520
    CALL LEX (N,TOKEN,1,RTN_CODE);                                       DUM00530
    IF N = 0  THEN LEAVE; /*GET OUT OF DMC*/                             DUM00540
    T.ENAME = TOKEN.D(1);                                                DUM00550
```

```
                                                                        DUM00560
  A:PUT SKIP LIST ('ENTER MODIFICATION OPERATORS AND NAMES OF ATTRIBUTEDUM00570
S SEPARATED BY COMMAS');                                                DUM00580
    MODE=1; /*SUPPRESS BLANKS*/                                         DUM00590
    CALL LEX (N, TOKEN, MODE,RTN_CODE);/*GET INPUT LINE THRU LEX*/      DUM00600
    IF RTN_CODE^=0 THEN DO;                                             DUM00610
       PUT SKIP LIST ('IMPROPER SYNTAX (LEX), PLEASE RE-ENTER');        DUM00620
       GO TO A;                                                         DUM00630
       END;                                                             DUM00640
                                                                        DUM00650
    IF N=0 THEN LEAVE; /*GET OUT OF DMC*/                               DUM00660
                                                                        DUM00670
    CALL LEXT (N,TOKEN,T1,T.TN,PRED,OP,RTN_CODE);/*BUILD T STRUCTURE*/  DUM00680
      IF RTN_CODE^=0 THEN DO;                                           DUM00690
         PUT SKIP LIST ('IMPROPER SYNTAX, PLEASE RE-ENTER');            DUM00700
         GO TO A;                                                       DUM00710
         END;                                                           DUM00720
                                                                        DUM00730
   /*CALL UP VNME TO CHECK FOR LEGALITY OF NAMES*/                      DUM00740
   CALL VNM1(T,RTN_CODE,N1);      /* VNM1 WILL PRINT ERR MSG IF ANY */ DUM00750
   IF N1 = -1 THEN GO TO A1;              /* ESET NAME ERROR        */  DUM00760
   ELSE IF N1 ^= 0 THEN GO TO A;          /* ATTR NAME ERROR        */  DUM00770
                                                                        DUM00780
   /*START ASKING FOR DATA THROUGH SUBROUTINE UPE1*/                    DUM00790
   OPE='M';                                                             DUM00800
   PUT SKIP LIST ('ENTER DATA:');                                       DUM00810
                                                                        DUM00820
   CALL UPE1 (OPE,T,OP,RTN_CODE);       /* UPE1 WILL PRINT ERR MSG */  DUM00830
   END /*DO WHILE*/;                                                   DUM00840
RETURN;                                                                 DUM00850
                                                                        DUM00860
/*********************************************************************/ DUM00870
HELP: PROC; /**** INTERNAL SUBROUTINE *********/                        DUM00880
PUT SKIP LIST ('********** BRIEF EXPLANATION ON HOW TO DO MODIFY ******DUM00890
****');                                                                 DUM00900
PUT SKIP LIST ('AT THE PROMPT FOR MODIFICATION OPERATORS AND ATTRI-'); DUM00910
PUT SKIP LIST ('BUTE NAMES PLEASE ENTER THE OPERATOR (INSERT,DELETE,');DUM00920
PUT SKIP LIST ('REPLACE) PREFIXED WITH "-", AND NAMES OF ATTRIBUTES'); DUM00930
PUT SKIP LIST ('THAT WILL BE THUS MODIFIED.  YOU SHOULD ALSO PROVIDE');DUM00940
PUT SKIP LIST ('A SPECIAL OPERATOR (ID) FOR NAMES OF ATTRIBUTES USED');DUM00950
PUT SKIP LIST ('TO IDENTIFY THE ENTITY TO BE MODIFIED. AN EXAMPLES');  DUM00960
PUT SKIP LIST ('FOLLOWS:');                                            DUM00970
PUT SKIP LIST ('TO CHANGE THE DEPTARTMENT AN EMPLOYEE (WHOSE');        DUM00980
PUT SKIP LIST ('EMPLOYEE NUMBER WILL BE GIVEN) WORKS IN TO ANOTHER');  DUM00990
PUT SKIP LIST ('DEPARTMENT (WHOSE DEPARTMENT NUMBER WILL BE GIVEN),'); DUM01000
PUT SKIP LIST ('ISSUE THE FOLLOWING MODIFICATION STATEMENT AT THE');    DUM01010
PUT SKIP LIST ('PROMPT FOR ATTRIBUTE NAMES AND MODIFICATION');         DUM01020
PUT SKIP LIST ('OPERATORS:');                                          DUM01030
PUT SKIP LIST ('-ID:EMPNUM,-REP:WORKS_IN(DEPTNUM)');                   DUM01040
PUT SKIP LIST ('AND LATER ENTER DATA FOR EMPNUM AND DEPTNUM.');        DUM01050
PUT SKIP LIST ('********** NOW TRY IT ****************************');   DUM01060
END /*HELP*/;                                                          DUM01070
   END /*DMM*/;                                                        DUM01080
```

```
DMQ: PROC;                                                              DUM00010
/*************************************************************************DUM00020
*                                                                        *DUM00030
*                    MODULE    DESCRIPTION                               *DUM00040
*************************************************************************/DUM00050
/*************************************************************************DUM00060
*****   PURPOSE:  THIS IS A                                              DUM00070
** S-PROC AT THE USER INTERFACE LEVEL, CALLED BY THE DMB MODULE, TO      DUM00080
** PROCESS QUERY REQUESTS ISSUED BY THE USERS WITHIN A DATA              DUM00090
** MANIPULATION - QUERY SESSION.                                        DUM00100
*************************************************************************DUM00110
*****   METHOD:                                                          DUM00120
*****     1. CALLS LEX TO PARSE INPUT STRING INTO TOKENS.                DUM00130
*****     2. CALLS LEXT TO BUILD NAME TREE AND PREDICATE TABLE.          DUM00140
*****     3. SET UP RETE CALL TO THE NEXT LEVEL.                         DUM00150
*****     4. CALL PRNT TO PRINT RESULT.                                  DUM00160
*****                                                                    DUM00170
*************************************************************************DUM00180
*****   CALLS PROCEDURES:                                                DUM00190
*****    INTER-LEVEL T-PROC: RETE.                                       DUM00200
*****    INTRA-LEVEL T-PROC: NONE.                                       DUM00210
*****    INTRA-LEVEL S-PROC: LEX,LEXT,PRNT                               DUM00220
*****    CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES:      DUM00230
*****          TCALL,SVC3                                                DUM00240
*************************************************************************/DUM00250
                                                                        DUM00260
                                                                        DUM00270
 %INCLUDE ELEX,ELEXT,SERVICE,ARETE,EPRNT;                               DUM00280
 DCL 1 T,   /*DATA STRUCTURE USED TO COMMUNICATE WITH LEXT*/           DUM00290
      2 ENAME CHAR(8),                                                  DUM00300
      2 TN FIXED BIN, /*NUMBER OF ATTR*/                                DUM00310
      2 T1 (25),                                                        DUM00320
         3 PARENT FIXED BIN,                                            DUM00330
         3 ANAME CHAR (8);                                              DUM00340
 DCL RTN_CODE FIXED BIN;                                                DUM00350
 DCL  N FIXED BIN;  /*THE FOLLOWING USED TO CALL LEX*/                  DUM00360
 DCL 1 TOKEN(25),                                                       DUM00370
       2 D CHAR(40),                                                    DUM00380
       2 L FIXED BIN;                                                   DUM00390
 DCL MODE FIXED BIN;                                                    DUM00400
 DCL N1 FIXED BIN;              /*USED TO COMM. WITH VNM1*/             DUM00410
 DCL 1 PRED(25),       /* PRED AND MOP COMMUNICATE WITH LEXT*/         DUM00420
       2 OP CHAR (1), /* =,>,<, */                                      DUM00430
       2 VALUE CHAR (40),                                               DUM00440
       2 VLEN FIXED BIN;                                                DUM00450
 DCL MOP (25) CHAR (1);/* NOT USED IN DMQ  */                           DUM00460
 DCL CMD CHAR (8);                                                      DUM00470
 DCL (TP,P,RP,LP) PTR, NULL BUILTIN ;                                   DUM00480
 DCL I FIXED BIN (31);                                                  DUM00490
                                                                        DUM00500
 /*BEGIN PROCESSING*/                                                   DUM00510
                                                                        DUM00520
 PUT SKIP LIST ('--QUERY SESSION--');                                   DUM00530
 PUT SKIP LIST ('TYPE YES IF NEED HELP');                               DUM00540
 GET EDIT (CMD) (A(8));                                                 DUM00550
```

```
  IF CMD='YES' THEN CALL HELP;                                          DUM00560
                                                                        DUM00570
 /* MAIN COURSE */                                                      DUM00580
  DO WHILE ('1'B); /*DO FOREVER*/                                       DUM00590
  A1:PUT SKIP LIST ('ENTER ENTITY SET NAME');                           DUM00600
     CALL LEX (N,TOKEN,1,RTN_CODE);                                     DUM00610
     IF N = 0 THEN LEAVE; /*GET OUT OF DMC*/                            DUM00620
     T.ENAME = TOKEN.D (1);                                             DUM00630
                                                                        DUM00640
   A:PUT SKIP LIST ('ENTER ATTRIBUTE NAMES AND PREDICATE, SEPARATED BY  DUM00650
COMMAS');                                                               DUM00660
     MODE=1; /*SUPPRESS BLANKS*/                                        DUM00670
     CALL LEX (N, TOKEN, MODE,RTN_CODE);/*GET INPUT LINE THRU LEX*/     DUM00680
      IF RTN_CODE^=0 THEN DO;                                           DUM00690
        PUT SKIP LIST ('IMPROPER SYNTAX (LEX), PLEASE RE-ENTER');       DUM00700
        GO TO A;                                                        DUM00710
        END;                                                            DUM00720
                                                                        DUM00730
     IF N=0 THEN LEAVE; /*GET OUT OF DMQ*/                              DUM00740
                                                                        DUM00750
     CALL LEXT (N,TOKEN,T1,T.TN,PRED,MOP,RTN_CODE); /*BUILD T STRUC*/   DUM00760
       IF RTN_CODE^=0 THEN DO;                                          DUM00770
          PUT SKIP LIST ('IMPROPER SYNTAX, PLEASE RE-ENTER');           DUM00780
          GO TO A;                                                      DUM00790
          END;                                                          DUM00800
                                                                        DUM00810
 /* SET UP CALL TO RETE */                                             DUM00820
                                                                        DUM00830
 ALLOCATE RETE_ARG SET (RP); /*ROOT NODE */                             DUM00840
 LP=RP;                                                                 DUM00850
 RP->RETE_ARG.NAME=T.ENAME;                                             DUM00860
 RP->RETE_ARG.GET =RETEGET.ALL;                                         DUM00870
 RP->RETE_ARG.N   =0; /*NO PREDICATE */                                 DUM00880
                                                                        DUM00890
 DO I=1 TO T.TN; /* ATTRIBUTE NODES */                                  DUM00900
    ALLOCATE RETE_ARG; /* P->RETE_ARG */                               DUM00910
    LP->RETE_ARG.PTR=P; LP=P;                                           DUM00920
    RETE_ARG.NAME=T.ANAME(I);                                           DUM00930
    RETE_ARG.PARENT=T.PARENT(I)+1;                                      DUM00940
    RETE_ARG.NODE=I+1;                                                  DUM00950
    RETE_ARG.GET= RETEGET.ANY;                                          DUM00960
                                                                        DUM00970
    /* PREDICATE */                                                     DUM00980
    IF PRED.VLEN(I)^=0 THEN DO;                                         DUM00990
       RETE_ARG.N = 1;                                                  DUM01000
       SELECT (PRED.OP(I));                                             DUM01010
          WHEN ('>') RETE_ARG.OP = RETEOP.GT;                           DUM01020
          WHEN ('<') RETE_ARG.OP = RETEOP.LT;                           DUM01030
          WHEN ('=') RETE_ARG.OP = RETEOP.EQ;                           DUM01040
          OTHERWISE;                                                    DUM01050
       END /* SELECT */;                                               DUM01060
       RETE_ARG.CDATA = PRED.VALUE(I);                                  DUM01070
       RETE_ARG.DLEN = PRED.VLEN(I);                                    DUM01080
     END /* PREDICATE */;                                              DUM01090
  END /*DO I */;                                                        DUM01100
```

```
                                                                        DUMO1110
 CALL TCALL ('RETE',1,RP,TP); /* TP-> RETE_RTN */                       DUMO1120
 IF TP->RETE_RTN.RTN_CODE ^= 0 /* CHECK RETE RETURN CODE */             DUMO1130
 THEN DO;                                                               DUMO1140
    CALL RETEMSG (TP->RETE_RTN.RTN_CODE);                               DUMO1150
    CALL SVC3 (RP); CALL SVC3 (TP);                                     DUMO1160
    GO TO A1;                                                           DUMO1170
    END;                                                                DUMO1180
                                                                        DUMO1190
 IF TP->RETE_RTN.N = 0 THEN DO;                                         DUMO1200
    PUT SKIP LIST ('NO DATA FOUND');                                    DUMO1210
    CALL SVC3 (RP); CALL SVC3 (TP);                                     DUMO1220
    GO TO A1;                                                           DUMO1230
    END;                                                                DUMO1240
                                                                        DUMO1250
 CALL PRNT (RP,TP,RTN_CODE);                                            DUMO1260
 CALL SVC3 (RP);  CALL SVC3 (TP);                                       DUMO1270
 END /*DO WHILE */;                                                     DUMO1280
                                                                        DUMO1290
 /*************** INTERNAL SUBROUTINES *********************/         DUMO1300
 HELP: PROC; /*INTERNAL SUBROUTINE */                                   DUMO1310
 PUT SKIP LIST ('******* BRIEF EXPLANATION ON HOW TO USE QUERY *******  DUMO1320
');                                                                     DUMO1330
 PUT SKIP LIST ('THE SYSTEM WILL ASK YOU TO PROVIDE THE NAMES');        DUMO1340
 PUT SKIP LIST ('OF THE ENTITY SET YOU ARE TO QUERY ABOUT,');           DUMO1350
 PUT SKIP LIST ('AND THE LIST OF ATTRIBUTES OF THIS ENTITY SET');       DUMO1360
 PUT SKIP LIST ('OF INTEREST TO YOU. YOU MAY ALSO PROVIDE PREDICATES');DUMO1370
 PUT SKIP LIST ('ALONG WITH THE LIST OF ATTRIBUTES.  ATTRIBUTES IN');   DUMO1380
 PUT SKIP LIST ('YOUR ATTRIBUTE LIST SHOULD BE SEPARATED BY COMMAS.');  DUMO1390
 PUT SKIP LIST ('THE FOLLOWING IS AN EXAMPLE OF AN ATTRIBUTE LIST');    DUMO1400
 PUT SKIP LIST ('PROVIDED BY A USER QUERYING ABOUT AN ENTITY SET');     DUMO1410
 PUT SKIP LIST ('EMPLOYEE.  THIS USER IS INTERESTED IN THE EMPNUM');    DUMO1420
 PUT SKIP LIST ('AND EMPNAME OF EMPLOYEES OLDER THAN 56 AND WORK IN');  DUMO1430
 PUT SKIP LIST ('DEPT WITH DEPTNUM 15.');                               DUMO1440
 PUT SKIP LIST ('EMPNUM,EMPNAME,AGE>56,DEPT(DEPTNUM=15)');              DUMO1450
 PUT SKIP LIST ('******** NOW TRY IT ******************************');  DUMO1460
 END /* HELP */;                                                       DUMO1470
                                                                        DUMO1480
/*********************************************************************/ DUMO1490
RETEMSG: PROC (CODE);        /* INTERNAL SUBROUTINE FOR RETE ERROR */  DUMO1500
DCL CODE FIXED BIN;                                                     DUMO1510
IF CODE = 100 THEN PUT SKIP LIST ('NO SUCH ENTITY SET.');               DUMO1520
                                                                        DUMO1530
IF CODE >100 & CODE <200                                                DUMO1540
THEN PUT SKIP EDIT ('ILLEGAL ATTRIBUTE ',T.ANAME(CODE-101))(A,A);       DUMO1550
                                                                        DUMO1560
IF CODE < 100                                                           DUMO1570
THEN PUT SKIP EDIT ('ILLEGAL PREDICATE DATA FOR ',T.ANAME(CODE-101))    DUMO1580
                   (A,A);                                               DUMO1590
                                                                        DUMO1600
IF CODE > 200 THEN PUT SKIP EDIT ('RETN RETURN CODE ',CODE)(A,F(3));    DUMO1610
                                                                        DUMO1620
PUT SKIP LIST ('PROBLEM IN QUERY STATEMENT: PLEASE REISSUE');           DUMO1630
RETURN;                                                                 DUMO1640
END;                                                                    DUMO1650
```

```
/*****************************************************************/  DUMO1660
                                                                     DUMO1670
END /* DMQ */;                                                      DUMO1680
```

```
FSTV: PROC;                                                          DUM00010
/*** A DRIVER PROGRAM USED WHEN FH IS RUN WITHOUT CONTROL STRUCTURE */ DUM00020
/*********************************************************************** DUM00030
*                                                                     * DUM00040
*                    MODULE    DESCRIPTION                            * DUM00050
***********************************************************************/ DUM00060
/*********************************************************************** DUM00070
*****   PURPOSE:  A T-PROC FOR CONSOLE INITIALIZATION FROM             DUM00080
*****     AN EXISTING FILE OR FROM SCRATCH, AND CONSOLE LOGOUT.        DUM00090
*********************************************************************** DUM00100
*****   METHOD:                                                        DUM00110
*****   IT ASKS FOR INIT METHOD FROM THE CONSOLE USER.  IF FILE INIT,  DUM00120
*****   IT ASKS FOR FILE NAME AND PERCOLATES THE FILE INIT DOWN. IF    DUM00130
*****   NEW INIT, IT CALLS SBRS TO INITIALIZE CAT FILES OF THIS LEVEL  DUM00140
*****   AND PERCOLATES THE NEW INIT DOWN.                              DUM00150
*****                                                                  DUM00160
*********************************************************************** DUM00170
*********************************************************************** DUM00180
*****   CALLS PROCEDURES:  RINIT, CINIT, VINIT, VSAVE, USER;           DUM00190
***** (INTRA-LEVEL S-PROCS: RINIT, CINIT: USED FOR INITIALIZING THE    DUM00200
*****   RELATIONAL VIEW PROCESSOR AND THE BN VIEW PROCESSOR, BOTH      DUM00210
*****   ARE CURRENTLY INCLUDED IN THE TOP LEVEL BUT EVENTUALLY WILL    DUM00220
*****   BE DEDICATED TO THE VTL, WHICH DOES NOT EXIST YET.             DUM00230
*****   USER: ENTER INTO USER MODE FROM CONSOLE.                       DUM00240
*****  INTER-LEVEL T-PROC: VINIT, VSAVE: USED FOR PERCOLATING INIT     DUM00250
*****   AND SAVE COMMANDS TO THE NEXT LEVEL.)                          DUM00260
*****                                                                  DUM00270
***********************************************************************/ DUM00280
                                                                       DUM00290
                                                                       DUM00300
                                                                       DUM00310
/*%INCLUDE  EUSER;   /*INTRA-LEVEL S-PROC ENTRY DCLS*/                 DUM00320
%INCLUDE AVINI ;  /*INTER-LEVEL T-PROC ARG LIST*/                      DUM00330
%INCLUDE SERVICE; /*CONTROL STRUCTURE SERVICES*/                       DUM00340
%INCLUDE FDEBUG,SH;                                                    DUM00350
DCL SETFH ENTRY;                                                       DUM00360
ON ZERODIVIDE;                                                         DUM00370
DCL NULL BUILTIN;                                                      DUM00380
DCL RTN_CODE FIXED BIN(15);                                            DUM00390
DCL (TP,P) POINTER;                                                    DUM00400
DCL (NAME, FNAME) CHAR(8);                                             DUM00410
DCL SAVESH FILE RECORD BUFFERED SEQUENTIAL ENV                         DUM00420
      (U BLKSIZE(30016) );                                             DUM00430
DCL OUT BIT(1) INIT ('0'B);                                            DUM00440
                                                                       DUM00450
CALL SETFH;                 /* SETS UP TRACE OPTIONS */                DUM00460
ALLOCATE SH;                                                           DUM00470
                                                                       DUM00480
/* BEGIN SESSION  */                                                   DUM00490
PUT SKIP (2) LIST ('--FUNCTIONAL HIERARCHY STV CONSOLE PROGRAM--');    DUM00500
                                                                       DUM00510
DO WHILE ('1'B);                                                       DUM00520
   PUT SKIP LIST ('INITIALIZATION: FILE OR NEW:');                     DUM00530
   GET EDIT (NAME) (A(8));                                             DUM00540
                                                                       DUM00550
```

```
    SELECT(NAME);                                                          DUM00560
                                                                           DUM00570
    WHEN ( 'F' , 'FILE','f','file') DO;                                    DUM00580
      OUT='1'B;                                                            DUM00590
      PUT SKIP LIST ('FILE NAME?');                                        DUM00600
      GET EDIT  (FNAME) (A(8));                                            DUM00610
      CALL READ (FNAME); /*READ IS INT SBR TO READ FILE INTO SH*/          DUM00620
      CALL NEXTI;/*INT SUBR TO SET UP CALL FOR FILE INIT AT NEXT LV*/      DUM00630
      END;                                                                 DUM00640
                                                                           DUM00650
    WHEN ( 'N' , 'NEW','n','new') DO;  /* NEW INIT*/                       DUM00660
      ALLOCATE PACKET IN (SH);                                             DUM00670
      OUT='1'B;                                                            DUM00680
      CALL RINIT;  /*REL VIEW PROCESSOR INIT*/                           DUM00690
      CALL CINIT;  /*BN VIEW PROCESSOR INIT*/                            DUM00700
      FNAME=(8) ' ';                                                       DUM00710
      CALL NEXTI;/*INT SBR TO SET UP CALL FOR NEW INIT IN NEXT LV*/        DUM00720
      END;                                                                 DUM00730
                                                                           DUM00740
      OTHERWISE PUT SKIP EDIT(NAME,' IS NOT A COMMAND')(A,A);              DUM00750
                                                                           DUM00760
    END;                                                                   DUM00770
    IF OUT THEN LEAVE;                                                     DUM00780
                                                                           DUM00790
END;                                                                       DUM00800
/*AFTER INITIALIZATION, CONSOLE MAY CHOOSE TO ENTER USER MODE*/            DUM00810
CALL TCALL ('USER',1,TP,P);                                                DUM00820
                                                                           DUM00830
/*CONSOLE LOG OUT AND SAVE FILE*/                                          DUM00840
PUT SKIP LIST ('SAVE FILE: FILE NAME?');                                   DUM00850
GET EDIT (FNAME) (A(8));                                                   DUM00860
IF FNAME ^= '' THEN CALL SAVE(FNAME);                                      DUM00870
/*SAVE IS INT SBR TO SET UP CALL TO NEXT LEV FOR SAVING DATA*/             DUM00880
                                                                           DUM00890
/***************** TIMING REPORT OPTION ***************************/      DUM00900
 ON ZERODIVIDE;                                                            DUM00910
 DCL ((ETIME,ELO,ECOUNT)(50) FIXED BIN(31)) EXT;                           DUM00920
 DCL FTIME BIT(1) EXT;                                                     DUM00930
 %INCLUDE FSTRUC;                                                          DUM00940
 IF FTIME THEN DO;                                                         DUM00950
 PUT EDIT ('LEVEL  PROCNAME  COUNT  TOT ELAPSED TIME  TOT RUN TIME  RUN   DUM00960
 TIME/INVOCATION')(SKIP,A);                                                DUM00970
 PUT EDIT ('-----  --------  -----  ---------------  ------------  ---     DUM00980
 ---------------')(SKIP,A);                                               DUM00990
 PUT EDIT(((ARCH.LEVEL(I),ARCH.PROCNAME(I),ECOUNT(I),ETIME(I),             DUM01000
                      (ETIME(I)-ELO(I)),                                   DUM01010
                      (ETIME(I)-ELO(I))/ECOUNT(I))                         DUM01020
          DO I=1 TO 19))                                                   DUM01030
          (SKIP,F(3),X(4),A,F(7),F(17),F(13),F(21));                       DUM01040
 END;                                                                      DUM01050
/*****************************************************************/      DUM01060
/*                                                                        DUM01070
CALL FINISH; /*SUICIDE****/                                                DUM01080
FREE SH;                                                                   DUM01090
                                                                           DUM01100
```

```
/***INTERNAL SUBROUTINES**********************/                    DUMO1110
NEXTI: PROC;  /* INT SBR FOR PASSING INIT DOWN NEXT LEVEL*/        DUMO1120
  ALLOCATE VINIT_ARG;  /*SET UP ARG LIST*/                         DUMO1130
  VINIT_ARG.PTR=NULL;                                             DUMO1140
  VINIT_ARG.FNAME=FNAME;                                          DUMO1150
  IF FDEBUG.LEVO THEN PUT SKIP LIST ('FSTV CALLS VINIT');         DUMO1160
  CALL TCALL ('VINIT',1,P,TP);                                    DUMO1170
  IF FDEBUG.LEVO THEN PUT SKIP LIST ('FSTV: VINIT RETURNS');      DUMO1180
  CALL SVC3 (P);                                                  DUMO1190
  IF TP->VINIT_RTN.RTN_CODE ^=0 THEN DO;                          DUMO1200
    PUT SKIP LIST ('ERROR IN INITIALIZATION, RESTART');           DUMO1210
    OUT='0'B;  /*REST INIT LOOP*/                                 DUMO1220
    END;                                                          DUMO1230
  CALL SVC3 (TP);                                                 DUMO1240
  END /*NEXTI*/;                                                  DUMO1250
                                                                  DUMO1260
/*****************************************/                       DUMO1270
SAVE:PROC(FNAME); /* SAVE EVERYTHING IN STORAGE HIERARCHY*/       DUMO1280
   DCL FNAME CHAR(8);                                             DUMO1290
   OPEN FILE(SAVESH) TITLE(FNAME) OUTPUT;                         DUMO1300
   WRITE FILE(SAVESH) FROM(SH);                                   DUMO1310
   WRITE FILE(SAVESH) FROM(PP);      /* OFFSET OF PACKET IN SH */ DUMO1320
   CLOSE FILE(SAVESH);                                            DUMO1330
  RETURN;                                                         DUMO1340
  END /* SAVE */;                                                 DUMO1350
                                                                  DUMO1360
READ: PROC(FNAME);/*SBR TO READ FILE INTO STORAGE HIERARCHY*/     DUMO1370
   DCL FNAME CHAR(8);                                             DUMO1380
   OPEN FILE(SAVESH) TITLE(FNAME) INPUT;                          DUMO1390
   READ FILE(SAVESH) INTO(SH);                                    DUMO1400
   READ FILE(SAVESH) INTO(PP);                                    DUMO1410
   CLOSE FILE(SAVESH);                                            DUMO1420
  RETURN;                                                         DUMO1430
 END;                                                             DUMO1440
                                                                  DUMO1450
 /*************************************************************/  DUMO1460
CINIT: PROC;                                                      DUMO1470
 /*IN THE FUTURE, THIS MODULE INITIALIZES THE BASE DATA MODEL VIEW DUMO1480
   SUBSYSTEM; PRESENTLY IT IS NOT AVAILABLE*/                     DUMO1490
 RETURN;                                                          DUMO1500
 END;                                                             DUMO1510
                                                                  DUMO1520
RINIT: PROC;                                                      DUMO1530
 /*IN THE FUTURE, THIS MODULE INITIALIZES THE RELATIONAL  VIEW    DUMO1540
   SUBSYSTEM; PRESENTLY IT IS NOT AVAILABLE*/                     DUMO1550
 RETURN;                                                          DUMO1560
 END;                                                             DUMO1570
                                                                  DUMO1580
                                                                  DUMO1590
END /*FSTV*/;                                                     DUMO1600
```

```
L1:PROC(ARGPTR);  /* A DUMMY ROUTINE TO SIMULATE ENTIRE SH-STV */      DUM00010
/* USED WHEN RUN WITHOUT CONTROL STRUCUTRE AND DSH */                  DUM00020
%INCLUDE ARG.SH;                                                       DUM00030
DCL INDX FIXED BIN(31);                                                DUM00040
                                                                       DUM00050
INDX = ARG.VIRTUAL_ADDRESS/8 + 1;                                      DUM00060
IF ARG.REQ_TYPE = 'READ'                                               DUM00070
THEN ARG.DATA = PACKET(INDX);                                          DUM00080
ELSE PACKET(INDX) = ARG.DATA;                                          DUM00090
RETURN;                                                                DUM00100
END;                                                                   DUM00110
```

```
DUMPIT:PROC (STRING,ID,P);                                                    DUM00010
%INCLUDE DBEU;                                                                DUM00020
DCL STRING CHAR(*);                                                           DUM00030
DCL ID BIT(32); DCL (P,BP BASED) PTR;                                         DUM00040
DCL I FIXED BIN;                                                              DUM00050
DCL PAA(40) PTR BASED;                                                        DUM00060
DCL NULL BUILTIN;                                                             DUM00070
%INCLUDE HEX;                                                                 DUM00080
                                                                              DUM00090
PUT SKIP EDIT('***MEMMGT: ',STRING,' ID=',HEX(ADDR(ID)->BP))(A,A,A,A);        DUM00100
IF P = NULL THEN GO TO END;                                                   DUM00110
                                                                              DUM00120
PUT EDIT('NUMPTR=',NUMPTR)(X(1),A,F(2));                                      DUM00130
DO I=1 TO NUMPTR;                                                             DUM00140
   PUT EDIT(HEX(ADDR(ID_ARRAY(I))->BP))(X(1),A); END;                         DUM00150
PUT EDIT('DLEN=',DLEN)(X(1),A,F(3));                                          DUM00160
DO I = 1 TO (DLEN+31)/32; PUT EDIT(HEX4(ADDR(DATA)->PAA(I)))(A);END;          DUM00170
END;                                                                          DUM00180
END;                                                                          DUM00190
```

```
 TBEG: PROC (PROC_ADDR,P);                                              DUM00010
                                                                        DUM00020
  /**** A SERVICE ROUTINE TO STRIF OFF MSG TOKEN FROM A CALLING ***     DUM00030
   *** ARGUMENT LIST; IT IS USED BY A T-PROC WHEN IT IS INVOKED ***/    DUM00040
                                                                        DUM00050
 /**** PROC_ADDR CONTAINS THE PROC_ADDR OF THE CALLING PROCEDURE, ***   DUM00060
  ***  P WILL PONT TO THE FIRST TOKEN OF THE CALLING ARGUMENT LIST. */  DUM00070
  DCL STAT_P1 PTR EXTERNAL STATIC;                                      DUM00080
  DCL P PTR;                                                            DUM00090
                                                                        DUM00100
 P= STAT_P1;                                                            DUM00110
 RETURN;                                                                DUM00120
 END;                                                                   DUM00130
                                                                        DUM00140
```

```
 (SUBRG):                                                              DUM00010
TCALL:PROC (PROCNAME,I,P1,P2) RECURSIVE ;                              DUM00020
                                                                       DUM00030
 /*THIS IS A DUMMY TCALL TO HANDLE PROBLEM OF CONTROL STRUCTURE*/       DUM00040
 /* IT IS USED TO INSULATE FH MODULES FROM CONTROL STRUCTURE */         DUM00050
                                                                       DUM00060
DCL FTCALL(4) BIT(1) STATIC EXT;     /* FTCALL IS DEBUG TRACE BIT */   DUM00070
DCL FTIME BIT(1) EXT;                /*  FTIME IS TIMING TRACE BIT*/   DUM00080
%INCLUDE FSTRUC;                    /* FSTRUC IS STRUCTURE OF FH */    DUM00090
DCL PROCNAME CHAR(7) ;                                                 DUM00100
DCL (VINIT,DEFE,RETE,UPDE,DEFP,DEFB,RETN) ENTRY;                       DUM00110
DCL (UPDN,DEFA, NINIT,VNME,SHWE) ENTRY;                                DUM00120
DCL (MINIT,RET,REP,DEL,CRT,FSTV,USER) ENTRY;                           DUM00130
DCL I FIXED BIN, (STAT_P1,STAT_P2) PTR EXTERNAL STATIC;                DUM00140
DCL (P1,P2)PTR;  DCL (BEGTIME,ENUM,LOTIME CTL) FIXED BIN(31);          DUM00150
DCL RTIMER ENTRY RETURNS(FIXED BIN(31));                               DUM00160
DCL (ECOUNT(50),ELO(50),ETIME(50)) FIXED BIN(31) EXT INIT((50)0);      DUM00170
STAT_P1=P1;                                                            DUM00180
                                                                       DUM00190
DO ENUM = 1 TO 18;                                  /* FIND LEVEL     */ DUM00200
   IF ARCH.PROCNAME(ENUM) = PROCNAME THEN LEAVE;                       DUM00210
   END;      -                                                         DUM00220
I=ARCH.LEVEL(ENUM);                                                    DUM00230
                                                                       DUM00240
/***************** TCALL TRACE ***********************************/    DUM00250
IF FTCALL(I)            /* IF TCALL TRACE BIT FOR THAT LEVEL IS ON*/   DUM00260
THEN PUT SKIP EDIT ('TCALL: ',PROCNAME)(A,A);                          DUM00270
/*****************************************************************/    DUM00280
                                                                       DUM00290
/***************** TIMING REPORT *********************************/    DUM00300
IF FTIME THEN DO;                                                      DUM00310
ALLOCATE LOTIME; LOTIME=0;                                             DUM00320
BEGTIME=RTIMER;END;                                                    DUM00330
/*****************************************************************/    DUM00340
  SELECT (PROCNAME);                                                   DUM00350
    WHEN ( 'FSTV    ') CALL FSTV;                                      DUM00360
    WHEN ( 'VINIT   ')   CALL VINIT;                                   DUM00370
    WHEN ( 'VNME    ' )   CALL VNME;                                   DUM00380
    WHEN ( 'DEFE    ' )   CALL DEFE;                                   DUM00390
    WHEN ( 'RETE    ' )   CALL RETE;                                   DUM00400
    WHEN ( 'UPDE    ' )   CALL UPDE;                                   DUM00410
    WHEN ( 'DEFP    ' )   CALL DEFP;                                   DUM00420
    WHEN ( 'SHWE    ') CALL SHWE;                                      DUM00430
    WHEN ( 'DEFB    ' )   CALL DEFB;                                   DUM00440
    WHEN ( 'RETN    ' )   CALL RETN;                                   DUM00450
    WHEN ( 'UPDN    ' )   CALL UPDN;                                   DUM00460
    WHEN ( 'DEFA    ') CALL DEFA;                                      DUM00470
    WHEN ( 'NINIT   ') CALL NINIT;                                     DUM00480
    WHEN ('RET      ') CALL RET;                                       DUM00490
    WHEN ('DEL      ') CALL DEL;                                       DUM00500
    WHEN ('CRT      ') CALL CRT;                                       DUM00510
    WHEN ('REP      ') CALL REP;                                       DUM00520
    WHEN ('MINIT    ') CALL MINIT;                                     DUM00530
    WHEN ('USER     ') CALL USER;                                      DUM00540
                                                                       DUM00550
```

```
     OTHERWISE DO;PUT SKIP EDIT (' *** NAME NOT FOUND') (A);          DUM00560
                  SIGNAL ERROR; END;                                  DUM00570
      END;                                                            DUM00580
                                                                      DUM00590
     /*WHEN RETURNED*/                                               DUM00600
 /***************** TIMING REPORT ******************************/     DUM00610
  IF FTIME THEN DO;                                                   DUM00620
  BEGTIME=RTIMER-BEGTIME;                                             DUM00630
  ETIME(ENUM)=ETIME(ENUM)+BEGTIME;                                    DUM00640
  ECOUNT(ENUM)=ECOUNT(ENUM)+1;                                        DUM00650
  ELO(ENUM)=ELO(ENUM)+LOTIME;                                         DUM00660
  FREE LOTIME; IF ALLOCATION(LOTIME)^= 0 THEN LOTIME=LOTIME+BEGTIME;  DUM00670
  END;                                                                DUM00680
  /*****************************************************************/ DUM00690
                                                                      DUM00700
 /************* TCALL TRACE ************************/                 DUM00710
    IF FTCALL(I) THEN                                                 DUM00720
    PUT EDIT(' ***TCALL: END ',PROCNAME)(A,A);                        DUM00730
 /**************************************************/                DUM00740
    P2=STAT_P2;                                                       DUM00750
                                                                      DUM00760
 END /* TCALL*/;                                                     DUM00770
```

581

```
TRTN: PROC(PROC_ADDR, RTP);                                          DUM00010
                                                                     DUM00020
/**** A SERVICE ROUTINE USED BY A T-PROC WHEN IT FINISHES PROCESSING  DUM00030
 **** IT ADDS A MSG TOKEN TO A RETURN ARGUMENT LIST AND CALLS SVCS2   DUM00040
***    AND THEN FINISHES ITSELF ****/                                DUM00050
                                                                     DUM00060
/*** PROC_ADDR CONTAINS THE ADDRS OF THE CALLING PROCEDURE TO RETURN  DUM00070
 *** THE RETURN MSG TO; RTP POINTS TO THE RETURN ARGUMENT LIST ***/   DUM00080
                                                                     DUM00090
                                                                     DUM00100
 %INCLUDE FDEBUG;                                                    DUM00110
 DCL (P,RTP) PTR, RTN_CODE FIXED BIN;                                DUM00120
 DCL 1 T BASED (P),                                                  DUM00130
     2 LEN FIXED BIN (15);                                           DUM00140
 DCL STAT_P2 PTR EXTERNAL STATIC;                                    DUM00150
 STAT_P2=RTP;                                                        DUM00160
                                                                     DUM00170
                                                                     DUM00180
 END /*TRTN*/;                                                       DUM00190
```

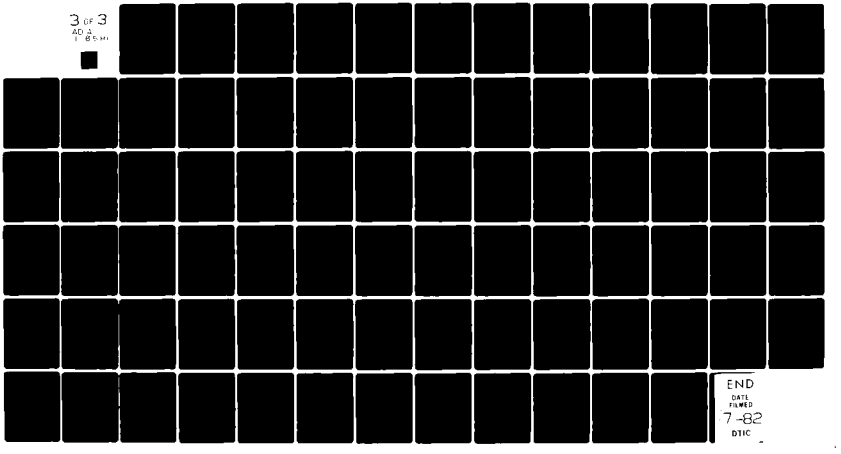
```
EBOT:PROC;                                                          EB000010
 /*A S-PROC AT THE ENTITY LEVEL USED BY VINIT MODULE TO BOOTSRAP     EB000020
  ** ENTITY CATALOGUE ENTRIES ****************************/         EB000030
 %INCLUDE AUPDN,SERVICE, DKEY,FDEBUG;                               EB000040
 DCL (STRING, UNSPEC )BUILTIN;                                      EB000050
 DCL (I,K,J) FIXED BIN, (LP,RP,TP,P)PTR, NULL BUILTIN;              EB000060
 DCL 1 MAINFO (5), /*AN ARRAY OF TEMPLATE  AINFO*/                  EB000070
      2 BSETID BIT (32),  /*BSETID OF THIS ATTRIBUTE**/              EB000080
      2 PSETID BIT (32),  /*PSETID OF TARGET NODE**/                 EB000090
      2 ENAME CHAR (8) INIT (' ',' ','E*ESET'),                     EB000100
      2 MAX FIXED BIN (31),                                         EB000110
      2 MIN FIXED BIN (31),                                         EB000120
      2 MLEN FIXED BIN INIT (8, 4,0,8,29),                          EB000130
      2 FUNC CHAR(1) INIT ('K','S','M','S','S'),                    EB000140
      2 TYPE CHAR (1) INIT('V','V','E','V','V'),                    EB000150
      2 VTYPE CHAR (1) INIT ('C','B',' ','C','B');                  EB000160
 DCL BIT_STRING BIT(232) BASED; /*BIT STRING FORM OF AINFO*/        EB000170
 DCL BAINFO(5) BIT(232);                                            EB000180
 DCL SEQ(5) FIXED BIN INIT (2,3,1,5,6);                             EB000190
 DCL ADDR BUILTIN;                                                  EB000200
                                                                    EB000210
 /*BEGIN PROCESSING*/                                               EB000220
 /*SET UP BAINFO*/                                                  EB000230
   DO I=1 TO 5;                                                     EB000240
    MAINFO.BSETID(I)=KEY(I+6);                                      EB000250
    MAINFO.PSETID(I)=KEY(SEQ(I));                                   EB000260
    BAINFO(I)=ADDR(MAINFO(I))->BIT_STRING;                          EB000270
    END;                                                            EB000280
                                                                    EB000290
DO J=1 TO 2; /*FOR TWO ENTITY SETS ESET AND ASET*/                  EB000300
                                                                    EB000310
   ALLOCATE UPDN_ARG; LP,RP=P;                                      EB000320
   UPDN_ARG.PSETID=KEY(KY.PESET);                                   EB000330
   UPDN_ARG.OP=UPDNOP.CRT;                                          EB000340
   UPDN_ARG.DLEN=0;                                                 EB000350
                                                                    EB000360
   DO I=1 TO 2;                                                     EB000370
      ALLOCATE UPDN_ARG;                                            EB000380
      LP->UPDN_ARG.PTR=P; LP=P;                                     EB000390
      UPDN_ARG.PARENT=1;                                            EB000400
      UPDN_ARG.OP=UPDNOP.CI; /*CI IS CREATE AND INSERT*/            EB000410
      UPDN_ARG.NODE=I+1;                                            EB000420
                                                                    EB000430
   SELECT (I);                                                      EB000440
      WHEN (1) DO;                                                  EB000450
          UPDN_ARG.PSETID=KEY(KY.PENAME);                           EB000460
          UPDN_ARG.BSETID=KEY(KY.BENAME);                           EB000470
          UPDN_ARG.DLEN=8;                                          EB000480
          IF J=1 THEN UPDN_ARG.DATA=UNSPEC ('E*ESET  ' );           EB000490
             ELSE UPDN_ARG.DATA=UNSPEC ('E*ASET  ');                EB000500
          END /*WHEN 1*/;                                           EB000510
                                                                    EB000520
     WHEN (2) DO;                                                   EB000530
          UPDN_ARG.PSETID=KEY(KY.PEINFO);                           EB000540
          UPDN_ARG.BSETID=KEY(KY.BEINFO);                           EB000550
```

```
          UPDN_ARG.DLEN=4;                                              EB000560
          IF J=1 THEN UPDN_ARG.DATA=KEY(KY.PESET);                      EB000570
                 ELSE UPDN_ARG.DATA=KEY(KY.PASET);                      EB000580
          END /*WHEN 2*/ ;                                              EB000590
                                                                        EB000600
       END /*SELECT I*/;                                                EB000610
                                                                        EB000620
   END /*DO I*/;                                                        EB000630
   CALL TCALL ('UPDN',1,RP, TP);  /*CALL UP UPDN*/                      EB000640
   END /*DO J*/;                                                        EB000650
                                                                        EB000660
   DO J=1 TO 5; /*5 ATTRIBUTE SETS IN THE CATALOGUE*/                   EB000670
      ALLOCATE UPDN_ARG; RP,LP=P;                                       EB000680
      UPDN_ARG.PSETID=KEY(KY.PASET);                                    EB000690
      UPDN_ARG.OP=UPDNOP.CRT;                                           EB000700
      UPDN_ARG.DLEN=0;                                                  EB000710
                                                                        EB000720
      DO I=1 TO 4;                                                      EB000730
         ALLOCATE UPDN_ARG;                                             EB000740
         LP->UPDN_ARG.PTR=P; LP=P;                                      EB000750
         UPDN_ARG.NODE=I+1; /*ASSIGN NODE NUMBER*/                      EB000760
         SELECT (I);                                                    EB000770
                                                                        EB000780
           WHEN (1) DO; /*ANAME NODE*/                                  EB000790
             UPDN_ARG.PSETID=KEY(KY.PANAME);                            EB000800
             UPDN_ARG.PARENT=1;                                         EB000810
             UPDN_ARG.BSETID=KEY(KY.BANAME);                            EB000820
             UPDN_ARG.OP=UPDNOP.CI;                                     EB000830
             UPDN_ARG.DLEN=8;                                           EB000840
             SELECT (J);                                                EB000850
               WHEN (1) UPDN_ARG.DATA=UNSPEC('A*ENAME ');               EB000860
               WHEN (2) UPDN_ARG.DATA=UNSPEC('A*EINFO ');               EB000870
               WHEN (3) UPDN_ARG.DATA=UNSPEC('A*ESET  ');               EB000880
               WHEN (4) UPDN_ARG.DATA=UNSPEC('A*ANAME ');               EB000890
               WHEN (5) UPDN_ARG.DATA=UNSPEC('A*AINFO ');               EB000900
             END /*SELECT J*/;                                          EB000910
         END /*WHEN 1*/;                                                EB000920
                                                                        EB000930
         WHEN (2) DO; /*ESET NODE*/                                     EB000940
             UPDN_ARG.PSETID=KEY(KY.PESET);                             EB000950
             UPDN_ARG.PARENT=1;                                         EB000960
             UPDN_ARG.BSETID=KEY(KY.BESET);                             EB000970
             UPDN_ARG.OP=UPDNOP.IST;                                    EB000980
             UPDN_ARG.DLEN=0;                                           EB000990
             END /*WHEN 2*/;                                            EB001000
                                                                        EB001010
         WHEN(3) DO; /*ENAME NODE*/                                     EB001020
             UPDN_ARG.PSETID=KEY(KY.PENAME);                            EB001030
             UPDN_ARG.PARENT=3;                                         EB001040
             UPDN_ARG.BSETID=KEY(KY.BENAME);                            EB001050
             UPDN_ARG.OP=UPDNOP.ID;                                     EB001060
             UPDN_ARG.DLEN=8;                                           EB001070
             IF J<3 THEN                                                EB001080
                UPDN_ARG.DATA=UNSPEC ('E*ESET  ');                      EB001090
               ELSE UPDN_ARG.DATA=UNSPEC ('E*ASET  ');                  EB001100
```

```
          END /*3*/;                                                    EB001110
                                                                        EB001120
        WHEN (4) DO; /*AINFO NODE*/                                     EB001130
             UPDN_ARG.PSETID=KEY(KY.PAINFO);                            EB001140
             UPDN_ARG.PARENT=1;                                         EB001150
             UPDN_ARG.BSETID=KEY(KY.BAINFO);                            EB001160
             UPDN_ARG.OP=UPDNOP.CI;                                     EB001170
             UPDN_ARG.DLEN=29;                                          EB001180
             UPDN_ARG.DATA=STRING (BAINFO(J));                          EB001190
            END /*4*/;                                                  EB001200
                                                                        EB001210
     END /*SELECT I */;                                                 EB001220
   END /*DO I*/;                                                        EB001230
                                                                        EB001240
   CALL TCALL ('UPDN',1, RP, TP);                                       EB001250
   END /*DO J*/;                                                        EB001260
   END /*EBOT*/;                                                        EB001270
```

```
L1:PROC;  /* A DUMMY ROUTINE TO SIMULATE ENTIRE SH-STV */          L1 00010
%INCLUDE USERS.VPX.ARG.SH;                                        L1 00020
                                                                  L1 00030
ARGPTR = VP.WAIT.MSG;                                             L1 00040
INDX = ARG.VIRTUAL_ADDRESS/8 + 1;                                 L1 00050
IF ARG.REQ_TYPE = 'READ'                                          L1 00060
THEN ARG.DATA = PACKET(INDX);                                     L1 00070
ELSE PACKET(INDX) = ARG.DATA;                                     L1 00080
CALL SEND (0,ARG.VPID,ARG.BOXID,'S',20,ARGPTR);                   L1 00090
CALL FINISH;                                                      L1 00100
RETURN;                                                           L1 00110
END;                                                              L1 00120
```

AD-A116 591 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA CEN--ETC F/G 9/2
FSTV: THE SOFTWARE TEST VEHICLE FOR THE FUNCTIONAL HIERARCHY OF--ETC(U)
JAN 82 M HSU N00039-81-C-0663
UNCLASSIFIED CISR-M010-8201-09 NL



```
FSTV: PROC;                                                              FST00010
/*****************************************************************     FST00020
*                                                                 *     FST00030
*                    MODULE    DESCRIPTION                        *     FST00040
*****************************************************************/     FST00050
/*****************************************************************     FST00060
*****   PURPOSE:  A T-PROC FOR CONSOLE INITIALIZATION FROM          *     FST00070
*****     AN EXISTING FILE OR FROM SCRATCH, AND CONSOLE LOGOUT.     *     FST00080
*****************************************************************     FST00090
*****   METHOD:                                                     *     FST00100
*****   IT ASKS FOR INIT METHOD FROM THE CONSOLE USER.  IF FILE INIT,*     FST00110
*****   IT ASKS FOR FILE NAME AND PERCOLATES THE FILE INIT DOWN. IF *     FST00120
*****   NEW INIT, IT CALLS SBRS TO INITIALIZE CAT FILES OF THIS LEVEL*     FST00130
*****   AND PERCOLATES THE NEW INIT DOWN.                           *     FST00140
*****                                                               *     FST00150
*****************************************************************     FST00160
*****************************************************************     FST00170
*****   CALLS PROCEDURES:  RINIT, CINIT, VINIT, VSAVE, USER;        *     FST00180
***** (INTRA-LEVEL S-PROCS: RINIT, CINIT: USED FOR INITIALIZING THE *     FST00190
*****   RELATIONAL VIEW PROCESSOR AND THE BN VIEW PROCESSOR, BOTH   *     FST00200
*****   ARE CURRENTLY INCLUDED IN THE TOP LEVEL BUT EVENTUALLY WILL *     FST00210
*****   BE DEDICATED TO THE VTL, WHICH DOES NOT EXIST YET.          *     FST00220
*****   USER: ENTER INTO USER MODE FROM CONSOLE.                    *     FST00230
*****  INTER-LEVEL T-PROC: VINIT, VSAVE: USED FOR PERCOLATING INIT  *     FST00240
*****   AND SAVE COMMANDS TO THE NEXT LEVEL.)                       *     FST00250
*****                                                               *     FST00260
*****************************************************************/     FST00270
ON ERROR SNAP SYSTEM;                                                    FST00280
%INCLUDE  EUSER;   /*INTRA-LEVEL S-PROC ENTRY DCLS*/                    FST00290
%INCLUDE AVINI ;  /*INTER-LEVEL T-PROC ARG LIST*/                       FST00300
%INCLUDE SERVICE,USERS,VPX /*CONTROL STRUCTURE SERVICES*/ ;              FST00310
%INCLUDE FDEBUG;                                                        FST00320
DCL NULL BUILTIN;                                                        FST00330
DCL RTN_CODE FIXED BIN(15);                                              FST00340
DCL (TP,P) POINTER;                                                      FST00350
DCL (NAME, FNAME) CHAR(8);                                               FST00360
DCL (SETFH) ENTRY;                                                       FST00370
/* BEGIN SESSION  */                                                    FST00380
PUT SKIP (2) LIST ('--FUNCTIONAL HIERARCHY STV CONSOLE PROGRAM--');      FST00390
CALL SETFH;         /* SETTING FSTV TRACE OPTIONS */                    FST00400
CALL NEXTI;         /* INITIALIZE SYSTEM CATALOGUES OF FSTV */          FST00410
CALL TCALL('USER',1,P,TP); /* THEN START USER PROCESSING */             FST00420
                           /* P AND TP IRRELEVANT */                    FST00430
                                                                         FST00440
  /* WHEN USER QUITS USER SESSION */                                     FST00450
/***************** TIMING REPORT OPTION ***************************/    FST00460
 ON ZERODIVIDE;                                                          FST00470
 DCL ((ETIME,ELO,ECOUNT)(50) FIXED BIN(31)) EXT;                         FST00480
 DCL FTIME BIT(1) EXT;                                                   FST00490
 %INCLUDE FSTRUC;                                                        FST00500
 IF FTIME THEN DO;                                                       FST00510
 PUT EDIT ('LEVEL  PROCNAME  COUNT  TOT ELAPSED TIME  TOT RUN TIME  RUNFST00520
 TIME/INVOCATION')(SKIP,A);                                              FST00530
 PUT EDIT ('-----  --------  -----  ----------------  ------------  ---FST00540
 ---------------')(SKIP,A);                                            FST00550
```

```
 PUT EDIT(((ARCH.LEVEL(I),ARCH.PROCNAME(I),ECOUNT(I),ETIME(I),          FST00560
                        (ETIME(I)-ELO(I)),                              FST00570
                        (ETIME(I)-ELO(I))/ECOUNT(I))                    FST00580
          DO I=1 TO 19))                                                FST00590
          (SKIP,F(3),X(4),A,F(7),F(17),F(13),F(21));                    FST00600
 END;                                                                   FST00610
/*****************************************************************/     FST00620
                                                                        FST00630
CALL FINISH;/* SUICIDE */                                              FST00640
                                                                        FST00650
                                                                        FST00660
/***INTERNAL SUBROUTINES**********************/                        FST00670
NEXTI: PROC;  /* INT SBR FOR PASSING INIT DOWN NEXT LEVEL*/            FST00680
GETI:                                                                   FST00690
  PUT LIST ('REGENERATE(NEW) OR OLD DATA BASE(FILE):');                 FST00700
  GET EDIT (FNAME) (A(8));                                              FST00710
  SELECT (FNAME);                                                       FST00720
    WHEN ('N','NEW','n','new') FNAME = '';                              FST00730
    WHEN ('F','FILE','f','file') FNAME = 'F';                           FST00740
    OTHERWISE GO TO GETI; END /*SELECT */;                              FST00750
  ALLOCATE VINIT_ARG;  /*SET UP ARG LIST*/                              FST00760
  VINIT_ARG.PTR=NULL;                                                   FST00770
  VINIT_ARG.FNAME=FNAME;                                                FST00780
  IF FDEBUG.LEVO THEN PUT SKIP LIST ('FSTV CALLS VINIT');               FST00790
  CALL TCALL ('VINIT',1,P,TP);                                          FST00800
  IF FDEBUG.LEVO THEN PUT SKIP LIST ('FSTV: VINIT RETURNS');            FST00810
  CALL SVC3 (P);                                                        FST00820
  CALL SVC3 (TP);                                                       FST00830
  END /*NEXTI*/;                                                       FST00840
                                                                        FST00850
/*****************************************************************/     FST00860
CINIT: PROC;                                                            FST00870
 /*IN THE FUTURE, THIS MODULE INITIALIZES THE BASE DATA MODEL VIEW      FST00880
   SUBSYSTEM; PRESENTLY IT IS NOT AVAILABLE*/                           FST00890
 RETURN;                                                                FST00900
 END;                                                                   FST00910
                                                                        FST00920
RINIT: PROC;                                                            FST00930
 /*IN THE FUTURE, THIS MODULE INITIALIZES THE RELATIONAL  VIEW          FST00940
   SUBSYSTEM; PRESENTLY IT IS NOT AVAILABLE*/                           FST00950
 RETURN;                                                                FST00960
 END;                                                                   FST00970
                                                                        FST00980
                                                                        FST00990
END /*FSTV*/;                                                          FST01000
```

```
GACT: PROC (ANAME, ENAME, AINFO, RTN_CODE);                              GAC00010
                                                                         GAC00020
/**** A S-PROC IN THE ENTITY LEVEL RESPONSIBLE FOR RETRIEVING ATTRI-     GAC00030
**** BUTE CAT ENTRY GIVEN ATTRIB NAME AND ENTITY NAME *********/         GAC00040
                                                                         GAC00050
%INCLUDE DECAT, DKEY, ARETN, SERVICE, FDEBUG;                            GAC00060
                                                                         GAC00070
DCL (ANAME, ENAME) CHAR (8);                                             GAC00080
DCL (P,RP,LP,TP,RTP) PTR,(NULL,ADDR) BUILTIN;                            GAC00090
DCL BIT_STRING BIT (232) BASED;                                          GAC00100
DCL RTN_CODE FIXED BIN, I FIXED BIN;                                     GAC00110
DCL UNSPEC BUILTIN, STRING BUILTIN;                                      GAC00120
                                                                         GAC00130
/*BEGIN PROCESSING*/                                                     GAC00140
 RTN_CODE=0; /*INIT TO O.K.*/                                            GAC00150
                                                                         GAC00160
/*RETN TREE ROOT NODE IS ASET*/                                          GAC00170
 I=0; /*NO PREDICATE*/                                                   GAC00180
 ALLOCATE RETN_ARG; RP,LP=P;                                             GAC00190
 RETN_ARG.PSETID=KEY(KY.PASET);                                          GAC00200
 RETN_ARG.GET=RETNGET.ANY;                                               GAC00210
 RETN_ARG.N=0; /*NO PREDICATE*/                                          GAC00220
                                                                         GAC00230
/*ANAME NODE*/                                                           GAC00240
I=1; /*1 PREDICATE*/                                                     GAC00250
ALLOCATE RETN_ARG;                                                       GAC00260
LP->RETN_ARG.PTR=P;   LP=P;                                              GAC00270
 RETN_ARG.PSETID=KEY(KY.PANAME);                                         GAC00280
 RETN_ARG.NODE=2;                                                        GAC00290
 RETN_ARG.PARENT=1;                                                      GAC00300
 RETN_ARG.BSETID=KEY(KY.BANAME);                                         GAC00310
 RETN_ARG.GET=RETNGET.NO;                                                GAC00320
 RETN_ARG.N=1; /*1 PREDICATE*/                                           GAC00330
 RETN_ARG.PRED.CN(1)=1;                                                  GAC00340
 RETN_ARG.PRED.OP(1)=RETNOP.EQ;                                          GAC00350
 RETN_ARG.PRED.DLEN(1)=8;                                                GAC00360
 RETN_ARG.PRED.DATA=UNSPEC (ANAME);                                      GAC00370
                                                                         GAC00380
/*ESET NODE*/                                                            GAC00390
 I=0;                                                                    GAC00400
 ALLOCATE RETN_ARG;                                                      GAC00410
 LP->RETN_ARG.PTR=P; LP=P;                                               GAC00420
 RETN_ARG.PSETID=KEY(KY.PESET);                                          GAC00430
 RETN_ARG.NODE=3;                                                        GAC00440
 RETN_ARG.PARENT=1;                                                      GAC00450
 RETN_ARG.BSETID=KEY(KY.BESET);                                          GAC00460
 RETN_ARG.GET=RETNGET.NO;                                                GAC00470
 RETN_ARG.N=0; /*NO PREDICATE*/                                          GAC00480
                                                                         GAC00490
/*ENAME NODE*/                                                           GAC00500
I=1; /*1 PREDICATE*/                                                     GAC00510
ALLOCATE RETN_ARG;                                                       GAC00520
LP->RETN_ARG.PTR=P;   LP=P;                                              GAC00530
 RETN_ARG.PSETID=KEY(KY.PENAME);                                         GAC00540
 RETN_ARG.NODE=4;                                                        GAC00550
```

```
 RETN_ARG.PARENT=3;                                                    GAC00560
 RETN_ARG.BSETID=KEY(KY.BENAME);                                       GAC00570
 RETN_ARG.GET=RETNGET.NO;                                              GAC00580
 RETN_ARG.N=1; /*1 PREDICATE*/                                         GAC00590
 RETN_ARG.PRED.CN(1)=1;                                                GAC00600
 RETN_ARG.PRED.OP(1)=RETNOP.EQ;                                        GAC00610
 RETN_ARG.PRED.DLEN(1)=8;                                              GAC00620
 RETN_ARG.PRED.DATA=UNSPEC (ENAME);                                    GAC00630
                                                                       GAC00640
/*AINFO NODE*/                                                        GAC00650
I=0; /*NO PREDICATE*/                                                 GAC00660
ALLOCATE RETN_ARG;                                                    GAC00670
LP->RETN_ARG.PTR=P;   LP=P;                                           GAC00680
 RETN_ARG.PSETID=KEY(KY.PAINFO);                                       GAC00690
 RETN_ARG.NODE=5;                                                      GAC00700
 RETN_ARG.PARENT=1;                                                    GAC00710
 RETN_ARG.BSETID=KEY(KY.BAINFO);                                       GAC00720
 RETN_ARG.GET=RETNGET.ANY;                                             GAC00730
 RETN_ARG.N=0; /*0 PREDICATE*/                                         GAC00740
                                                                       GAC00750
/*CALL UP RETN*/                                                      GAC00760
CALL TCALL ('RETN',1,RP,TP);                                          GAC00770
                                                                       GAC00780
/*DECODE RETURN*/                                                     GAC00790
IF TP->RETN_RTN.RTN_CODE^=0 THEN DO;                                  GAC00800
   RTN_CODE=1;/*PROBLEM IN RETRIEVAL*/                                GAC00810
   CALL SVC3 (RP);                                                    GAC00820
   CALL SVC3 (TP);                                                    GAC00830
   RETURN;                                                            GAC00840
   END;                                                               GAC00850
                                                                       GAC00860
IF TP->RETN_RTN.N^=1 THEN DO;                                         GAC00870
   RTN_CODE=2; /*NO SUCH ATTRIBUTE*/                                  GAC00880
   CALL SVC3 (RP);                                                    GAC00890
   CALL SVC3 (TP);                                                    GAC00900
   RETURN;                                                            GAC00910
   END;                                                               GAC00920
                                                                       GAC00930
/*GET DATA AND DECODE*/                                               GAC00940
P=TP->RETN_RTN.PTR->RETN_RTN1.PTR; /*THE 3RD TOKEN RETURNED*/         GAC00950
ADDR(AINFO)->BIT_STRING=RETN_RTN1.DATA;                               GAC00960
                                                                       GAC00970
RETURN: CALL SVC3 (RP); CALL SVC3 (TP);                               GAC00980
RETURN; /*RETURNS*/                                                   GAC00990
END /*GACT*/;                                                         GAC01000
```

```
GBCT: PROC(BSETID,BINFO,RTN_CODE);                                    GBC00010
 /*A SUBROUTINE WHICH ACCEPTS THE BSETID AND RETURNS A DECODED BINFO   GBC00020
   *** TO THE CALLER**********************/                           GBC00030
                                                                      GBC00040
%INCLUDE DBCAT,FDEBUG,DBEU;                                           GBC00050
%INCLUDE ESUB1;                                                       GBC00060
DCL P PTR, (NULL,ADDR) BUILTIN, RTN_CODE FIXED BIN;                   GBC00070
DCL BSETID BIT (32);                                                  GBC00080
DCL BIT_STRING BIT (88) BASED;                                        GBC00090
                                                                      GBC00100
CALL RET1 (BSETID,P,RTN_CODE);                                        GBC00110
IF P=NULL THEN DO;                                                    GBC00120
   RTN_CODE=1; /***ILLEGA BSETID --1*****/                            GBC00130
   RETURN;                                                            GBC00140
   END;                                                               GBC00150
                                                                      GBC00160
ADDR(BINFO)->BIT_STRING=BEU.DATA;                                     GBC00170
FREE BEU;                                                             GBC00180
RTN_CODE=0;                                                           GBC00190
RETURN;                                                               GBC00200
END;                                                                  GBC00210
```

```
GECT :PROC (ENAME, EINFO, RTN_CODE);                                    GEC00010
                                                                        GEC00020
/****** A S-PROC IN THE ENTITY LEVEL TO OBTAIN THE ENTITY CAT. ENTRY   GEC00030
***** OF ENTITY SET WHOSE NAME IS SPECIFIED IN ENAME; RETURNS EINFO*/   GEC00040
                                                                        GEC00050
%INCLUDE DECAT, ARETN, DKEY, SERVICE, FDEBUG;                           GEC00060
                                                                        GEC00070
DCL RTN_CODE FIXED BIN, ENAME CHAR (8);                                 GEC00080
DCL (P,TP,RP,LP) PTR, I FIXED BIN, NULL BUILTIN;                        GEC00090
DCL UNSPEC BUILTIN;                                                     GEC00100
                                                                        GEC00110
/*BEGIN PROCESSING*/                                                    GEC00120
RTN_CODE=0; /*INIT TO O.K.*/                                            GEC00130
/*ALLOCATE ROOT RETN TREE - ESET*/                                      GEC00140
I=0; /*NO PREDICATE*/                                                   GEC00150
ALLOCATE RETN_ARG; LP,RP=P;                                             GEC00160
RETN_ARG.PSETID=KEY(KY.PESET);                                          GEC00170
RETN_ARG.GET=RETNGET.ANY;                                               GEC00180
RETN_ARG.N=0; /*0 PREDICATE*/                                           GEC00190
                                                                        GEC00200
/*NEXT NODE -- ENAME**/                                                 GEC00210
I=1; /*NO PREDICATE*/                                                   GEC00220
ALLOCATE RETN_ARG;                                                      GEC00230
LP->RETN_ARG.PTR=P; LP=P;                                               GEC00240
RETN_ARG.PSETID=KEY(KY.PENAME);                                         GEC00250
RETN_ARG.GET=RETNGET.NO;                                                GEC00260
RETN_ARG.NODE=2;                                                        GEC00270
RETN_ARG.BSETID=KEY(KY.BENAME);                                         GEC00280
RETN_ARG.PARENT=1;                                                      GEC00290
RETN_ARG.N=1; /*1 PREDICATE*/                                           GEC00300
RETN_ARG.CN(1)=1;                                                       GEC00310
RETN_ARG.OP(1)=RETNOP.EQ;                                               GEC00320
RETN_ARG.DLEN(1)=8;                                                     GEC00330
RETN_ARG.DATA(1)=UNSPEC (ENAME);                                        GEC00340
                                                                        GEC00350
/*NEXT NODE -- EINFO **/                                                GEC00360
I=0; /*NO PREDICATE*/                                                   GEC00370
ALLOCATE RETN_ARG;                                                      GEC00380
LP->RETN_ARG.PTR=P; LP=P;                                               GEC00390
RETN_ARG.PSETID=KEY(KY.PEINFO);                                         GEC00400
RETN_ARG.GET=RETNGET.ANY;                                               GEC00410
RETN_ARG.NODE=3;                                                        GEC00420
RETN_ARG.BSETID=KEY(KY.BEINFO);                                         GEC00430
RETN_ARG.PARENT=1;                                                      GEC00440
RETN_ARG.N=0; /*0 PREDICATE*/                                           GEC00450
                                                                        GEC00460
/*CALL UP RETN*/                                                        GEC00470
CALL TCALL ('RETN',1,RP,TP);                                            GEC00480
                                                                        GEC00490
/*CHECK RETURNS*/                                                       GEC00500
IF TP->RETN_RTN.RTN_CODE^=0 THEN DO;                                    GEC00510
   RTN_CODE=1;/*PROBLEM IN RETRIEVAL*/                                  GEC00520
   CALL SVC3 (RP);                                                      GEC00530
   CALL SVC3 (TP);                                                      GEC00540
   RETURN;                                                              GEC00550
```

```
   END;                                                              GEC00560
                                                                     GEC00570
IF TP->RETN_RTN.N^=1 THEN DO;                                        GEC00580
   RTN_CODE=2; /*NO SUCH ENTITY*/                                    GEC00590
/* IF FDEBUG.LEV1 THEN PUT SKIP LIST ('GECT: NO SUCH ENTITY'); */    GEC00600
   CALL SVC3 (RP);                                                   GEC00610
   CALL SVC3 (TP);                                                   GEC00620
   RETURN;                                                           GEC00630
   END;                                                              GEC00640
                                                                     GEC00650
/*GET DATA AND DECODE*/                                              GEC00660
P=TP->RETN_RTN.PTR->RETN_RTN1.PTR; /*THE 3RD TOKEN RETURNED*/        GEC00670
EINFO=P->RETN_RTN1.DATA;                                             GEC00680
CALL SVC3 (TP);                                                      GEC00690
CALL SVC3 (RP);                                                      GEC00700
RETURN;                                                              GEC00710
END /*GECT*/;                                                        GEC00720
```

```
GMEM: PROC(NUM_BYTES, ID);                                              GME00010
                                                                        GME00020
/* THIS IS A SUBROUTINE AT THE MM LEVEL WHICH RESPOND TO A 'GET MAIN'   GME00030
   REQUEST BY LOOKING UP THE NEXT FREE AREA AND ASSIGN AN ID TO IT */   GME00040
                                                                        GME00050
DCL NUM_BYTES FIXED BIN, ID BIT (32), DATA CHAR(8), DT4 CHAR(4);        GME00060
DCL (AVAIL_LOC,I) FIXED BIN(31), (UNSPEC,SUBSTR) BUILTIN;               GME00070
%INCLUDE EPAKT;                                                         GME00080
CALL PAKT (0,DATA,'R');            /* GET FIRST PACKET */               GME00090
UNSPEC(AVAIL_LOC) = UNSPEC(SUBSTR(DATA,1,4));  /* OBTAIN AVAIL_LOC */   GME00100
ID = UNSPEC(AVAIL_LOC);                          /* ASSIGN ID */        GME00110
I =(NUM_BYTES-1)/8+1;          /* FORCE TO BE MULTIPLE OF 8 */          GME00120
AVAIL_LOC = AVAIL_LOC +I*8;                       /* UPDATE IT */       GME00130
UNSPEC(DT4) = UNSPEC(AVAIL_LOC);                                        GME00140
SUBSTR(DATA,1,4) = DT4;                                                 GME00150
CALL PAKT (0,DATA,'W');                    /* WRITE NEW AVAIL_LOC */    GME00160
RETURN;                                                                 GME00170
END;                                                                    GME00180
```

```
GPCT: PROC(PSETID,PINFO,RTN_CODE);                                    GPC00010
 /*A SUBROUTINE WHICH ACCEPTS THE PSETID AND RETURNS A DECODED PINFO   GPC00020
   *** TO THE CALLER**********************/                           GPC00030
                                                                      GPC00040
%INCLUDE DPCAT,FDEBUG,DBEU;                                           GPC00050
%INCLUDE ESUB1;                                                       GPC00060
DCL P PTR, (NULL,ADDR) BUILTIN, RTN_CODE FIXED BIN;                   GPC00070
DCL PSETID BIT (32);                                                  GPC00080
DCL BIT_STRING BIT (144) BASED;                                       GPC00090
                                                                      GPC00100
CALL RET1 (PSETID,P,RTN_CODE);                                        GPC00110
IF P=NULL THEN DO;                                                    GPC00120
   RTN_CODE=1; /***ILLEGA PSETID --1*****/                            GPC00130
   RETURN;                                                            GPC00140
   END;                                                               GPC00150
                                                                      GPC00160
ADDR(PINFO)->BIT_STRING=BEU.DATA;                                     GPC00170
FREE BEU;                                                             GPC00180
RTN_CODE=0;                                                           GPC00190
RETURN;                                                               GPC00200
END;                                                                  GPC00210
```

```
GSEU: PROC (ID,SEUPTR,RTN_CODE);                                        GSE00010
                                                                        GSE00020
/*** THIS IS A SUBROUTINE AT THE MM LEVEL WHICH RETURNS A PTR TO AN     GSE00030
 *** SEU (SEUPTR) OR A RETURN CODE (RTN_CODE) GIVEN THIS SEU'S ID;      GSE00040
 *** IT CALLS PAKT TO GET PACKETS FROM THE DSH ********************/    GSE00050
                                                                        GSE00060
%INCLUDE DSEU1,EPAKT;                                                   GSE00070
DCL DATA CHAR(8), VA FIXED BIN(31), TP PTR, DT5 CHAR(5);                GSE00080
DCL (I,RTN_CODE,L,L1,L2) FIXED BIN, ID BIT (32);                        GSE00090
DCL (NULL,ADDR,SUBSTR,UNSPEC,MIN) BUILTIN;                              GSE00100
                                                                        GSE00110
RTN_CODE = 0;                                                           GSE00120
UNSPEC(VA) = ID;                                                        GSE00130
CALL PAKT (VA,DATA,'R');          /* OBTAIN FIRST 8 BYTES OF THIS SEU */ GSE00140
SEUPTR = ADDR (DATA);             /* IMPOSE SEU STRUCTURE ON DATA */    GSE00150
IF SEU.INVAL ^= 'N' THEN DO;      /* IF ATTEMPT TO RETRIEVE INVAL SEU */ GSE00160
   RTN_CODE = 1;                                                        GSE00170
   RETURN;                                                              GSE00180
   END;                                                                 GSE00190
                                                                        GSE00200
I=SEU.N;                          /* ESTABLISH FIRST 8 BYTES OF SEU */  GSE00210
ALLOC SEU SET (TP);                                                     GSE00220
TP->SEU.INVAL = SEU.INVAL;                                              GSE00230
SEUPTR = TP;                                                            GSE00240
L = MIN (5,SEU.N);                                                      GSE00250
DATA = SUBSTR(DATA,4,L);                                                GSE00260
DT5 = SUBSTR(DATA,1,L);                                                 GSE00270
SUBSTR(SEU.DATA,1,L) = DT5;                                             GSE00280
                                  /* ESTABLISH THE REST OF SEU */       GSE00290
L = (SEU.N + 2)/8;                /* L = # OF PACKETS IN THIS SEU -1 */ GSE00300
DO I = 1 TO L;                    /* GET ALL PACKETS          */        GSE00310
   CALL PAKT (VA+I*8,DATA,'R');                                         GSE00320
   L1 = I*8-2; L2 = MIN(8,SEU.N-L1+1);                                  GSE00330
   DATA = SUBSTR (DATA,1,L2);                                           GSE00340
   SUBSTR (SEU.DATA,L1,L2)= DATA;                                       GSE00350
   END;                                                                 GSE00360
                                                                        GSE00370
RETURN;                                                                 GSE00380
END;                                                                    GSE00390
```

```
LEX: PROC (I, TOKEN, MODE, RTN_CODE);                                   LEX00010
/*********************************************************************  LEX00020
*                                                                   *  LEX00030
*                    MODULE    DESCRIPTION                          *  LEX00040
*********************************************************************/ LEX00050
/*********************************************************************  LEX00060
*****   PURPOSE: THIS IS THE LEXICAL ANALYZER WHICH PARSES AN INPUT    LEX00070
*****     LINE INTO TOKENS. TOKEN DELIMITERS ARE ',' AND ':'.          LEX00080
*****     DELIMITERS IN QUOTES ARE NOT TREATED AS EXCEPTIONS.          LEX00090
*****     MORE THAN 1 LINE OF INPUT CAN BE STRUNG TOGETHER BY          LEX00100
*****     ENDING PREVIOUS LINES IN BACK SLASH.                         LEX00110
*****     BLANKS CAN BE OPTIONALLY SUPPRESSED.  BLANKS ENCLOSED IN     LEX00120
*****     QUOTES ARE NOT SUPPRESSED.                                   LEX00130
*********************************************************************  LEX00140
*****   INPUT PARAMETERS:                                              LEX00150
*****     MODE: IF SET INDICATES THAT BLANKS ARE TO BE SUPPRESSED.     LEX00160
*********************************************************************  LEX00170
*****   OUTPUT PARAMETERS:                                             LEX00180
*****     I: NUMBER OF TOKENS                                          LEX00190
*****     TOKEN: AN ARRAY CONTAINING THE TOKENS (CHAR$ UP              LEX00200
*****            TO 40 CHAR'S) AND LENGTH OF TOKENS                    LEX00210
*****     RTN_CODE: 0 - O.K.                                           LEX00220
*****               2 - UNBALANCED QUOTES                              LEX00230
*****               1 - EXCESSIVE SEPARATORS                           LEX00240
*********************************************************************  LEX00250
                                                                        LEX00260
        /* LEXICAL ANALYZER VARIABLES */                                LEX00270
        DCL 1 TOKEN (25),                                               LEX00280
             2 D CHAR(40),                                              LEX00290
             2 L FIXED BIN;                                             LEX00300
        DCL RTN_CODE FIXED BIN;                                         LEX00310
        DCL I FIXED BIN,                                                LEX00320
            LAST FIXED BIN(15), LAST1 FIXED BIN;                        LEX00330
        DCL MODE FIXED BIN; /*1=SUPPRESS ALL BLANKS*/                   LEX00340
                                                                        LEX00350
        /* INPUT STRING VARIABLES */                                    LEX00360
        DCL (STR1,STR2,STR) CHAR(240) VARYING,                          LEX00370
            LINE CHAR(80),                                              LEX00380
            POS FIXED BIN(15);                                          LEX00390
        DCL (VERIFY,SUBSTR,INDEX,LENGTH) BUILTIN;                       LEX00400
                                                                        LEX00410
/* START SUBROUTINE */                                                  LEX00420
         RTN_CODE=0;                                                    LEX00430
                                                                        LEX00440
        /* GET INPUT STRING */                                          LEX00450
        STR = '';                                                       LEX00460
LOOP:   GET EDIT (LINE) (A(80));                                        LEX00470
        POS = INDEX(LINE,'/');                                          LEX00480
        IF POS ^= 0                                                     LEX00490
        THEN DO;                                                        LEX00500
             STR = STR || SUBSTR(LINE,1,POS-1);                         LEX00510
             GO TO LOOP;                                                LEX00520
             END;                                                       LEX00530
        STR = STR || LINE;                                              LEX00540
                                                                        LEX00550
```

```
/* FIX UP LINE */                                                    LEX00560
DO I = LENGTH(STR) TO 1 BY -1 WHILE(SUBSTR(STR,I,1) = ' ');          LEX00570
END;                                                                 LEX00580
STR = SUBSTR(STR,1,I) || '..';                                       LEX00590
                                                                     LEX00600
 /*SUPPRESS BLANKS*/                                                 LEX00610
IF MODE=1 THEN DO;                                                   LEX00620
   DO I=1 TO LENGTH(STR);                                            LEX00630
     /*SKIP BLANKS IN QUOTES*/                                       LEX00640
     IF SUBSTR(STR,I,1)='''' THEN DO WHILE ('1'B);                   LEX00650
        I=I+1;                                                       LEX00660
        IF I>LENGTH(STR) THEN DO;                                    LEX00670
           RTN_CODE=2; /*ILLEGAL SYNTAX */                           LEX00680
           RETURN;                                                   LEX00690
           END;                                                      LEX00700
        IF SUBSTR(STR,I,1)='''' THEN LEAVE;                          LEX00710
        END;                                                         LEX00720
                                                                     LEX00730
    ELSE                                                             LEX00740
     IF SUBSTR(STR,I,1)=' ' THEN  DO;                                LEX00750
      STR1=SUBSTR(STR,1,I-1) ;                                       LEX00760
      STR2=SUBSTR(STR,I+1);                                          LEX00770
      STR=STR1||STR2;                                                LEX00780
      I=I-1; /*FIX UP I*/                                            LEX00790
       END;                                                          LEX00800
   END;                                                              LEX00810
END;                                                                 LEX00820
                                                                     LEX00830
/* LEX STARTS */                                                     LEX00840
I = 0;                                                               LEX00850
/* EXTRACT TOKENS */                                                LEX00860
DO WHILE(VERIFY(STR,',') ^= 0);   /* DO TILL END OF STRING */        LEX00870
    LAST = INDEX(STR,',') - 1;   /* LOOKING FOR DELIMITER */         LEX00880
    LAST1 = INDEX (STR,':') -1;                                      LEX00890
    IF LAST1 > -1 & LAST1 < LAST THEN LAST = LAST1;                  LEX00900
    ELSE IF LAST < 0 THEN LAST = LAST1;                              LEX00910
                                                                     LEX00920
    IF LAST = 0                                                      LEX00930
    THEN DO; /* MISUSE OF DELIMITER */                               LEX00940
       RTN_CODE = 1;                                                 LEX00950
       RETURN;                                                       LEX00960
       END;                                                          LEX00970
                                                                     LEX00980
    I = I + 1;                                                       LEX00990
    TOKEN.D(I) = SUBSTR(STR,1,LAST);                                 LEX01000
    TOKEN.L(I) = LAST;                                               LEX01010
    STR = SUBSTR(STR,LAST+2);                                        LEX01020
END;                                                                 LEX01030
END /*LEX*/;                                                         LEX01040
```

```
 LEXT: PROC(N,TOKEN,T1,TN,PRED,MOP,RTN_CODE);                          LEX00010
/*********************************************************************  LEX00020
*                                                                   *  LEX00030
*                    MODULE    DESCRIPTION                          *  LEX00040
*********************************************************************/ LEX00050
/*********************************************************************  LEX00060
*****   PURPOSE:                                                       LEX00070
*** S-PROC AT THE USER INTERFACE LEVEL WHICH ACCEPTS TOKENS            LEX00080
*** IN A MATRIX AND IDENTIFIES THE TREE STRUCTURE AND THE PREDICATE    LEX00090
*** VALUES IN THESE TOKENS; IT ALSO DISCERNS MODIFICATION OPERATORS    LEX00100
*** FROM REGULAR ATTRIBUTE NAMES AND STORE THEM IN STRUCTURE MOP */    LEX00110
/****                                                                  LEX00120
*********************************************************************  LEX00130
*****   METHOD:                                                        LEX00140
*****    THE HIERARCHICAL STRUCTURE OF NAMES IS IDENTIFIED BY          LEX00150
*****    NESTED PARANTHESES.  FOR EXAMPLE, IF TOKEN (1) IS 'ABC(',     LEX00160
*****    TOKEN(2) IS 'DEF' AND TOKEN(3) IS 'GHI)', THEN DEF AND GHI    LEX00170
*****    HAVE ABC AS THEIR PARENT NAME.                                LEX00180
*****    THERE ARE TWO TYPES OF NON-NAME TOKENS: MODIFICATION OPERATOR LEX00190
*****    (MOP) AND PREDICATE TOKEN.  MOP IS DISTINGUISHED FROM REGULAR LEX00200
*****    NAME TOKENS BY ITS SPECIAL STARTING CHAR '-'.  TOKEN THAT     LEX00210
*****    CONTAIN SPECIAL COPARISON OPERATORS ('=','<','>') IS TAKEN    LEX00220
*****    AS PREDICATE TOKEN AND IS BROKEN INTO A NAME TOKEN AND A      LEX00230
*****    VALUE TOKEN.                                                  LEX00240
*****                                                                  LEX00250
*********************************************************************  LEX00260
*****   INPUT PARAMETERS:                                              LEX00270
*****     N: NUMBER OF TOKENS IN THE TOKEN ARRAY                       LEX00280
*****     TOKEN: TOKEN ARRAY                                           LEX00290
*********************************************************************  LEX00300
*****   OUTPUT PARAMETERS:                                             LEX00310
*****      T1: AN ARRAY DESCRIBING THE HIERARCHICAL STRUCTURE OF       LEX00320
*****          NAME TOKENS.  ROOT NAMES HAVE PARENT = 0 WHILE          LEX00330
*****          NON-ROOT NAMES HAVE A PARENT EQUAL TO THE INDEX         LEX00340
*****          OF THE PARENT IN T1.                                    LEX00350
*****      TN: NUMBER OF NAMES IN T1.                                  LEX00360
*****      PRED: AN ARRAY USED TO STORE PREDICATE OP, VALUE AND VALUE  LEX00370
*****            LENGTH IDENTIFIED FOR CORRESPONDING NAME IN T1.       LEX00380
*****      MOP: AN ARRAY USED TO OMDOCATE MODIFICATION OP IDENTIFIED   LEX00390
*****           FOR CORRESPONDING NAME IN T1.                          LEX00400
*****   RTN_CODE: 0 - O.K.                                             LEX00410
*****             1,2: MISMATCH OF PARANTHESES                         LEX00420
*****             3: ILLEGAL MODIFICATION OPERATOR                     LEX00430
*********************************************************************/ LEX00440
                                                                       LEX00450
                                                                       LEX00460
   DCL (N,TN,RTN_CODE,I,J,K)FIXED BIN;                                 LEX00470
                                                                       LEX00480
   DCL 1 T1 (25), /* ENTITY AND ATTRIBUTE NAMES TREE STRUCTURE*/       LEX00490
        2 PARENT FIXED BIN,                                            LEX00500
        2 ANAME CHAR(8);                                               LEX00510
   DCL 1 PRED (25), /*PRED VALUE STRUCTURE */                          LEX00520
        2 OP CHAR (1),                                                 LEX00530
        2 VALUE CHAR (40),                                             LEX00540
        2 VLEN FIXED BIN;                                              LEX00550
```

```
   DCL MOP (25) CHAR (1), CURRENT_MOP CHAR (1) INIT ('X');                LEX00560
                          /* INIT TO 'DON'T KNOW' */                      LEX00570
                                                                          LEX00580
    DCL 1 TOKEN (25),                                                     LEX00590
          2 D CHAR(40),                                                   LEX00600
          2 L FIXED BIN;                                                  LEX00610
    DCL ST(25) FIXED BIN, STP FIXED BIN;                                  LEX00620
    DCL (STR,DATA) CHAR(40) VARYING;                                      LEX00630
    DCL (SUBSTR,INDEX,LENGTH) BUILTIN;                                    LEX00640
                                                                          LEX00650
  /*BEGIN PROCESSING*/                                                    LEX00660
                                                                          LEX00670
  RTN_CODE=0;                                                             LEX00680
  TN=0; /*INIT T1 COUNT TO 0*/                                            LEX00690
  STP=1; ST(STP)=0; /*INIT STACK*/                                        LEX00700
                                                                          LEX00710
 DO I=1 TO N; /*N IS NUMBER OF TOKEN*/                                    LEX00720
    STR=SUBSTR(TOKEN.D(I),1,TOKEN.L(I)); /* EXTRACT TOKEN */              LEX00730
                                                                          LEX00740
    LOOP:  /* FOR EACH TOKEN PERFORM THIS LOOP */                         LEX00750
                                                                          LEX00760
    K=INDEX(STR,'(');                                                     LEX00770
    IF K^=0                                                               LEX00780
    THEN DO;                                                              LEX00790
       CALL T1P(K);   /* PROCESS UP TO '(' */                            LEX00800
       IF RTN_CODE ^= 0 THEN RETURN; /* SET BY T1P */                     LEX00810
       STP=STP+1; /*PUSH STACK*/                                          LEX00820
       ST(STP)=TN;                                                        LEX00830
       END /* '(' */;                                                    LEX00840
                                                                          LEX00850
     ELSE DO;                                                             LEX00860
        K=INDEX(STR,')');                                                 LEX00870
        IF K^=0                                                           LEX00880
        THEN DO;                                                          LEX00890
           CALL T1P(K);/* PROCESS UP TO ')'  */                           LEX00900
           IF RTN_CODE ^= 0 THEN RETURN;                                  LEX00910
           STP=STP-1;                                                     LEX00920
           IF STP=0 THEN DO;                                              LEX00930
              RTN_CODE=1;                    /*MISMATCH OF PARENTH*/     LEX00940
              RETURN;                                                     LEX00950
              END /* RTN_CODE */;                                         LEX00960
           END;                                                           LEX00970
                                                                          LEX00980
        ELSE CALL T1P(K);/* PROCESS ENTIRE STRING */                      LEX00990
        IF RTN_CODE ^= 0 THEN RETURN;                                     LEX01000
        END;                                                              LEX01010
                                                                          LEX01020
      IF STR^='' THEN GO TO LOOP;       /* RECYCLE IF STR NOT EMPTY */    LEX01030
      END;                                                                LEX01040
                                                                          LEX01050
  IF STP^=1 THEN RTN_CODE=2;          /* MISMATCH OF PARENTH */           LEX01060
                                                                          LEX01070
  RETURN;                                                                 LEX01080
                                                                          LEX01090
 /*****************************************************************/      LEX01100
```

```
T1P: PROC(K); /*INTERNAL SUBROUTINE*/                                   LEXO1110
DCL K FIXED BIN;                                                        LEXO1120
DCL OP(3) CHAR (1) INIT ('>','<','=');                                  LEXO1130
DCL VALUE CHAR (40) VARYING; DCL CMD CHAR (8);                          LEXO1140
DCL END FIXED BIN;                                                      LEXO1150
                                                                        LEXO1160
IF K=1                                                                  LEXO1170
THEN DO;           /*NO DATA*/                                         LEXO1180
   STR=SUBSTR(STR,K+1); RETURN;                                         LEXO1190
   END;                                                                 LEXO1200
                                                                        LEXO1210
IF K>1 THEN DO;                                                         LEXO1220
   DATA=SUBSTR(STR,1,K-1);                                              LEXO1230
   STR=SUBSTR(STR,K+1);                                                 LEXO1240
   END;                                                                 LEXO1250
                                                                        LEXO1260
IF K=0 THEN DO;                                                         LEXO1270
   DATA=STR;                                                            LEXO1280
   STR='';                                                              LEXO1290
   END;                                                                 LEXO1300
                                                                        LEXO1310
/* SEE IF TOKEN IS MODIFICATION OPERATORS */                            LEXO1320
K=INDEX(DATA, '-');                                                     LEXO1330
IF K = 1 THEN DO ; /* IT IS A MOP */                                    LEXO1340
   DATA = SUBSTR (DATA,2);                                              LEXO1350
   CMD = DATA;                                                          LEXO1360
                                                                        LEXO1370
   SELECT (CMD);                                                        LEXO1380
      WHEN ('ID') CURRENT_MOP='I';                                      LEXO1390
      WHEN ('INSERT','INS','IST') CURRENT_MOP='N';                      LEXO1400
      WHEN ('REPLACE','REP','R') CURRENT_MOP='R';                       LEXO1410
      WHEN ('DELETE','DEL','D') CURRENT_MOP='D';                        LEXO1420
      OTHERWISE RTN_CODE = 3; /* ILLEGAL SYNTAX */                      LEXO1430
      END /* SELECT */;                                                LEXO1440
                                                                        LEXO1450
   RETURN; /* FINISHED THIS TOKEN */                                    LEXO1460
   END /* MOP */;                                                      LEXO1470
                                                                        LEXO1480
/* PROCESS DATA - NOT A MOP */                                          LEXO1490
TN=TN+1;                                                                LEXO1500
DO J=1 TO 3; /* CHECK FOR VALUE */                                      LEXO1510
   K=INDEX(DATA,OP(J));                                                 LEXO1520
   IF K^=0 THEN LEAVE; /* FOUND */                                      LEXO1530
   END;                                                                 LEXO1540
                                                                        LEXO1550
IF K^=0 THEN DO ;/* OP FOUND, PROCESS VALUE BY BREAKING TOKEN INTO     LEXO1560
TWO: VALUE AND NAME DATA */                                             LEXO1570
   PRED.OP(TN)=SUBSTR(DATA,K,1);                                        LEXO1580
   VALUE=SUBSTR(DATA,K+1);                                              LEXO1590
   DATA=SUBSTR(DATA,1,K-1);                                             LEXO1600
                                                                        LEXO1610
   IF SUBSTR(VALUE,1,1)='''' THEN DO; /*TAKE QUOTES OUT */              LEXO1620
      END=LENGTH(VALUE)-2;                                              LEXO1630
      VALUE=SUBSTR(VALUE,2,END);                                        LEXO1640
      END;                                                              LEXO1650
```

```
    END;                                                                LEX01660
                                                                        LEX01670
  ELSE VALUE='';                                                        LEX01680
                                                                        LEX01690
  PRED.VALUE(TN)=VALUE;                                                 LEX01700
  PRED.VLEN(TN)=LENGTH(VALUE);                                          LEX01710
                                                                        LEX01720
  /* REGISTER NAME*/                                                   LEX01730
  T1.ANAME(TN)=DATA;                                                    LEX01740
  T1.PARENT(TN)=ST(STP);                                                LEX01750
  MOP(TN) = CURRENT_MOP;                                                LEX01760
  RETURN;                                                               LEX01770
                                                                        LEX01780
  END /*T1P*/;                                                          LEX01790
  /*******************************************************************/ LEX01800
                                                                        LEX01810
  END /*LEXT*/;                                                        LEX01820
```

```
MINIT: PROC;                                                          MIN00010
                                                                      MIN00020
/*********************************************************************  MIN00030
*                    MODULE    DESCRIPTION                          *  MIN00040
*********************************************************************/ MIN00050
/****   PURPOSE:                                                      MIN00060
** A T-PROC AT THE MM LEVEL WHICH INITIALIZES THIS LEVEL AND OBTAINS*  MIN00070
** THE KEY FOR THE UPPER LEVEL IF DATA ALREADY EXISTS IN SH.       *  MIN00080
** (FOR INTEGRATED VERSION: MAKE USE OF SUBROUTINE PAKT)           *  MIN00090
*****                                                                 MIN00100
*********************************************************************  MIN00110
*****   METHOD:                                                       MIN00120
*****     1. IF IT IS INIT REQUEST AND IT IS NEW INIT (I.E., SH DOES* MIN00130
*****        NOT CONTAIN ANY DATA YET), THEN AVAIL_LOC WHICH INDI-  * MIN00140
*****        CATES THE STARTING LOCATION OF FREE VIRTUAL MEMORY AND * MIN00150
*****        IS STORED AT THE FIRST PACKET OF SH IS INITIALZED TO   * MIN00160
*****        BE 16 BY CALLING SUBROUTIN PAKT.                       * MIN00170
*****     2. IF IT IS INIT REQUEST AND IT IS FILE INIT (I.E., SH AL-* MIN00180
*****        READY EXISTS WITH DATA) THE KEY FOR THE UPPER LEVEL*     MIN00190
*****        (4 BYTES) IS READ IN FROM VIRTUAL MEMORY LOCATION 5    * MIN00200
*****        AND THEN GIVEN TO THE UPPER LEVEL                      * MIN00210
*****     3. IF IT IS STORE KEY REQUEST THEN THE KEY IS STORED AT   * MIN00220
*****        VM LOC 5.                                              * MIN00230
*****                                                               * MIN00240
*********************************************************************  MIN00250
*****   INPUT PARAMETERS:  AS INDICATED BY AMINIT.                    * MIN00260
*****                                                                 * MIN00270
*********************************************************************/ MIN00280
                                                                      MIN00290
%INCLUDE DSEU1,EPAKT,SERVICE, FDEBUG,AMINI;                           MIN00300
DCL KEY BIT (32);                                                     MIN00310
DCL (P,RTP) PTR, (NAME,DATA) CHAR (8), OP CHAR (1), DT4 CHAR(4);      MIN00320
DCL (NULL,BIN,UNSPEC,SUSBSTR) BUILTIN;                                MIN00330
                                                                      MIN00340
CALL TBEG ( PROC_ADDR, P);                                            MIN00350
ALLOCATE MINIT_RTN SET (RTP);                                         MIN00360
OP=MINIT_ARG.OP;                                                      MIN00370
NAME=MINIT_ARG.FNAME;                                                 MIN00380
                                                                      MIN00390
SELECT (OP);                                                          MIN00400
  WHEN ('I') DO; /*INIT**/                                            MIN00410
                                                                      MIN00420
     IF NAME=(8)' ' THEN DO; /*NEW INIT*/                             MIN00430
                         /*SET STARTING LOCATION OF AVAILABLE         MIN00440
                           VIRTUAL MEMORY AT 16: 4 FOR KEY, 4 FOR     MIN00450
                           AVAIL_LOC, 8 FOR EXPANSION      */         MIN00460
        UNSPEC(DT4) = UNSPEC(BIN(16,31));                             MIN00470
        SUBSTR(DATA,1,4) = DT4;                                       MIN00480
        CALL PAKT (0,DATA,'W');                                       MIN00490
        GO TO RTN;                                                    MIN00500
        END;                                                          MIN00510
                                                                      MIN00520
     ELSE DO;                    /* FILE INIT */                      MIN00530
        CALL PAKT (0,DATA,'R');     /* GET FIRST PACKET IN THE SH */  MIN00540
        DT4 = SUBSTR(DATA,5,4); /* GET THE 2ND HALF OF THE PACKET */  MIN00550
```

```
         KEY = UNSPEC (DT4);      /* GET KEY */                    MIN00560
         RTP->MINIT_RTN.KEY=KEY;                                   MIN00570
         GO TO RTN;                                                MIN00580
         END;                                                      MIN00590
                                                                   MIN00600
      END /*WHEN I*/;                                              MIN00610
                                                                   MIN00620
   WHEN ('S') DO; /*PROCESS STORE KEY REQUEST*/                    MIN00630
      CALL PAKT (0,DATA,'R');       /* GET 1ST PACKET IN THE SH */ MIN00640
      UNSPEC(DT4) = MINIT_ARG.KEY;                                 MIN00650
      SUBSTR(DATA,5,4) = DT4;       /* UPDATE IT WITH KEY */       MIN00660
      CALL PAKT (0,DATA,'W');                                      MIN00670
      END /*WHEN S*/;                                             MIN00680
                                                                   MIN00690
   END /*SELECT */;                                                MIN00700
RTN:  CALL TRTN( PROC_ADDR,RTP);                                   MIN00710
                                                                   MIN00720
      RETURN;                                                      MIN00730
      END /*MINIT*/;                                               MIN00740
```

204

```
MODN: PROC(MP,OFF_NODE,RTN_CODE);                                       MOD00010
                                                                        MOD00020
/*A S-PROC AT THE N-ARY LEVEL USED AS A SUBROUTINE CALLED BY THE        MOD00030
*UPDN PROGRAM TO PERFORM MOD OP AT ROOT NODE*/                          MOD00040
                                                                        MOD00050
%INCLUDE AUPDN;                                                         MOD00060
%INCLUDE ECRTB,EDELB,EREPB,ESRCH;                                       MOD00070
                                                                        MOD00080
DCL (MP,P) PTR, (N,RTN_CODE) FIXED BIN,                                 MOD00090
  OFF_NODE(25) BIT (1) CONNECTED,                                       MOD00100
  DATA BIT (320) VARYING, (PS,ID,BS) BIT (32), (SUBSTR,NULL)BUILTIN;    MOD00110
                                                                        MOD00120
/*DO P=MP REPEAT UPDN_ARG.PTR WHILE(P^=NULL);                           MOD00130
     PUT SKIP LIST('MODN ARG LIST',UPDN_ARG); END;  */                  MOD00140
                                                                        MOD00150
P=MP;                                                                   MOD00160
/*LOCATE ROOT NODE ID*/                                                 MOD00170
PS=UPDN_ARG.PSETID;                                                     MOD00180
DATA=SUBSTR(UPDN_ARG.DATA,1,UPDN_ARG.DLEN * 8);                         MOD00190
CALL SRCH(PS,DATA,ID,RTN_CODE);                                         MOD00200
                                                                        MOD00210
/*FOLLOW THE TREE*/                                                     MOD00220
DO P=UPDN_ARG.PTR REPEAT UPDN_ARG.PTR WHILE (P^=NULL);                  MOD00230
   IF OFF_NODE(UPDN_ARG.NODE)='0'B THEN DO;                             MOD00240
      BS=UPDN_ARG.BSETID;                                               MOD00250
      DATA=SUBSTR(UPDN_ARG.DATA,1,UPDN_ARG.DLEN * 8);                   MOD00260
      SELECT (UPDN_ARG.OP);                                             MOD00270
                                                                        MOD00280
        WHEN (UPDNOP.IST) CALL CRTB(PS,BS,ID,DATA,'A',RTN_CODE);        MOD00290
        WHEN(UPDNOP.CI) CALL CRTB (PS,BS,ID,DATA,'N',RTN_CODE);         MOD00300
        WHEN (UPDNOP.DEL) CALL DELB (PS,BS,ID,'A',RTN_CODE);            MOD00310
        WHEN (UPDNOP.DD) CALL DELB (PS,BS,ID,'N',RTN_CODE);             MOD00320
        WHEN (UPDNOP.REP)                                               MOD00330
             CALL REPB (PS,BS,ID,DATA,'A',RTN_CODE);                    MOD00340
       WHEN (UPDNOP.REPP)                                               MOD00350
             CALL REPB (PS,BS,ID,DATA,'N',RTN_CODE);                    MOD00360
                                                                        MOD00370
       END /*SELECT*/;                                                  MOD00380
       END /*DO IF*/;                                                   MOD00390
     END /*DO*/;                                                        MOD00400
  END /*MODN*/; /*RTN_CODE IS SET TO RTN FROM SUBROUTINE CALLS*/        MOD00410
```

```
MPST: PROC(MAP,POS);                                                    MPS00010
                                                                        MPS00020
/*S-PROC AT THE N-ARY LEVEL USED BY DEFP AND DEFB FOR ASSIGNING         MPS00030
*ID SLOTS IN THE ID ARRAY****/                                          MPS00040
                                                                        MPS00050
DCL MAP BIT (32), (POS,I) FIXED BIN;                                    MPS00060
DCL SUBSTR BUILTIN;                                                     MPS00070
                                                                        MPS00080
DO I=32 TO 1 BY -1;                                                     MPS00090
  IF SUBSTR(MAP,I,1)='0'B THEN DO;                                      MPS00100
   SUBSTR(MAP,I,1)='1'B;                                                MPS00110
   POS=I;                                                               MPS00120
   RETURN;                                                              MPS00130
   END;                                                                 MPS00140
END;                                                                    MPS00150
POS=0; /*NOT AVAILABLE*/                                                MPS00160
RETURN;                                                                 MPS00170
END /*MPST*/;                                                           MPS00180
```

```
NINIT: PROC;                                                          NIN00010
                                                                      NIN00020
/* A T-PROC AT THE N-ARY LEVEL FOR INITIATING THE CATALOGUE STRUC     NIN00030
** AT THE N-ARY LEVEL *******/                                        NIN00040
                                                                      NIN00050
%INCLUDE ANINI,SERVICE,FDEBUG,AMINI,DNKEY,DPCAT,DBEU,ADEFP;           NIN00060
%INCLUDE ESUB1,EMPST;                                                 NIN00070
DCL (P,RP,TP,TP1,MP,RTP) PTR,(NULL,STRING,UNSPEC,ADDR) BUILTIN;       NIN00080
DCL AKEY BIT (32) STATIC , NKEY(3) BIT (32);                          NIN00090
DCL MAP BIT (32), POS FIXED BIN, (I,J) FIXED BIN, ID BIT (32);        NIN00100
DCL RTN_CODE FIXED BIN;                                               NIN00110
DCL BIT_STRING BIT (144) BASED;                                       NIN00120
/*BEGIN PROCESSING*/                                                  NIN00130
 CALL TBEG (PROC_ADDR, P);   MP=P;                                    NIN00140
 ALLOCATE NINIT_RTN SET (RTP);                                        NIN00150
 RTP->NINIT_RTN.RTN_CODE=0;                                           NIN00160
                                                                      NIN00170
 SELECT (NINIT_ARG.OP);                                               NIN00180
                                                                      NIN00190
   WHEN ('I') DO; /**'I' TYPE NINIT COMMAND****/                      NIN00200
      ALLOCATE MINIT_ARG  SET (RP);                                   NIN00210
      RP->MINIT_ARG.OP='I';                                           NIN00220
      RP->MINIT_ARG.FNAME=NINIT_ARG.FNAME;                            NIN00230
      CALL TCALL ('MINIT',1,RP,P);                                    NIN00240
      CALL SVC3(RP);                                                  NIN00250
                                                                      NIN00260
      /** PROCESS INIT AT THIS LEVEL***/                            NIN00270
      IF MP->NINIT_ARG.FNAME^=(8)' ' THEN DO; /*FILE INIT**/          NIN00280
         AKEY=MINIT_RTN.KEY; /**OBTAIN KEY FROM MINIT RETURN***/      NIN00290
         CALL SVC3 (P);                                               NIN00300
         CALL RET1(AKEY,P,RTN_CODE);/*CONTENT OF ANCHOR KEY*/         NIN00310
         STRING (NKEY)=BEU.DATA;                                      NIN00320
         FREE BEU;                                                    NIN00330
         PCATID=NKEY(1); /**OBTAIN PCATID*****/                       NIN00340
         BCATID=NKEY(2); /*OBTAIN BCATID******/                       NIN00350
         RTP->NINIT_RTN.KEY=NKEY(3); /*ALSO RETURN KEY TO UPPER LVL*/ NIN00360
         CALL TRTN(PROC_ADDR, RTP); /*RETURNS*/                       NIN00370
         RETURN;                                                      NIN00380
         END; /*FILE INIT*/                                           NIN00390
                                                                      NIN00400
      IF MP->NINIT_ARG.FNAME=(8)' ' THEN DO; /*NEW INIT*/             NIN00410
         /*INIT PCAT*/                                               NIN00420
         CALL SVC3(P);                                                NIN00430
         PINFO.NUMPTR=4;                                              NIN00440
         PINFO.PLEN=18; /*18 BYTES LONG*/                             NIN00450
         PINFO.PTYPE='8';                                             NIN00460
         PINFO.LTYPE=PINFOLTYPE.LL; /*LINK LIST*/                     NIN00470
         PINFO.L_ID=UNSPEC (NULL); /*INIT TO NULL*/                   NIN00480
         /**CALL UP MPST TO OBTAIN L_POS'S**/                         NIN00490
         MAP='00001111111111111111111111111111'B;                     NIN00500
         CALL MPST (MAP,POS);                                         NIN00510
         PINFO.L_POS=POS;                                             NIN00520
         CALL MPST(MAP,POS);                                          NIN00530
         PINFO.L_POS2=POS;                                            NIN00540
         PINFO.MAP=MAP;                                               NIN00550
```

```
                                                                    NIN00560
        /*STREAMLINE PINFO INTO BIT STRING*/                        NIN00570
        /*CREATE THIS PCAT CAT ENTRY***/                            NIN00580
        I=4; /*4 IDS*/                                              NIN00590
        J=18*8;                                                     NIN00600
        ALLOCATE BEU;                                               NIN00610
        BEU.DATA=ADDR(PINFO)->BIT_STRING;                           NIN00620
        CALL CRT1 (ID,P, RTN_CODE);                                 NIN00630
        PCATID=ID;                                                  NIN00640
        FREE BEU;                                                   NIN00650
                                                                    NIN00660
       /*INITIALIZE BCAT AS A PSET*/                               NIN00670
       ALLOCATE DEFP_ARG SET (TP);                                  NIN00680
       TP->DEFP_ARG.PTYPE='B';                                      NIN00690
       TP->DEFP_ARG.PLEN=11;                                        NIN00700
       TP->DEFP_ARG.IMP.HOW=DEFPHOW.SA;                             NIN00710
       TP->DEFP_ARG.IMP.NUMPTR=4;                                   NIN00720
                                                                    NIN00730
       CALL TCALL ('DEFP',1, TP,TP1);                               NIN00740
       BCATID=TP1->DEFP_RTN.PSETID;                                 NIN00750
       CALL SVC3 (TP);                                              NIN00760
       CALL SVC3 (TP1);                                             NIN00770
                                                                    NIN00780
                                                                    NIN00790
       NKEY(1)=PCATID;                                              NIN00800
       NKEY(2)=BCATID;                                              NIN00810
       NKEY(3)=UNSPEC (NULL); /*RESERVED*/                          NIN00820
       I=1; J=96;  ALLOCATE BEU;                                    NIN00830
        BEU.DATA=STRING(NKEY) ;                                     NIN00840
       CALL CRT1(AKEY,P,RTN_CODE); /*OBTAINS ANCHOR KEY(AKEY)*/     NIN00850
       FREE BEU;                                                    NIN00860
                                                                    NIN00870
       /*STORE KEY BY CALLING MINIT*/                              NIN00880
       ALLOCATE MINIT_ARG;                                          NIN00890
       MINIT_ARG.OP='S';  MINIT_ARG.KEY=AKEY;                       NIN00900
       CALL TCALL ('MINIT',1,P,TP);                                 NIN00910
                                                                    NIN00920
       /*RETURNS**/                                                NIN00930
       CALL SVC3 (P);                                               NIN00940
       CALL SVC3 (TP);                                              NIN00950
       CALL TRTN (PROC_ADDR, RTP);                                  NIN00960
       RETURN;                                                      NIN00970
       END /*NEW INIT*/;                                            NIN00980
    END /*WHEN'I'*/;                                               NIN00990
WHEN ('S') DO; /*'S' OP FOR NINIT*/                                 NIN01000
    NKEY(1)=PCATID;                                                 NIN01010
    NKEY(2)=BCATID;                                                 NIN01020
    NKEY(3)=NINIT_ARG.KEY;                                          NIN01030
    I=1; J=96; ALLOCATE BEU; BEU.DATA=STRING(NKEY);                 NIN01040
    IF AKEY^=UNSPEC (NULL) THEN                                     NIN01050
         CALL REP1(AKEY,P,RTN_CODE);                                NIN01060
        ELSE CALL CRT1(AKEY,P,RTN_CODE);                            NIN01070
   FREE BEU;                                                        NIN01080
   CALL TRTN(PROC_ADDR, RTP);                                       NIN01090
    END /*WHEN 'S'*/;                                               NIN01100
```

```
                                                                    NINO1110
  END /*SELECT*/;                                                   NINO1120
                                                                    NINO1130
  END /*NINIT*/;                                                    NINO1140
                                                                    NINO1150
```

```
PAKT:PROC(VA,DATA,RW);                                                  PAK00010
/*** THIS IS A SUBROUTINE AT THE MM LEVEL WHICH RETRIEVES AN 8-BYTE     PAK00020
 *** PACKET (DATA) GIVEN ITS VIRTUAL ADDRESS (VA) IF RW IS 'R', OR      PAK00030
 *** WRITES AN 8-BYTE PACKET GIVEN ITS VA IF RW IS 'W'.  ****          PAK00040
 *** IT INTERFACES DIRECTLY WITH THE DSH STV ************************/  PAK00050
                                                                        PAK00060
/*** DUE TO INCOMPATIBILITY OF FORMAT OF 'ARG' TO THE SET OF SERVICE   PAK00070
 *** ROUTINES COMMONLY USED BY OTHER FH MODULES, CONTROL STRUCTURE     PAK00080
 *** IS DIRECTLY ACCESSED FROM PAKT, AND TIMING CODE IS DUPLICATED     PAK00090
 *** HERE TO ACCUMULATE STATISTICS OF MODULE L1 WHICH IS NORMALLY      PAK00100
 *** PERFORMED BY SERVICE ROUTINE TCALL ***************************/   PAK00110
                                                                        PAK00120
%INCLUDE USERS,VPX,PFSVC,ARG;                                           PAK00130
DCL VA FIXED BIN(31), DATA CHAR(8), RW CHAR(1);                         PAK00140
                                                                        PAK00150
J=8; ALLOC ARG;                        /* PREPARE ARG TO COMM. WITH DSH */ PAK00160
ARG.VIRTUAL_ADDRESS = VA;                                               PAK00170
ARG.VPID = THISVP->VP.VPID;                                             PAK00180
ARG.BOXID = 1;                                                          PAK00190
IF RW = 'R' THEN ARG.REQ_TYPE = 'READ';                                 PAK00200
ELSE DO;                                                                PAK00210
   ARG.REQ_TYPE = 'WRITE';                                              PAK00220
   ARG.DATA = DATA;                                                     PAK00230
   END;                                                                 PAK00240
                                                                        PAK00250
ALLOC PF_SVC;                          /* PREPARE FOR SEND PROTOCOL */  PAK00260
PF_SVC.SVC = 'L1';                                                      PAK00270
PF_SVC.PTR = ARGPTR;                                                    PAK00280
                                                                        PAK00290
/******************** TIMING REPORT OPTION ************************/   PAK00300
DCL FTIME BIT(1) EXT;                  /* FTIME IS TIMING REPORT OPTION */ PAK00310
DCL (BEGTIME,ENUM,LOTIME CTL) FIXED BIN(31);                            PAK00320
DCL RTIMER ENTRY RETURNS(FIXED BIN(31));                                PAK00330
DCL (ECOUNT(50),ELO(50),ETIME(50)) FIXED BIN(31) EXT INIT((50)0);       PAK00340
IF FTIME THEN DO;                                                       PAK00350
ALLOCATE LOTIME; LOTIME=0;                                              PAK00360
BEGTIME=RTIMER;        END;                                             PAK00370
/*******************************************************************/   PAK00380
                                                                        PAK00390
CALL SEND (1,0,1,'S',20,PF_SVC);                                        PAK00400
CALL WAIT(1);                                                           PAK00410
                                                                        PAK00420
 /********************** TIMING REPORT OPTION **********************/   PAK00430
 IF FTIME THEN DO;                                                      PAK00440
 BEGTIME=RTIMER-BEGTIME;                                                PAK00450
 ENUM = 19;                /* L1 IS THE 19TH PROC NAME TO BE TRACED */  PAK00460
 ETIME(ENUM)=ETIME(ENUM)+BEGTIME;/* ACCUMULATE TOTAL TIME */            PAK00470
 ECOUNT(ENUM)=ECOUNT(ENUM)+1;/* ACCUMULATE COUNT */                     PAK00480
 ELO(ENUM)=ELO(ENUM)+LOTIME;/* ACCUMULATE LOWER LEVEL PROC TIME */      PAK00490
 FREE LOTIME; IF ALLOCATION(LOTIME)^= 0 THEN LOTIME=LOTIME+BEGTIME;     PAK00500
 END;                                                                   PAK00510
/*******************************************************************/   PAK00520
ARGPTR = VP.WAIT.MSG;          /* OBTAIN RETURN ARG */                  PAK00530
IF RW = 'R' THEN DATA = ARG.DATA;                                       PAK00540
FREE ARG;                                                               PAK00550
```

```
RETURN;                                                    PAK00560
END;                                                       PAK00570
```

```
PRNT: PROC (P1,P2,RTN_CODE);                                          PRN00010
                                                                      PRN00020
/* S-PROC AT THE USER INTERFACE LEVEL, CALLED BY DMQ. THIS MODULE WILL PRN00030
 FORMAT OUTPUT OF A RETE CALL. IT IS PASSED A POINTER POINTING TO THE  PRN00040
 RETE_ARG CHAIN AND A POINTER POINTING TO THE RETE_RTN CHAIN */        PRN00050
                                                                      PRN00060
%INCLUDE ARETE;                                                       PRN00070
DCL 1 T(25), /* TEMP STRUCTURE */                                     PRN00080
      2 NAME CHAR (8) INIT ((25) (8)' '),                             PRN00090
      2 PARENT FIXED BIN INIT((25)0),                                 PRN00100
      2 MLEN FIXED BIN,                                               PRN00110
     (2 POS, 2 LINE) FIXED BIN (31);                                  PRN00120
                                                                      PRN00130
DCL HEADER(2) CHAR (80) VARYING;                                      PRN00140
DCL TOKEN CHAR (20) VARYING;                                          PRN00150
DCL STR (4) CHAR (80) VARYING;                                        PRN00160
DCL (I,J,K,L,CURRENT_PARENT,LAST,SUM,NN,I1,I2,I3) FIXED BIN (31);     PRN00170
DCL RTN_CODE FIXED BIN;                                               PRN00180
DCL (P1,P2,P) PTR, (NULL,SUBSTR,MAX) BUILTIN;                         PRN00190
DCL NODE FIXED BIN;                                                   PRN00200
DCL ST (10) FIXED BIN, STP FIXED BIN;                                 PRN00210
                                                                      PRN00220
/* BEGIN */                                                           PRN00230
                                                                      PRN00240
PUT SKIP LIST ('ENTITY SET NAME', P1->RETE_ARG.NAME);                 PRN00250
                                                                      PRN00260
/*BUILD T STRUCTURE */                                                PRN00270
DO P=P1 REPEAT RETE_ARG.PTR WHILE (P^=NULL);                          PRN00280
   NN=RETE_ARG.NODE; /* NN IS BOTH INDEX TO T AND NODE # */           PRN00290
   T.NAME(NN) = RETE_ARG.NAME;                                        PRN00300
   T.PARENT(NN) = RETE_ARG.PARENT;                                    PRN00310
   END;                                                               PRN00320
DO P=P2->RETE_RTN.PTR REPEAT RETE_RTN1.PTR WHILE (P^=NULL);           PRN00330
   NN=RETE_RTN1.NODE;                                                 PRN00340
   T.MLEN(NN)=RETE_RTN1.DLEN; /*NOTE IN CURRENT VERSION, MLEN A       PRN00350
                          DLEN OF A DATA ELEMENT IS THE SAME */       PRN00360
   END;                                                               PRN00370
                                                                      PRN00380
/*HEADER */                                                           PRN00390
HEADER(1)=''; HEADER(2) = ''; LAST=99; STP=1; ST(STP) =1; L=1;        PRN00400
DO J=2 TO 25;                                                         PRN00410
   IF T.NAME(J) ^= (8)' ' THEN DO;                                    PRN00420
      TOKEN = '';                                                     PRN00430
      SELECT (T.PARENT(J));                                           PRN00440
        WHEN (LAST) DO;                                               PRN00450
           TOKEN = TOKEN||' ('||T.NAME(J);                            PRN00460
           STP = STP +1;                                              PRN00470
           ST(STP) = T.PARENT(J);                                     PRN00480
           END;                                                       PRN00490
        WHEN (ST(STP)) TOKEN = TOKEN||' ' | '||T.NAME(J);             PRN00500
        OTHERWISE DO;                                                 PRN00510
           DO WHILE (ST(STP) ^= T.PARENT(J));                         PRN00520
              STP=STP-1;                                              PRN00530
              TOKEN = TOKEN||')';                                     PRN00540
              END;                                                    PRN00550
```

```
            TOKEN = TOKEN||' ' ||T.NAME(J);                          PRN00560
            END;                                                     PRN00570
          END /* SELECT */;                                          PRN00580
                                                                     PRN00590
        LAST=J;                                                      PRN00600
        IF LENGTH(HEADER(L))+LENGTH(TOKEN) > 80                      PRN00610
        THEN DO;                                                     PRN00620
           L = L + 1;                                                PRN00630
           HEADER(L)= TOKEN;                                         PRN00640
           END;                                                      PRN00650
        ELSE HEADER(L) = HEADER(L)||TOKEN;                           PRN00660
        END /* IF */;                                               PRN00670
     END /*DO J */;                                                  PRN00680
     DO WHILE (STP ^= 1); /* FIX UP LAST TOKEN */                    PRN00690
        STP=STP-1;                                                   PRN00700
        HEADER(L) = HEADER(L)||')';                                  PRN00710
        END;                                                         PRN00720
     HEADER(L) = HEADER(L)||' |';                                    PRN00730
     DO I = 1 TO L;                                                  PRN00740
        PUT SKIP LIST (HEADER(I));                                   PRN00750
        END;                                                         PRN00760
     PUT SKIP;                                                       PRN00770
                                                                     PRN00780
     /* DATA */                                                      PRN00790
    SUM=3; L=1;                                                      PRN00800
    DO J=2 TO 25;                                                    PRN00810
      IF T.NAME(J)^=(8)' ' THEN DO;                                  PRN00820
        T.POS(J)=SUM+1;                                              PRN00830
        SUM=SUM+T.MLEN(J)+3;                                         PRN00840
        IF SUM > 80 THEN DO;                                         PRN00850
           L=L+1;                                                    PRN00860
           T.POS(J)=1;                                               PRN00870
           SUM=T.MLEN(J)+1;                                          PRN00880
           END;                                                      PRN00890
        T.LINE(J)=L;                                                 PRN00900
        END;                                                         PRN00910
     END /*DO J */;                                                  PRN00920
                                                                     PRN00930
    /* PRINT DATA */                                                PRN00940
    DO I = 1 TO L; /* INIT STR */                                    PRN00950
       STR(I)=(100)' ';                                              PRN00960
       END;                                                          PRN00970
    SUBSTR(STR(1),1,3)=' | ';                                        PRN00980
                                                                     PRN00990
    DO P=P2->RETE_RTN.PTR->RETE_RTN1.PTR REPEAT RETE_RTN1.PTR WHILE  PRN01000
            (P^=NULL);                                               PRN01010
      NODE=RETE_RTN1.NODE;                                           PRN01020
      IF NODE=1                                                      PRN01030
      THEN DO;                                                       PRN01040
         DO I = 1 TO L;                                              PRN01050
             PUT SKIP LIST (STR(I));                                 PRN01060
             STR(I)=(100)' ';                                        PRN01070
             END;                                                    PRN01080
         SUBSTR(STR(1),1,3)=' | ';                                   PRN01090
         END;                                                        PRN01100
```

```
   ELSE  IF RETE_RTN1.DLEN ^= 0                                          PRNO1110
         THEN DO;                                                        PRNO1120
    I1=T.LINE(NODE);                                                     PRNO1130
    I2=T.POS(NODE);                                                      PRNO1140
    I3=T.MLEN(NODE);                                                     PRNO1150
         SUBSTR(STR(I1),I2,I3)                                           PRNO1160
          =SUBSTR(RETE_RTN1.CDATA,1,RETE_RTN1.DLEN);                     PRNO1170
         SUBSTR(STR(I1),I2+I3,3) = ' | ';                                PRNO1180
        END;                                                             PRNO1190
END /* DO P */;                                                         PRNO1200
    DO I= 1 TO L; /*WRITE OUT THE LAST OCCURRENCE */                     PRNO1210
          PUT SKIP LIST (STR(I));                                        PRNO1220
       END;                                                              PRNO1230
                                                                         PRNO1240
END /* PRNT */;                                                         PRNO1250
```

```
PSEU: PROC (ID,SEUPTR,RTN_CODE);                                        PSE00010
                                                                        PSE00020
/*** THIS IS A SUBROUTINE AT THE MM LEVEL WHICH PUTS AN SEU ACCORDING  PSE00030
 *** TO ITS PREASSIGNED ID INTO THE STORAGE AREA.  IT DECOMPOSES AN     PSE00040
 *** SEU INTO 8-BYTE PACKTES AND CALLS PAKT TO WRITE THEM INTO DSH */   PSE00050
                                                                        PSE00060
%INCLUDE DSEU1,EPAKT;                                                   PSE00070
DCL DATA CHAR(8), VA FIXED BIN(31), (TP,BP) PTR;                        PSE00080
DCL (I,RTN_CODE) FIXED BIN, BASED_DATA CHAR (8) BASED (BP);             PSE00090
DCL (NULL,ADDR,SUBSTR,MIN,UNSPEC) BUILTIN, (L,L1,L2) FIXED BIN(31);     PSE00100
DCL ID BIT(32);                                                         PSE00110
                                                                        PSE00120
RTN_CODE = 0;                                                           PSE00130
UNSPEC(VA) = ID;                                                        PSE00140
DATA = SEUPTR -> BASED_DATA;                                            PSE00150
CALL PAKT (VA,DATA,'W');          /* WRITE FIRST 8 BYTES OF THIS SEU */ PSE00160
                                  /* ESTABLISH THE REST OF SEU */       PSE00170
L = (SEU.N + 2)/8;                /* L = # OF PACKETS IN THIS SEU -1 */ PSE00180
DO I = 1 TO L;                    /* PUT ALL PACKETS           */       PSE00190
   L1 = I*8-2; L2 = MIN(8,SEU.N-L1+1);                                  PSE00200
   DATA = SUBSTR (SEU.DATA,L1,L2);                                      PSE00210
   CALL PAKT (VA+I*8,DATA,'W');                                         PSE00220
   END;                                                                 PSE00230
                                                                        PSE00240
RETURN;                                                                 PSE00250
END;                                                                    PSE00260
```

```
REP:PROC;                                                                    REP00010
                                                                             REP00020
/*T-PROC AT THE MM LEVEL TO REPLACE A SEU GIVEN ITS ID*****/                 REP00030
/* FOR INTEGRATED VERSION: MAKE USE OF SUBROUTINES GSEU & PSEU */            REP00040
                                                                             REP00050
%INCLUDE SERVICE, AREP, DSEU1, EGSEU, EPSEU;                                 REP00060
DCL (P,RTP)PTR;                                                              REP00070
DCL (I,RTN_CODE) FIXED BIN;                                                  REP00080
DCL (NULL, UNSPEC) BUILTIN;                                                  REP00090
                                                                             REP00100
 /*BEGIN*/                                                                   REP00110
CALL TBEG (PROC_ADDR, P);                                                    REP00120
ALLOCATE REP_RTN SET (RTP);                                                  REP00130
RTP->REP_RTN.RTN_CODE =0;                                                    REP00140
                                                                             REP00150
CALL GSEU (REP_ARG.ID, SEUPTR, RTN_CODE); /*GET SEU DESIGNATED BY ID*/       REP00160
IF RTN_CODE ^= 0                                                             REP00170
THEN RTP->REP_RTN.RTN_CODE = 1;           /* ILLEGAL SEU */                  REP00180
ELSE DO;                                                                     REP00190
   SEU.N=REP_ARG.N/8;                                                        REP00200
   UNSPEC(SEU.DATA) = REP_ARG.DATA;                                          REP00210
   CALL PSEU (REP_ARG.ID, SEUPTR, RTN_CODE);                                 REP00220
   END /*ELSE DO*/;                                                          REP00230
                                                                             REP00240
CALL TRTN(PROC_ADDR, RTP);                                                   REP00250
RETURN;                                                                      REP00260
END /*REP*/;                                                                 REP00270
```

```
REPB: PROC (PARENT,BSETID, ID1,DATA,MODE,RTN_CODE);                    REP00010
                                                                       REP00020
%INCLUDE DBCAT, DBEU;                                                  REP00030
%INCLUDE EBCT,ESRCH,ECRTP,ESUB1;                                       REP00040
                                                                       REP00050
DCL (PARENT, BSETID, ID1,ID2) BIT (32), DATA BIT (*);                  REP00060
DCL RTN_CODE FIXED BIN, MODE CHAR (1), P PTR, NULL BUILTIN;            REP00070
                                                                       REP00080
CALL GBCT (BSETID, BINFO, RTN_CODE);                                   REP00090
CALL RET1 (ID1,P,RTN_CODE); /* GET SOURCE OUT */                       REP00100
ID2 = BEU.ID_ARRAY (BINFO.POS); /* GET OLD TARGET */                   REP00110
SELECT (MODE);                                                         REP00120
                                                                       REP00130
  WHEN ('N') DO;/* NODE MODE OF REPLACE OF BINARY ASSOC*/              REP00140
    IF BINFO.FUNC='S' THEN DO; /*REPLACE IT*/                          REP00150
      IF ID2 ^= UNSPEC (NULL) THEN                                     REP00160
                                                                       REP00170
        CALL REPP (BINFO.PSETID(2), DATA, ID2, RTN_CODE);              REP00180
      ELSE CALL CRTP (BINFO.PSETID(2), DATA, ID2, RTN_CODE);           REP00190
      END /* IF 'S' */;                                               REP00200
                                                                       REP00210
    ELSE DO; /* WHEN FUNC IS MULTIPLE SEE IF DATA ALREADY EXISTS */    REP00220
        CALL SRCH(BINFO.PSETID(2),DATA,ID2,RTN_CODE);                  REP00230
        IF RTN_CODE=1 /*NOT FOUND*/                                    REP00240
          THEN CALL CRTP(BINFO.PSETID(2),DATA,ID2,RTN_CODE);           REP00250
       END;                                                            REP00260
  END /*WHEN N*/;                                                      REP00270
                                                                       REP00280
  WHEN ('A') DO; /*ARC MODE OF REPLACEMENT */                          REP00290
                                                                       REP00300
     CALL SRCH (BINFO.PSETID(2), DATA,ID2, RTN_CODE);                  REP00310
     IF RTN_CODE=1 THEN RETURN;/*RTNCODE 1 IS TARGET OF ARC NOT FND*/  REP00320
     END /*WHEN 'A'*/;                                                 REP00330
                                                                       REP00340
 END /*SLECT*/;                                                        REP00350
                                                                       REP00360
 /*ENTER ID2 INTO ID1 ID ARRAY*/                                       REP00370
 BEU.ID_ARRAY(BINFO.POS)=ID2;                                          REP00380
 CALL REP1 (ID1,P,RTN_CODE);                                           REP00390
 RTN_CODE=0;                                                           REP00400
 END /*REPB*/;                                                         REP00410
```

```
REPP: PROC (PSETID, DATA, ID, RTN_CODE);                                REP00010
                                                                        REP00020
/* A S-PROC AT THE N-ARY LEVEL USED TO REPLACE THE DATA OF A P-ELEMENT REP00030
 (DESIGNATED BY ID) IN A PSET (DESIGNATED BY PSETID) WITH NEW DATA      REP00040
 (GIVEN AS DATA); IT IS AN ERROR IF ID IS NULL OR IS BAD */             REP00050
                                                                        REP00060
%INCLUDE DBEU;                                                          REP00070
%INCLUDE ESUB1;                                                         REP00080
DCL (PSETID,ID) BIT (32), P PTR, RTN_CODE FIXED BIN;                    REP00090
DCL (NULL,UNSPEC) BUILTIN;                                              REP00100
DCL DATA BIT (*);                                                       REP00110
                                                                        REP00120
CALL RET1 (ID,P,RTN_CODE); /* GET SOURCE */                             REP00130
IF P = NULL | RTN_CODE ^= 0 THEN DO; RTN_CODE= 1 + RTN_CODE;            REP00140
                        RETURN; END; /*** 1+ IS BAD SOURCE */          REP00150
                                                                        REP00160
BEU.DATA = DATA; /* REPLACE DATA */                                     REP00170
CALL REP1 (ID,P,RTN_CODE);                                              REP00180
END /*REPP */;                                                          REP00190
```

```
REP1:PROC (ID,P,RTN_CODE);                                              DUM00010
                                                                        DUM00020
 /*A S-PROC AT THE N-ARY LEVEL FOR COMMUNICATING DIRECTLY WITH THE      DUM00030
 ** REP MODULE AT THE MM LEVEL*/                                        DUM00040
                                                                        DUM00050
%INCLUDE AREP, DBEU, SERVICE;                                           DUM00060
DCL ID BIT (32), (P,P1,BP) PTR,(RTN_CODE,I,J,K) FIXED BIN;              DUM00070
DCL (NULL,UNSPEC,STRING,SUBSTR) BUILTIN;                                DUM00080
DCL BIT_STRING BIT (800) BASED;                                         DUM00090
                                                                        DUM00100
/*FILL UP REP_ARG*/                                                     DUM00110
 ALLOCATE REP_ARG SET (P1);                                             DUM00120
 P1->REP_ARG.N=NUMPTR*32+DLEN+32;                                       DUM00130
 P1->REP_ARG.DATA = P->BIT_STRING;                                      DUM00140
 P1->REP_ARG.ID=ID;                                                     DUM00150
/*CALL UP REP*/                                                         DUM00160
CALL TCALL ('REP',1,P1,BP);                                             DUM00170
                                                                        DUM00180
/*************** MEMORY REQUEST TRACE ***********************/        DUM00190
IF FFF THEN CALL DUMPIT('REPLCED',ID,P); DCL FFF BIT(1) EXT;  /*/     DUM00200
/************************************************************/          DUM00210
                                                                        DUM00220
RTN_CODE=BP->REP_RTN.RTN_CODE;                                          DUM00230
CALL SVC3(P1); CALL SVC3(BP);                                           DUM00240
RETURN;                                                                 DUM00250
END;                                                                    DUM00260
```

```
RET: PROC;                                                              RET00010
                                                                        RET00020
/* A T-PROC AT THE MM LEVEL FOR RETRIEVAL OF BEU GIVEN ITS ID */       RET00030
/* FOR INTEGRATED VERSION: MAKE USE OF SUBROUTINE GSEU (GET SEU) */     RET00040
                                                                        RET00050
%INCLUDE ARET, SERVICE, DSEU1, EGSEU;                                   RET00060
DCL (P,RTP,TP) PTR;                                                     RET00070
DCL (UNSPEC,NULL) BUILTIN;                                              RET00080
DCL (I,RTN_CODE) FIXED BIN;                                             RET00090
                                                                        RET00100
CALL TBEG (PROC_ADDR,P);                                                RET00110
ALLOCATE RET_RTN SET (RTP);                                             RET00120
                                                                        RET00130
CALL GSEU (RET_ARG.ID,SEUPTR,RTN_CODE); /* GET SEU DESIGNATED BY ID */ RET00140
IF RTN_CODE ^= 0 THEN RTP->RET_RTN.RTN_CODE = 1;                        RET00150
ELSE DO;                                                                RET00160
  RTP->RET_RTN.N = SEU.N*8;                                             RET00170
  RTP->RET_RTN.DATA = UNSPEC(SEU.DATA);                                 RET00180
  END;                                                                  RET00190
                                                                        RET00200
CALL TRTN (PROC_ADDR,RTP);                                              RET00210
RETURN;                                                                 RET00220
END;                                                                    RET00230
```

224

```
RETB: PROC (PARENT,BINFO,PTR1,PTR2,RTN_CODE);                          RET00010
                                                                       RET00020
/* A S-PROC WITHIN N-ARY LEVEL WHICH OBTAINS THE ID OF THE BINARY      RET00030
* ASSOCIATED ELEMENT IN BSETID OF ID1 IN PARENT PSET*/                  RET00040
                                                                       RET00050
%INCLUDE OBCAT,ESUB1,DBEU;                                             RET00060
%INCLUDE EBCT;                                                         RET00070
DCL PARENT BIT(32) ALIGNED,                                            RET00080
 (RTN_CODE,POS) FIXED BIN;                                             RET00090
DCL (PTR1,PTR2) PTR,(NULL,UNSPEC) BUILTIN;                             RET00100
DCL ID2 BIT(32);                                                       RET00110
POS=BINFO.POS; /*GET PTR SLOT FOR BSET*/                               RET00120
                                                                       RET00130
IF PTR1=NULL THEN DO; RTN_CODE=10+RTN_CODE; RETURN; END;               RET00140
                                                                       RET00150
ID2=PTR1->BEU.ID_ARRAY(POS);                                           RET00160
                                                                       RET00170
IF ID2=UNSPEC(NULL)                                                    RET00180
THEN DO;                                                               RET00190
   PTR2 = NULL;                                                        RET00200
   RTN_CODE = 0;                                                       RET00210
   END;                                                                RET00220
ELSE CALL RET'(ID2,PTR2,RTN_CODE);  /*CHECK IF ID2 LEGAL*/             RET00230
RETURN;                                                                RET00240
END /*RETB*/;                                                          RET00250
```

```
RETE:PROC(                                                          RET00010
 /*T-PROC TO PROCESS RETE **/                                       RET00020
%INCLUDE ARETE,FDEBUG,SERVICE,ARETN;                                RET00030
%INCLUDE DECAT;                                                     RET00040
%INCLUDE EGECT,EGACT,EVTPE;                                         RET00050
                                                                    RET00060
DCL (P,TP,RP,LP,TP1,RTP,PP) PTR, NULL BUILTIN;                      RET00070
DCL(RTN_CODE,I,J) FIXED BIN, NUM F XED BIN (31);                    RET00080
DCL BDATA BIT (320), CDATA CHAR (40), DLEN FIXED BIN;               RET00090
DCL TO_BIT FIXED BIN INIT (1), TO_CHAR FIXED BIN INIT (2);          RET00100
DCL (ENAME,ANAME,ENAME1) CHAR (8);                                  RET00110
DCL 1 MAINFO (25), /*AN ARRAY OF AINFO TO STORE RETRIEVED ACAT ENT*/ RET00120
      2 BSETID BIT (32),  /*BSETID OF THIS ATTRIBUTE**/             RET00130
      2 PSETID BIT (32),  /*PSETID OF TARGET NODE**/               RET00140
      2 ENAME CHAR (8), /*IF TARGET IS A ENTITY SET*/              RET00150
      2 MAX FIXED BIN (31),                                         RET00160
      2 MIN FIXED BIN (31),                                         RET00170
      2 MLEN FIXED BIN,                                             RET00180
      2 FUNC CHAR(1),                                               RET00190
      2 TYPE CHAR (1),                                              RET00200
      2 VTYPE CHAR (1);                                             RET00210
DCL NODE FIXED BIN;                                                 RET00220
                                                                    RET00230
/*BEGIN PROCESSING**/                                               RET00240
CALL TBEG (PROC_ADDR,P);                                            RET00250
ALLOCATE RETE_RTN SET (RTP);                                        RET00260
RTP->RETE_RTN.RTN_CODE=0; /*INIT TO O.K.*/                          RET00270
ENAME=RETE_ARG.NAME;                                                RET00280
MAINFO.ENAME(1)=ENAME; /*INIT AINFO ARRAY'S FIRST TOKEN*/           RET00290
/**ALLOCATE ROOT NODE**/                                            RET00300
 CALL GECT (ENAME,EINFO,RTN_CODE); /*GECT GETS ENTITY CAT ENTRY */  RET00310
 IF RTN_CODE^=0 THEN DO;                                            RET00320
    RTP->RETE_RTN.RTN_CODE=100; /**100 IS NO SUCH ENTITY**/         RET00330
    CALL TRTN (PROC_ADDR,RTP);   /**RETURNS**/                      RET00340
    RETURN;                                                         RET00350
    END;                                                            RET00360
 I=0; /*NO PRED*/                                                   RET00370
 ALLOCATE RETN_ARG SET (TP);                                        RET00380
 RP,LP=TP;                                                          RET00390
 I=TP->RETN_ARG.CBTP; J=TP->RETN_ARG.LEN; PP=TP->RETN_ARG.PTR;      RET00400
 TP->RETN_ARG=RETE_ARG, BY NAME;                                    RET00410
 TP->RETN_ARG.CBTP=I; TP->RETN_ARG.LEN=J; TP->RETN_ARG.PTR=PP;      RET00420
                                                                    RET00430
 TP->RETN_ARG.PSETID=EINFO;                                         RET00440
                                                                    RET00450
/*FOLLOW RETE TREE **/                                              RET00460
DO P=RETE_ARG.PTR REPEAT P->RETE_ARG.PTR WHILE (P^=NULL);           RET00470
  NODE=RETE_ARG.NODE;                                               RET00480
  ENAME1=MAINFO.ENAME(RETE_ARG.PARENT);                             RET00490
  ANAME=RETE_ARG.NAME;                                              RET00500
  CALL GACT (ANAME,ENAME1,AINFO,RTN_CODE); /*GACT GETS ATTR CAT ENT*/ RET00510
  IF RTN_CODE^=0 THEN DO;                                           RET00520
    RTP->RETE_RTN.RTN_CODE=NODE+100; /*100+ IS NO SUCH ATTRIBUTE***/ RET00530
    CALL SVC3 (RP); /*FREE UP RETN ALLOCATION UP TO NOW*/           RET00540
    CALL TRTN (PROC_ADDR, RTP); /*RETURNS**/                       RET00550
```

```
     RETURN;                                                          RETO0560
     END;                                                             RETO0570
   MAINFO(NODE)=AINFO; /*ENTER INTO AINFO ARRAY*/                     RETO0580
                                                                      RETO0590
   /*VERIFY PREDICATE DATA ***/                                       RETO0600
   IF RETE_ARG.N^=0 THEN DO;                                          RETO0610
      CDATA=RETE_ARG.PRED.CDATA(1);                                   RETO0620
      DLEN=RETE_ARG.PRED.DLEN(1);                                     RETO0630
      CALL VTPE (AINFO,TO_BIT,BDATA,CDATA,DLEN,RTN_CODE);             RETO0640
      IF RTN_CODE^=0 THEN DO; /*PROBLEMS**/                           RETO0650
         RTP->RETE_RTN.RTN_CODE=NODE; /*1+ IS ILLEGAL NODE DATA*/      RETO0660
         CALL SVC3(RP); /*FREE UP RETN ALLOCATION UP TO NOW*/         RETO0670
         CALL TRTN (PROC_ADDR, RTP); /*RETURNS*/                     RETO0680
         RETURN;                                                      RETO0690
         END;                                                         RETO0700
      END;                                                            RETO0710
                                                                      RETO0720
  /*REPLACE ANAME WITH PSETID AND BSETID AND FIX UP DATA**/           RETO0730
  I=RETE_ARG.N;                                                       RETO0740
  ALLOCATE RETN_ARG SET (TP);                                         RETO0750
  LP->RETN_ARG.PTR=TP; LP=TP;                                         RETO0760
                                                                      RETO0770
  I=TP->RETN_ARG.CBTP; J=TP->RETN_ARG.LEN;  PP=TP->RETN_ARG.PTR;      RETO0780
  TP->RETN_ARG=RETE_ARG, BY NAME;                                     RETO0790
  TP->RETN_ARG.CBTP=I; TP->RETN_ARG.LEN=J;  TP->RETN_ARG.PTR=PP;      RETO0800
                                                                      RETO0810
  TP->RETN_ARG.PSETID=AINFO.PSETID;                                   RETO0820
  TP->RETN_ARG.BSETID=AINFO.BSETID;                                   RETO0830
  IF RETE_ARG.N ^=0 THEN DO;                                          RETO0840
    TP->RETN_ARG.PRED.DLEN(1)=DLEN;                                   RETO0850
    TP->RETN_ARG.PRED.DATA(1)=BDATA;                                  RETO0860
    END;                                                              RETO0870
  END./*DO P=RETE_ARG.PTR...*/;                                       RETO0880
                                                                      RETO0890
/*CALL UP RETN **/                                                    RETO0900
CALL TCALL ('RETN',1,RP,TP);                                          RETO0910
                                                                      RETO0920
/*CHECK RETURNS*/                                                     RETO0930
I=TP->RETN_RTN.RTN_CODE;                                              RETO0940
IF I^=0 THEN DO;                                                      RETO0950
   RTP->RETE_RTN.RTN_CODE=I+200; /*200+ IS RTN CODE FROM RETN**/      RETO0960
   CALL SVC3 (RP);                                                    RETO0970
   CALL SVC3 (TP);                                                    RETO0980
   CALL TRTN (PROC_ADDR, RTP);                                        RETO0990
     RETURN;                                                          RETO1000
   END;                                                               RETO1010
RTP->RETE_RTN.N=TP->RETN_RTN.N; /*COPY*/                              RETO1020
LP=RTP;                                                               RETO1030
                                                                      RETO1040
/*COPY RETN_RTN1 INTO RETE_RTN1 WHILE FIXING UP DATA **/             RETO1050
DO P=TP->RETN_RTN.PTR REPEAT P->RETN_RTN1.PTR WHILE (P^=NULL);        RETO1060
                                                                      RETO1070
/*   %INCLUDE HEX;*/                                                 RETO1080
/* PUT SKIP LIST ('(RETE) RETN_RTN1', RETN_RTN1,HEX(RETE_RTN1.PTR));*/ RETO1090
   ALLOCATE RETE_RTN1 SET (TP1);                                      RETO1100
```

```
    LP->RETE_RTN1.PTR=TP1;  LP=TP1;                                          RETO1110
    I=TP1->RETE_RTN1.CBTP; J=TP1->RETE_RTN1.LEN; PP=TP1->RETE_RTN1.PTR;      RETO1120
    TP1->RETE_RTN1=RETN_RTN1, BY NAME;/*COPY*/                               RETO1130
    TP1->RETE_RTN1.CBTP=I; TP1->RETE_RTN1.LEN=J; TP1->RETE_RTN1.PTR=PP;      RETO1140
    /*FIX UP DATA */                                                         RETO1150
    NODE=RETN_RTN1.NODE;                                                     RETO1160
    IF NODE = 1 THEN DO; /*ROOT HAS NO DATA */                               RETO1170
       TP1->RETE_RTN1.DLEN=0; GO TO END; END;                                RETO1180
    AINFO=MAINFO(NODE);                                                      RETO1190
    SELECT(AINFO.TYPE);                                                      RETO1200
      WHEN ('V') DO;                                                         RETO1210
                                                                             RETO1220
        BDATA=RETN_RTN1.DATA;                                                RETO1230
        DLEN=RETN_RTN1.DLEN;                                                 RETO1240
        CALL VTPE (AINFO,TO_CHAR,BDATA,CDATA,DLEN,RTN_CODE);                 RETO1250
        TP1->RETE_RTN1.DLEN=DLEN;                                            RETO1260
        TP1->RETE_RTN1.CDATA=CDATA;                                          RETO1270
        END;                                                                 RETO1280
                                                                             RETO1290
     WHEN ('E')                                                              RETO1300
        TP1->RETE_RTN1.DLEN=0; /*IGNORE DATA*/                               RETO1310
     END /*SELECT*/;    /*                                                   RETO1320
PUT SKIP LIST('(RETE)RETERTN',TP1->RETE_RTN1,HEX(TP1->RETE_RTN1.PTR));       RETO1330
*/   END; END;                                                               RETO1340
                                                                             RETO1350
CALL SVC3 (RP);     /* FREE UP ARG AND RTN LIST FROM RETN*/                  RETO1360
CALL SVC3 (TP);                                                              RETO1370
 CALL TRTN (PROC_ADDR, RTP); /*RETURNS WITH ARGUMENTS*/                      RETO1380
 DO P=RTP REPEAT RETE_RTN.PTR WHILE (P^=NULL);                               RETO1390
  END;                                                                       RETO1400
 END /*RETE*/;                                                               RETO1410
```

```
RETN:PROC;                                                              RET00010
/*************************************************************************
*                                                                       *  RET00020
*                                                                       *  RET00030
*                    MODULE    DESCRIPTION                            *  RET00040
*************************************************************************/ RET00050
/*************************************************************************
*****   PURPOSE:                                                          RET00060
*****    THIS IS A T-PROC AT THE N-ARY LEVEL WHICH PROCESSES              RET00070
*****     RETRIEVAL REQUESTS EXPRESSED IN A RETN TREE AS                  RET00080
*****     DEMONSTRATED BY THE RETN_ARG; IT RETURNS A LINK                 RET00090
*****     LIST OF DATA FOR EACH NODE IN THE RETN TREE.                    RET00100
*************************************************************************
*****   METHOD:                                                           RET00110
*****    1. THIS MODULE USES A VERY SIMPLE METHOD TO TRAVERSE             RET00120
*****       THE DATA BASE; IT FIRST VISITS THE ROOT NODE AND              RET00130
*****       OBTAINS THE FIRST OCCURENCE OF THE ROOT NODE; IT              RET00140
*****       THEN VERIFIES THE OCCURENCE AGAINST THE PREDICATE             RET00150
*****       (IF ANY) TO DECIDE WHETHER THIS OCCURENCE IS VALID;           RET00160
*****       IF YES, THEN CONTINUE; OTHERWISE DISCARD IT AND               RET00170
*****       GET THE NEXT ROOT OCCURENCE UNTIL ALL ROOT OCCURENCES         RET00180
*****        ARE EXAMINED.                                                RET00190
*****    2. AFTER A ROOT OCCURENCE IS ESTABLISHED, THIS ROUTINE           RET00200
*****       FOLLOWS THE RETN TREE TO ESTABLISH OCCURENCE OF ITS           RET00210
*****       ASSOCIATED ELEMENTS AND VERIFY THEIR CONTENT AGAINST          RET00220
*****       THEIR PREDICATES. IF ANY MISMATCH IS ENCOUNTERED, THE         RET00230
*****       CURRENT ROOT NODE OCCURENCE IS ABANDONED AND A NEW            RET00240
*****       ROOT NODE OCCURENCE IS OBTAINED AS IN 1.                      RET00250
*****    3. AFTER ONE OCCURENCE OF THE ENTIRE RETN TREE IS ESTABLISH      RET00260
*****       ED, A SUBROUTINE BUILD IS CALLED TO GENERATE THE DATA         RET00270
*****       FROM THE CURSOR IDS; AFTER WHICH THE NEXT ROOT NODE           RET00280
*****       IS OBTAINED AS IN 1.                                          RET00290
*************************************************************************
*****   INPUT PARAMETERS:                                                 RET00300
*****       AS INDICATED IN RETN_ARG;                                     RET00310
*************************************************************************
*****   OUTPUT PARAMETERS:                                                RET00320
*****       AS INDICATED IN RETN_RTN;                                     RET00330
*************************************************************************
*****   CALLING ARGUMENTS :                                             */ RET00340
%INCLUDE ARETN;                                                         RET00350
/****   CALLS PROCEDURES:                                                RET00360
*****    INTER-LEVEL T-PROC: NONE                                        RET00370
*****    INTRA-LEVEL T-PROC: NONE                                        RET00380
*****    INTRA-LEVEL S-PROC: BUILD,VERIF,RETB,RETP;                    */ RET00390
%INCLUDE EBUIL,EVERI,ERETB,ERETP;                                       RET00400
/****    CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES: */    RET00410
%INCLUDE SERVICE,FDEBUG;                                                RET00420
/****                                                                    RET00430
*************************************************************************/ RET00440
                                                                        RET00450
%INCLUDE EBCT,EPCT,DBEU;                                                RET00460
DCL CURSOR (25) PTR INIT((25) NULL); /* CURRENT PTR TO NODES*/          RET00470
DCL PSET (25) BIT(32) ALIGNED; /*TEMP FOR PSETID OF RETN TREE NDS*/     RET00480
DCL (P,MP,RTP,TP,RP,LP,ROOT_PTR) PTR,(NULL,SUBSTR,UNSPEC) BUILTIN;      RET00490
DCL (PTR1,PTR2) PTR;                                                    RET00500
                                                                        RET00510
                                                                        RET00520
                                                                        RET00530
                                                                        RET00540
                                                                        RET00550
```

```
DCL (RNODE,RTN_CODE) FIXED BIN;                                          RETO0560
DCL BDATA BIT(320) VARYING ALIGNED;                                      RETO0570
DCL (OP BIT(8), YES BIT(1)) ALIGNED;                                     RETO0580
DCL (PARENT,BSETID) BIT(32) ALIGNED;                                     RETO0590
DCL (K,NODE,I) FIXED BIN;                                                RETO0600
/*DCL OF BCAT TEMPLATE*/                                                 RETO0610
DCL 1 BINFO (25),                                                        RETO0620
      2 PSETID(2) BIT (32),                                              RETO0630
      2 POS FIXED BIN, /*POS OF POINTER ARRAY OF PSETID1 USED */         RETO0640
      2 FUNC CHAR (1); /*'S' OR 'M'*/;                                   RETO0650
/*DCLS OF PSET CATALOGUE AT THE N-ARY LEVEL******/                       RETO0660
DCL 1 PINFO (25),                                                        RETO0670
       3 NUMPTR FIXED BIN,                                               RETO0680
       3 PLEN /*IN NUMBER OF BYTES*/ FIXED BIN,                          RETO0690
       3  PTYPE CHAR (1), /*'N' OR'X' OR 'B' OR 'C'*/                    RETO0700
       3 LTYPE BIT (8), /*'00000001' IS LINK LIST*/                      RETO0710
       3 L_ID BIT (32), /*ID OF THE FIRST BEU IN THIS SET**/             RETO0720
       3 L_POS FIXED BIN , /*POS IN ID_ARRY USED FOR LINEAR CHAINING     RETO0730
                  OF BEU'S WITHIN THIS SET*/                             RETO0740
       3 L_POS2 FIXED BIN,                                               RETO0750
       3 MAP BIT (32) ; /*UP TO 16 PTRS ARE ALLOWED FOR A BEU*/          RETO0760
                                                                         RETO0770
/*BEGIN PROCESSING*/                                                     RETO0780
                                                                         RETO0790
CALL TBEG(PROC_ADDR,MP);                                                 RETO0800
ALLOCATE RETN_RTN SET (RTP);                                             RETO0810
RTP->RETN_RTN.RTN_CODE=0; /*INIT RTN CODE TO O.K.*/                      RETO0820
LP=RTP; /*LP INIT TO FIRST TOKEN TO RETURN*/  K=0;                       RETO0830
P=MP;                                                                    RETO0840
RNODE=RETN_ARG.NODE; /*RNODE IS ROOT NODE# */                            RETO0850
    /*INIT PSET STRUCTURE*/                                              RETO0860
                                                                         RETO0870
/********************* RETN REQUEST TRACE ****************************/RETO0880
DCL FFFN BIT (1) EXT;                                                 /**/RETO0890
IF FFFN THEN DO;                                                      /**/RETO0900
PUT SKIP EDIT('NODE *PSETID* PAR **GET*** N *BSETID* ***OP*** DL DATA')RETO0910
   (A);                                                               /**/RETO0920
                                                                      /**/RETO0930
DO P=MP REPEAT P->RETN_ARG.PTR WHILE (P^=NULL);                       /**/RETO0940
                                                                      /**/RETO0950
      PUT SKIP EDIT(RETN_ARG.NODE,HEX4(ADDR(RETN_ARG.PSETID)->BP),    /**/RETO0960
      RETN_ARG.PARENT,RETN_ARG.GET,RETN_ARG.N,' ')                    /**/RETO0970
      (F(3),X(2),A(8),F(3),X(2),B(8),F(2),A);                         /**/RETO0980
      IF RETN_ARG.PARENT^=0 THEN PUT EDIT(                            /**/RETO0990
      HEX4(ADDR(RETN_ARG.BSETID)->BP))(A(8));                         /**/RETO1000
      ELSE PUT EDIT('        ')(A);                                   /**/RETO1010
      IF RETN_ARG.N^=0                                                /**/RETO1020
      THEN DO;                                                        /**/RETO1030
        PUT EDIT(RETN_ARG.OP(1),RETN_ARG.DLEN(1),' ')                 /**/RETO1040
              (X(1),B(8),F(3),A);                                     /**/RETO1050
        DO II=1 TO RETN_ARG.DLEN(1)/4;DCL II FIXED BIN;               /**/RETO1060
           PUT EDIT(HEX4(ADDR(RETN_ARG.DATA(1))->PAA(II)))(A(8));     /**/RETO1070
           %INCLUDE HEX; DCL BP PTR BASED;                            /**/RETO1080
           END;                          DCL PAA(10) PTR BASED;       /**/RETO1090
        END;                                                          /**/RETO1100
```

```
      END;                                                              /**/RETO1110
    END;                                                                /**/RETO1120
   /*******************************************************************/ RETO1130
                                                                         RETO1140
 DO P=MP REPEAT P->RETN_ARG.PTR WHILE (P^=NULL);                         RETO1150
    IF RETN_ARG.NODE>25                                                  RETO1160
    THEN DO;                                                             RETO1170
       RTP->RETN_RTN.RTN_CODE=1;                                         RETO1180
       IF FDEBUG.LEV2  THEN PUT SKIP LIST ('(RETN) ERR RTN ',            RETO1190
       'PSET DATA STRUCTURE OVER FLOW (LIMIT 25)');                      RETO1200
       GO TO RTN;                                                        RETO1210
       END;                                                              RETO1220
    CALL GPCT (RETN_ARG.PSETID, PINFO(RETN_ARG.NODE), RTN_CODE);         RETO1230
    IF RETN_ARG.NODE ^= RNODE                                            RETO1240
    THEN CALL GBCT (RETN_ARG.BSETID, BINFO (RETN_ARG.NODE), RTN_CODE);   RETO1250
    PSET(RETN_ARG.NODE)=RETN_ARG.PSETID;                                 RETO1260
    END;                                                                 RETO1270
 CALL RETP(PINFO(RNODE),NULL,ROOT_PTR,RTN_CODE); /*1ST BEU*/             RETO1280
 /*BEGIN TREE PROCESSING*/                                              RETO1290
 OUTER_LOOP:                                                             RETO1300
 DO WHILE ('1'B);                                                        RETO1310
    /*ROOT NODE PROCESSING*/                                            RETO1320
     P=MP;                                                               RETO1330
     IF ROOT_PTR = NULL                                                  RETO1340
     THEN LEAVE OUTER_LOOP;                                              RETO1350
     ELSE CURSOR(RNODE) = ROOT_PTR;                                      RETO1360
                                                                         RETO1370
     IF RETN_ARG.N ^= 0                                                  RETO1380
     THEN DO; /*VERIFY NODE DATA AGAINST PREDICATE*/                     RETO1390
        OP=RETN_ARG.PRED.OP(1);                                          RETO1400
        BDATA=SUBSTR(RETN_ARG.PRED.DATA(1),1,                            RETO1410
               RETN_ARG.PRED.DLEN(1)*8);                                 RETO1420
        CALL VERIF (PINFO(RNODE),ROOT_PTR,OP,BDATA,RTN_CODE,YES);        RETO1430
        IF RTN_CODE^=0                                                   RETO1440
        THEN DO;                                                         RETO1450
           RTP->RETN_RTN.RTN_CODE=3;  /******* RTN CODE 3 **********/   RETO1460
           IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(RETN) ERR RTN ',         RETO1470
           'AFTER CALLING VERIF   WHICH GIVES RTN CODE',RTN_CODE);       RETO1480
           GO TO RTN;                                                    RETO1490
           END;                                                          RETO1500
                                                                         RETO1510
        IF ^YES THEN GO TO NEXT_ROOT; /*DISCARD THIS ROOT NODE*/         RETO1520
        END;                                                             RETO1530
                                                                         RETO1540
  /*NEXT RET TREE LEAF*/                                                RETO1550
    DO P=RETN_ARG.PTR REPEAT P->RETN_ARG.PTR WHILE (P^=NULL);            RETO1560
        NODE=RETN_ARG.NODE;                                              RETO1570
        PARENT =PSET(RETN_ARG.PARENT);                                   RETO1580
        BSETID=RETN_ARG.BSETID;                                          RETO1590
        PTR1=CURSOR(RETN_ARG.PARENT);                                    RETO1600
                                                                         RETO1610
        IF PTR1 = NULL                                                   RETO1620
        THEN DO;                                                         RETO1630
           IF RETN_ARG.N ^= 0 THEN GO TO NEXT_ROOT;                      RETO1640
           CURSOR(NODE) = NULL;                                          RETO1650
```

```
         END;                                                          RETO1660
                                                                       RETO1670
      ELSE DO;                                                         RETO1680
         CALL RETB(PARENT,BINFO(NODE),PTR1,PTR2,RTN_CODE);             RETO1690
         IF RTN_CODE^=0                                                RETO1700
         THEN DO;                                                      RETO1710
            RTP->RETN_RTN.RTN_CODE=4;  /******* RTN CODE 4 **********/ RETO1720
            IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(RETN) ERR RTN ',      RETO1730
            'AFTER CALLING RETB   WHICH GIVES RTN CODE',RTN_CODE);     RETO1740
            GO TO RTN;                                                 RETO1750
            END;                                                       RETO1760
         CURSOR(NODE)=PTR2;                                            RETO1770
         IF PTR2=NULL & RETN_ARG.N ^= 0 THEN GO TO NEXT_ROOT;          RETO1780
            /* IF PREDICATE ISSUED THEN DISCARD NULL ATTRIBUTE */      RETO1790
                                                                       RETO1800
         IF PTR2^= NULL & RETN_ARG.N^=0                                RETO1810
         THEN DO;                                                      RETO1820
            OP=RETN_ARG.PRED.OP(1);                                    RETO1830
            BDATA = SUBSTR(RETN_ARG.PRED.DATA(1),1,                    RETO1840
                    RETN_ARG.PRED.DLEN(1)*8);                          RETO1850
            CALL VERIF(PINFO(RNODE),PTR2,OP,BDATA,RTN_CODE,YES);       RETO1860
            IF RTN_CODE^=0                                             RETO1870
            THEN DO;                                                   RETO1880
               RTP->RETN_RTN.RTN_CODE=5;  /*** RTN CODE 5 ****/      RETO1890
               IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(RETN) ERR RTN ',   RETO1900
               'AFTER CALLING VERIF WHICH GIVES RTN CODE',RTN_CODE);   RETO1910
               GO TO RTN;                                              RETO1920
               END;                                                    RETO1930
            IF ^YES THEN GO TO NEXT_ROOT; /*DISCARD CURRENT ROOT*/     RETO1940
            END;                                                       RETO1950
         END;                                                          RETO1960
      END;                                                             RETO1970
                                                                       RETO1980
 /*NOW A COMPLETE SET OF CURSOR HAS BEEN SET UP*/                      RETO1990
   K=K+1;  /*INDEX TO OCCUR OF ROOT NODE*/                            RETO2000
   CALL BUILD (CURSOR,PSET,MP,K,LP,RTN_CODE);                          RETO2010
   IF RTN_CODE^=0                                                      RETO2020
   THEN DO;                                                            RETO2030
      RTP->RETN_RTN.RTN_CODE=6;  /******* RTN CODE 6 **********/      RETO2040
      IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(RETN) ERR RTN ',            RETO2050
      'AFTER CALLING BUILD WHICH GIVES RTN CODE',RTN_CODE);            RETO2060
      GO TO RTN;                                                       RETO2070
      END;                                                             RETO2080
NEXT_ROOT:                                                             RETO2090
   CALL RETP(PINFO(RNODE),CURSOR(RNODE),ROOT_PTR,RTN_CODE);            RETO2100
   DO I = 1 TO 25;                                                     RETO2110
      IF CURSOR(I) ^= NULL                                             RETO2120
      THEN DO;                                                         RETO2130
         FREE CURSOR(I)->BEU;                                          RETO2140
         CURSOR(I) = NULL;                                             RETO2150
         END;                                                          RETO2160
      END;                                                             RETO2170
   IF RTN_CODE ^= 0                                                    RETO2180
   THEN DO;                                                            RETO2190
      RTP->RETN_RTN.RTN_CODE=2; /******RTN CODE 2 *********/           RETO2200
```

```
      IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(RETN) ERR RTN ',          RETO2210
      'AFTER CALLING RETP WHICH GIVES RTN CODE',RTN_CODE);           RETO2220
      GO TO RTN;                                                     RETO2230
      END;                                                           RETO2240
   IF MP->RETN_ARG.GET=RETNGET.ANY & K > 0  /*DONE*/                 RETO2250
   THEN DO;                                                          RETO2260
      IF ROOT_PTR ^= NULL THEN FREE ROOT_PTR->BEU;                   RETO2270
      LEAVE OUTER_LOOP;                                              RETO2280
      END;                                                           RETO2290
   END /*DO WHILE*/;                                                 RETO2300
RTP->RETN_RTN.N = K;                                                 RETO2310
RTN: CALL TRTN(PROC_ADDR,RTP);                                       RETO2320
                                                                     RETO2330
END /*RETN*/;                                                        RETO2340
```

```
RETP: PROC(PINFO,PTR1,PTR2,RTN_CODE);                                  RET00010
                                                                       RET00020
/* A S-PROC AT THE N-ARY LEVEL WHICH RETRIEVES THE ID OF THE P ELEM    RET00030
* THAT FOLLOWS ID1 IN PSETID. IF ID1 IS NULL THEN IT RETRIEVES THE     RET00040
* THE FIRST OCCR IN PSETID*/                                           RET00050
                                                                       RET00060
%INCLUDE DBEU.DPCAT;                                                   RET00070
%INCLUDE EPCT.ESUB1;                                                   RET00080
                                                                       RET00090
DCL RTN_CODE FIXED BIN;                                                RET00100
DCL (P,PTR1,PTR2) PTR,(NULL,UNSPEC) BUILTIN;                           RET00110
DCL ID2 BIT (32);                                                      RET00120
                                                                       RET00130
IF PTR1= NULL  /*GET FIRST*/                                           RET00140
THEN ID2=PINFO.L_ID;                                                   RET00150
                                                                       RET00160
ELSE ID2=PTR1->BEU.ID_ARRAY(PINFO.L_POS);                              RET00170
IF ID2 = UNSPEC (NULL) THEN DO;                                        RET00180
   RTN_CODE = 0;                                                       RET00190
   PTR2 = NULL;                                                        RET00200
   RETURN;                                                             RET00210
   END;                                                                RET00220
CALL RET1 (ID2, PTR2, RTN_CODE);                                       RET00230
RETURN;                                                                RET00240
END;                                                                   RET00250
```

```
RET1:PROC(ID,P,RTN_CODE);                                              DUM00010
 /*S-PROC AT THE LOWEST LAYER OF THE N-ARY LEVEL TO SET UP CALL FOR     DUM00020
  ** RETRIEVAL OF ELEMENT DESIGNATED BY ID*/                           DUM00030
                                                                       DUM00040
%INCLUDE SERVICE,ARET,DBEU;                                            DUM00050
DCL ID BIT (32), (I,J,K) FIXED BIN, (P,RP) PTR;                        DUM00060
DCL (SUBSTR,UNSPEC,NULL,ADDR) BUILTIN, RTN_CODE FIXED BIN;             DUM00070
                                                                       DUM00080
ALLOCATE RET_ARG; RP=P;                                                DUM00090
RET_ARG.ID=ID;                                                         DUM00100
CALL TCALL ('RET',1,RP,AP);                                            DUM00110
                                                                       DUM00120
/*************************************/                                DUM00130
IF FFF                                /* MEMORY REQUEST TRACE ON */    DUM00140
THEN CALL DUMPIT ('RETRIEVED',ID,NULL);  /**/                          DUM00150
DCL FFF BIT (1) EXT;                  /**/                             DUM00160
/*************************************/                                DUM00170
                                                                       DUM00180
CALL SVC3 (RP);                                                        DUM00190
                                                                       DUM00200
IF AP->RET_RTN.RTN_CODE^=0 THEN DO; /*ID INVALID*/                     DUM00210
   CALL SVC3(AP);                                                      DUM00220
   P=NULL;                                                             DUM00230
   RETURN;                                                             DUM00240
   END;                                                                DUM00250
                                                                       DUM00260
P=ADDR(AP->RET_RTN.DATA);                                              DUM00270
I=NUMPTR;J=DLEN;                                                       DUM00280
ALLOCATE BEU SET (RP);                                                 DUM00290
RP->BEU.DATA=BEU.DATA;                                                 DUM00300
RP->BEU.ID_ARRAY=BEU.ID_ARRAY;                                         DUM00310
CALL SVC3 (AP);                                                        DUM00320
P=RP;                                                                  DUM00330
RTN_CODE=0;                                                            DUM00340
RETURN;                                                                DUM00350
END/*RET1*/;                                                           DUM00360
                                                                       DUM00370
```

```
SETFH: PROC;/* SUBROUTINE FOR SETTING FH TRACE OPTIONS */                SET00010
/*** FTCALL -- TCALL ONLY;                                              SET00020
     FDEBUG -- ALL MODULES AT ALL LEVELS;                               SET00030
     FFF -- RET1, REP1, CRT1, DEL1 ONLY;                                SET00040
     FFFN -- RETN, UPDN ONLY;    ********/                              SET00050
                                                                        SET00060
DCL (P,RP) PTR;   %INCLUDE FDEBUG;                                      SET00070
DCL FTCALL(4) BIT (1) EXT;                                              SET00080
ON ERROR SNAP SYSTEM;                                                   SET00090
PUT SKIP LIST ('ENTER FH TRACE OPTIONS');                               SET00100
PUT LIST ('ENTER TCALL TRACE OPTION (4 BITS)');                         SET00110
DO I = 1 TO 4;                                                          SET00120
   GET EDIT (FTCALL(I))(B(1));                                          SET00130
   END;                                                                 SET00140
                                                                        SET00150
PUT LIST ('ENTER SYSTEM ERROR MESSAGE TRACE OPTION (4 BITS)');          SET00160
GET EDIT (FDEBUG.LEV0) (B(1));                                          SET00170
GET EDIT (FDEBUG.LEV1) (B(1));                                          SET00180
GET EDIT (FDEBUG.LEV2) (B(1));                                          SET00190
GET EDIT (FDEBUG.LEV3) (B(1));                                          SET00200
                                                                        SET00210
DCL FFF BIT(1) EXT;                                                     SET00220
PUT LIST('ENTER MEMORY REQUEST TRACE OPTION (1 BIT):');                 SET00230
GET EDIT(FFF)(B(1));                                                    SET00240
                                                                        SET00250
DCL FFFN BIT (1) EXT;                                                   SET00260
PUT LIST ('ENTER UPDN/RETN REQUEST TRACE OPTION (1 BIT):');             SET00270
GET EDIT (FFFN) (B(1));                                                 SET00280
                                                                        SET00290
DCL FTIME BIT(1) EXT;                                                   SET00300
PUT LIST ('ENTER TIMING REPORT TRACE OPTION (1 BIT):');                 SET00310
GET EDIT (FTIME) (B(1));                                                SET00320
PUT SKIP LIST ('FH TRACE OPTIONS ENDS');                                SET00330
END;                                                                    SET00340
```

```
SHWE: PROC;                                                          SHW00010
                                                                     SHW00020
/* T-PROC AT THE ENTITY LEVEL, INVOKED IN RESPONSE TO A CALL FROM    SHW00030
MODULE DDQ AT THE USER INTERFACE LEVEL TO RETRIEVE DATA DEFINITION   SHW00040
INFO */                                                              SHW00050
                                                                     SHW00060
%INCLUDE SERVICE, ASHWE, DECAT, EGACT,EGECT,ARETE,EALLA;             SHW00070
DCL (P,LP,TP,PP,RP) PTR, NULL BUILTIN, (RTN_CODE,I,N) FIXED BIN;     SHW00080
DCL MOD BUILTIN;                                                     SHW00090
                                                                     SHW00100
/* BEGIN */                                                          SHW00110
CALL TBEG (PROC_ADDR, SHWE_ARG_P);                                   SHW00120
ALLOCATE SHWE_RTN; LP=SHWE_RTN_P;                                    SHW00130
                                                                     SHW00140
IF NAME1='*' THEN DO; /*ALL ENTITY SETS QUERIED */                   SHW00150
                                                                     SHW00160
   /* CALL UP RETE TO RETRIEVE DATA */                               SHW00170
    ALLOCATE RETE_ARG; RP,LP=P;                                      SHW00180
    RETE_ARG.NAME='E*ESET';                                          SHW00190
    RETE_ARG.GET=RETEGET.ALL;                                        SHW00200
                                                                     SHW00210
    ALLOCATE RETE_ARG; LP->RETE_ARG.PTR=P; LP=P;                     SHW00220
    RETE_ARG.NODE=2;                                                 SHW00230
    RETE_ARG.NAME='A*ENAME';                                         SHW00240
    RETE_ARG.GET=RETEGET.ALL;                                        SHW00250
    RETE_ARG.PARENT=1;                                               SHW00260
                                                                     SHW00270
    CALL TCALL ('RETE',1,RP,TP);                                     SHW00280
    CALL SVC3 (RP);                                                  SHW00290
                                                                     SHW00300
    /* EXAMINE RETURNS */                                            SHW00310
    P=TP;                                                            SHW00320
    IF RETE_RTN.RTN_CODE^=0 THEN DO;                                 SHW00330
       SHWE_RTN.RTN_CODE=30+RETE_RTN.RTN_CODE; /** 30+ ************/  SHW00340
       CALL SVC3 (P);   GO TO RTN; END;                              SHW00350
                                                                     SHW00360
    IF RETE_RTN.N=0 THEN DO;                                         SHW00370
       SHWE_RTN.RTN_CODE=-2; /*** -2 IS NO ENTITY SET DEFINED ******/ SHW00380
       CALL SVC3 (P);  GO TO RTN; END;                               SHW00390
                                                                     SHW00400
     /*COPY ENAMES RETRIEVED INTO SHWE_RTN2 */                       SHW00410
     N=RETE_RTN.N;                                                   SHW00420
     ALLOCATE SHWE_RTN2 SET (PP);  SHWE_RTN.PTR=PP;                  SHW00430
     PP->SHWE_RTN2.NUM_ENAME=N;                                      SHW00440
                                                                     SHW00450
     DO I = 1 TO N;                                                  SHW00460
        DO P= RETE_RTN.PTR REPEAT RETE_RTN1.PTR WHILE (P^=NULL);     SHW00470
           IF RETE_RTN1.NODE=2 /* ENAME NODE */                     SHW00480
               THEN DO; PP->SHWE_RTN2.ENAME(I)=RETE_RTN1.CDATA;      SHW00490
                        GO TO NEXTI;                                 SHW00500
                        END; /*GET OUT P LOOP */                     SHW00510
         END /* P LOOP */;                                          SHW00520
     NEXTI: END /* DO I */;                                         SHW00530
                                                                     SHW00540
     CALL SVC3 (TP); GO TO RTN; /* DONE */                          SHW00550
```

```
      END /*IF ENAME = '*' */;                                          SHW00560
                                                                        SHW00570
 /*CHECK ENTITY SET , WHEN ENTITY SET NAME IS GIVEN */                  SHW00580
 CALL GECT (NAME1,EINFO,RTN_CODE);                                      SHW00590
 IF RTN_CODE ^=0 THEN DO;                                               SHW00600
    SHWE_RTN.RTN_CODE=-1; /***** -1 IS PROBLEM WITH ENTITY SET NME */   SHW00610
    GO TO RTN;  END;                                                    SHW00620
                                                                        SHW00630
 /*CHECK ATTRIBUTE */                                                   SHW00640
                                                                        SHW00650
    IF NUM_NAME2=999 /* ALL ATTRIBUTES */                              SHW00660
    THEN DO;      /* CALL UP ALLA TO RETRIEVE ALL ATTR NAMES */         SHW00670
      NUM_NAME2 = 0;                                                    SHW00680
      CALL ALLA (NAME1,NAME2,NUM_NAME2,RTN_CODE);                       SHW00690
     IF RTN_CODE^=0 THEN DO;                                            SHW00700
        SHWE_RTN.RTN_CODE=500 + RTN_CODE; /** 500+ **********/          SHW00710
        GO TO RTN; END;                                                 SHW00720
                                                                        SHW00730
     IF NUM_NAME2 =0 THEN DO; /* NO ATTRIBUTES DEFINED */               SHW00740
        SHWE_RTN.RTN_CODE=-3; /*** -3 IS NO ATTRIBUTES FOUND ********/ SHW00750
        GO TO RTN; END;                                                 SHW00760
    END /* IF 999 */;                                                  SHW00770
                                                                        SHW00780
  /* BY NOW NAME2 CONTAINS ATTRIBUTES TO BE SHOWN */                    SHW00790
 LP=SHWE_RTN_P;                                                         SHW00800
 DO I = 1 TO NUM_NAME2;                                                 SHW00810
    CALL GACT (NAME2(I),NAME1,AINFO,RTN_CODE);                          SHW00820
    IF RTN_CODE ^= 0 THEN DO;                                           SHW00830
       SHWE_RTN.RTN_CODE=I; /*0+ IS PROBLEM WITH ATTRIBUTE NAME ****/   SHW00840
       GO TO RTN;  END;                                                 SHW00850
                                                                        SHW00860
    ALLOCATE SHWE_RTN1 SET (TP);                                        SHW00870
    LP->SHWE_RTN1.PTR=TP; LP=TP;                                        SHW00880
    TP->SHWE_RTN1.DATA=AINFO, BY NAME;/* COPY AINFO INTO RETURN TOKEN*/ SHW00890
    TP->SHWE_RTN1.ANAME=NAME2(I);                                       SHW00900
    END /* DO I */;                                                    SHW00910
                                                                        SHW00920
 RTN: CALL TRTN (PROC_ADDR, SHWE_RTN_P);                                SHW00930
 RETURN;                                                                SHW00940
 END /*SHWE */;                                                        SHW00950
```

```
SRCH: PROC(PSETID,DATA,ID,RTN_CODE);                                    SRC00010
                                                                        SRC00020
/* A S-PROC AT THE N-ARY LEVEL WHICH LOCATES DATA IN A PSET AND         SRC00030
* RETURNS ITS ID. IF NOT FOUND, RTN_CODE IS SET TO 1*/                  SRC00040
                                                                        SRC00050
%INCLUDE DBEU,DPCAT;                                                    SRC00060
%INCLUDE EPCT,ESUB1;                                                    SRC00070
DCL (PSETID,ID) BIT (32), DATA BIT (*), RTN_CODE FIXED BIN, P PTR;      SRC00080
DCL(NULL,UNSPEC) BUILTIN;                                               SRC00090
DCL POS FIXED BIN;                                                      SRC00100
CALL GPCT (PSETID, PINFO, RTN_CODE);                                    SRC00110
  ID=PINFO.L_ID;                                                        SRC00120
  POS=PINFO.L_POS;                                                      SRC00130
                                                                        SRC00140
  DO WHILE (ID^=UNSPEC(NULL));                                          SRC00150
     CALL RET1 (ID,P,RTN_CODE);                                         SRC00160
     IF P=NULL THEN DO;                                                 SRC00170
        RTN_CODE=2; /*BAD PSET CHAIN***********/                        SRC00180
        RETURN;                                                         SRC00190
        END;                                                            SRC00200
    IF BEU.DATA=DATA THEN LEAVE;                                        SRC00210
    ID=BEU.ID_ARRAY(POS); /*NEXT BEU IN THIS PSET*/                     SRC00220
 END;/*DO WHILE*/                                                       SRC00230
                                                                        SRC00240
 IF ID=UNSPEC(NULL) THEN RTN_CODE=1; /*NOT FOUND*/                      SRC00250
  ELSE RTN_CODE=0;                                                      SRC00260
END /*SRCH*/;                                                          SRC00270
```

```
SVCS1: PROC (PROC_NAME,PTR,RTN_CODE);                                   SVC00010
/*********************************************************************  SVC00020
*                                                                   *  SVC00030
*                    MODULE    DESCRIPTION                          *  SVC00040
*********************************************************************/  SVC00050
/*********************************************************************  SVC00060
*****    PURPOSE:    IT SERVES AS THE                                 SVC00070
*****    SVC SEND ROUTINE IN THE LOS, TO INSULATE APPLN PROGRAMS      SVC00080
*****     FROM THE HARDWARE COMM PROTOCOL SEND; THIS ROUTINE          SVC00090
*****     CREATES A NEW PROCESS AT THE PROPER LEVEL.                  SVC00100
*****                                                                 SVC00110
*********************************************************************  SVC00120
*****    METHOD:                                                      SVC00130
*****     1. DETERMINE THE LEVEL TO WHICH THE PROC BELONGS TO;        SVC00140
*****     2. DETRMINE THE TOTAL LENGTH OF MSG BY FOLLOWING CHAIN;     SVC00150
*****     3. CALL SEND.                                               SVC00160
*****                                                                 SVC00170
*********************************************************************  SVC00180
*****    INPUT PARAMETERS:                                            SVC00190
*****     PROC_NAME: CHAR(7), NAME OF PROC TO BE CREATED;             SVC00200
*****     PTR: PTR, POINT TO MSG;                                     SVC00210
*****                                                                 SVC00220
*********************************************************************  SVC00230
*****    OUTPUT PARAMETERS:  RTN_CODE: FIXED BIN, 1=NO SUCH PROC;     SVC00240
*****                                                                 SVC00250
*********************************************************************  SVC00260
*****    CALLS PROCEDURES:  SEND.                                     SVC00270
*****                                                                 SVC00280
*********************************************************************/  SVC00290
%INCLUDE USERS;    /*DCLS OF SEND PROTOCOLS OF THE CONTROL STRUCTURE*/  SVC00300
%INCLUDE VPX; /*DCLS OF VP DATA STRUCTURES IN THE LOS*/                 SVC00310
%INCLUDE PFSVC;                                                         SVC00320
%INCLUDE FDEBUG;                                                        SVC00330
DCL PROC_NAME CHAR(7), (PTR,P) PTR,                                     SVC00340
    (RTN_CODE,I,J) FIXED BIN(15);                                       SVC00350
DCL 1 ARG_LIST BASED (P), /*TEMPLATE OF MSG CHAIN*/                     SVC00360
      2 LEN FIXED BIN (15),                                             SVC00370
      2 CBTP FIXED BIN (15),                                            SVC00380
      2 PTR PTR;                                                        SVC00390
DCL (LEVEL,VPID,BOXID,LEN) FIXED BIN, TYPE CHAR(1);                     SVC00400
%INCLUDE SVCTB; /*MACRO DEF OF PROC TABLE*/                             SVC00410
/*BEGIN PROCESSING*/                                                    SVC00420
/*DETERMINE THE LEVEL TO WHICH THE PROCESS TO BE CREATED BELONGS*/      SVC00430
J=0;                                                                    SVC00440
DO I=1 TO PROC_TBL_LEN;                                                 SVC00450
 IF PROC_TBL.N(I)=PROC_NAME THEN J=I;                                   SVC00460
 IF J^=0 THEN LEAVE;                                                    SVC00470
END;                                                                    SVC00480
IF J=0 THEN DO; /*NO SUCH PROC_NAME*/                                   SVC00490
 RTN_CODE=1;                                                            SVC00500
 RETURN;                                                                SVC00510
 END;                                                                   SVC00520
/* DETERMINE THE TOTAL LENGTH OF MSG BY FOLLOWING MSG CHAIN*/          SVC00530
LEN=0;                                                                  SVC00540
DO P=PTR REPEAT P->ARG_LIST.PTR WHILE (P^=NULL());                      SVC00550
```

```
   LEN=LEN + P->ARG_LIST.LEN;                                          SVC00560
   END;                                                                SVC00570
 /*SET UP SEND CALL*/                                                  SVC00580
 LEVEL=PROC_TBL.LEVEL(J);                                              SVC00590
 VPID=0; BOXID=1; TYPE='S';                                            SVC00600
 ALLOCATE PF_SVC;                                                      SVC00610
 PF_SVC.SVC = PROC_NAME;                                               SVC00620
 PF_SVC.PTR = PTR;                                                     SVC00630
 CALL SEND (LEVEL,VPID,BOXID,TYPE,LEN,PT_SVC);                         SVC00640
 RTN_CODE=0;  /*O.K.*/                                                 SVC00650
 END /*SVCS1*/;                                                        SVC00660
```

```
SVCS2: PROC (PROC_ADDR,PTR,RTN_CODE);                                   SVC00010
/*********************************************************************** SVC00020
*                                                                       * SVC00030
*                    MODULE    DESCRIPTION                              * SVC00040
***********************************************************************/ SVC00050
/*********************************************************************** SVC00060
*****   PURPOSE:    IT SERVES AS THE                                    SVC00070
*****   SVC SEND ROUTINE IN THE LOS, TO INSULATE APPLN PROGRAMS         SVC00080
*****    FROM THE          HARDWARE COMM PROTOCOL SEND; THIS ROUTINE    SVC00090
*****    SENDS A MSG TO AN OLD PROCESS WHOSE ADDR IS PASSED AS          SVC00100
*****    TO THIS ROUTINE AS PROC_ADDR.                                  SVC00110
*********************************************************************** SVC00120
*****   METHOD:                                                         SVC00130
*****    1. DETRMINE THE TOTAL LENGTH OF MSG BY FOLLOWING CHAIN;        SVC00140
*****    2. CALL SEND.                                                  SVC00150
*****                                                                   SVC00160
*********************************************************************** SVC00170
*****   INPUT PARAMETERS:                                               SVC00180
*****    PROC_ADDR: (3) FIXED BIN, THE ADDR OF THE OLD PROCESS;         SVC00190
*****    PTR: PTR, POINT TO MSG;                                        SVC00200
*****                                                                   SVC00210
*********************************************************************** SVC00220
*****   OUTPUT PARAMETERS:  RTN_CODE: FIXED BIN, 0= O.K.                SVC00230
*****                                                                   SVC00240
*********************************************************************** SVC00250
*****   CALLS PROCEDURES:  SEND.                                        SVC00260
*****                                                                   SVC00270
***********************************************************************/ SVC00280
                                                                        SVC00290
%INCLUDE FDEBUG;                                                        SVC00300
%INCLUDE USERS;    /*DCLS OF SEND PROTOCOLS OF THE CONTROL STRUCTURE*/  SVC00310
%INCLUDE VPX; /*DCLS OF VP DATA STRUCTURES IN THE LOS*/                 SVC00320
DCL 1 PROC_ADDR,                                                        SVC00330
    (2 LEVEL,                                                           SVC00340
     2 VPID,                                                            SVC00350
     2 BOXID) FIXED BIN (15);                                           SVC00360
DCL                    (PTR,P) PTR,                                     SVC00370
     RTN_CODE      FIXED BIN(15);                                       SVC00380
DCL 1 ARG_LIST BASED (P), /*TEMPLATE OF MSG CHAIN*/                     SVC00390
      2 LEN FIXED BIN (15),                                             SVC00400
      2 DUM_CBTP FIXED BIN (15),                                        SVC00410
      2 PTR PTR;                                                        SVC00420
DCL (LEVEL,VPID,BOXID,LEN) FIXED BIN, TYPE CHAR(1);                     SVC00430
/* DETERMINE THE TOTAL LENGTH OF MSG BY FOLLOWING MSG CHAIN*/           SVC00440
LEN=0;                                                                  SVC00450
DO P=PTR REPEAT P->ARG_LIST.PTR WHILE (P^=NULL());                      SVC00460
  IF FDEBUG.LEVO THEN                                                   SVC00470
  LEN=LEN + P->ARG_LIST.LEN;                                            SVC00480
  END;                                                                  SVC00490
/*SET UP SEND CALL*/                                                    SVC00500
LEVEL=PROC_ADDR.LEVEL;                                                  SVC00510
VPID=PROC_ADDR.VPID; BOXID=PROC_ADDR.BOXID; TYPE='S';                   SVC00520
CALL SEND (LEVEL,VPID,BOXID,TYPE,LEN,PTR    );                         SVC00530
RTN_CODE=0;  /*O.K.*/                                                  SVC00540
END /*SVCS2*/;                                                          SVC00550
```

```
SVC3:PROC(P);                                                          SVC00010
                                                                       SVC00020
/*** THIS IS A SERVICE ROUTINE TO FREE UP TOKEN LINK LIST *****/       SVC00030
/* TOKEN HAS BEEN REPRIEVED   */                                      SVC00040
  DCL 1 TOKEN BASED (P) ,                                              SVC00050
        2 LEN   FIXED BIN(15),                                         SVC00060
        2 CBTP  FIXED BIN(15),                                         SVC00070
        2 PTR   PTR,                                                   SVC00080
        2 DATA CHAR (I REFER (TOKEN.LEN)); /*  */                     SVC00090
                                                                       SVC00100
DCL (P,P1) PTR, (I,LEN,CBTP) FIXED BIN(15),                            SVC00110
    NULL BUILTIN;                                                      SVC00120
%INCLUDE AREP ,ACRT ,ADEFA,ADEFB,ADEFE,ADEFP,ADEL ,AMINI,ANINI,ARET ;  SVC00130
%INCLUDE ARETE,ARETN,AUPDE,AUPDN,AVINI,AVNME,ASHWE;                    SVC00140
%INCLUDE HEX;                                                          SVC00150
%INCLUDE FDEBUG;                                                       SVC00160
DO WHILE (P^=NULL);                                                    SVC00170
  P1=TOKEN.PTR; /*SAVE NEXT PTR*/                                      SVC00180
  CBTP=TOKEN.CBTP;                                                     SVC00190
  LEN=TOKEN.LEN;                                                       SVC00200
  /*                                                                   SVC00210
  IF FDEBUG.LEVO                                                       SVC00220
  THEN PUT SKIP EDIT('   SVC3: PTR ',HEX(P),' LEN',LEN,' TYP',CBTP)    SVC00230
                   (A,A(6),A,F(6),A,F(3));                             SVC00240
  */                                                                   SVC00250
  SELECT (CBTP);                                                       SVC00260
    WHEN (1) FREE P->REP_ARG;                                          SVC00270
    WHEN (2) FREE P->REP_RTN;                                          SVC00280
    WHEN (3) FREE P->CRT_ARG;                                          SVC00290
    WHEN (4) FREE P->CRT_RTN;                                          SVC00300
    WHEN (5) FREE P->DEFA_ARG;                                         SVC00310
    WHEN (6) FREE P->DEFA_RTN;                                         SVC00320
    WHEN (7) FREE P->DEFB_ARG;                                         SVC00330
    WHEN (8) FREE P->DEFB_RTN;                                         SVC00340
    WHEN (9) FREE P->DEFE_ARG;                                         SVC00350
    WHEN(10) FREE P->DEFE_RTN;                                         SVC00360
    WHEN(11) FREE P->DEFP_ARG;                                         SVC00370
    WHEN(12) FREE P->DEFP_RTN;                                         SVC00380
    WHEN(13) FREE P->DEL_ARG;                                          SVC00390
    WHEN(14) FREE P->DEL_RTN;                                          SVC00400
    WHEN(15) FREE P->MINIT_ARG;                                        SVC00410
    WHEN(16) FREE P->MINIT_RTN;                                        SVC00420
    WHEN(17) FREE P->NINIT_ARG;                                        SVC00430
    WHEN(18) FREE P->NINIT_RTN;                                        SVC00440
    WHEN(19) FREE P->RET_ARG;                                          SVC00450
    WHEN(20) FREE P->RET_RTN;                                          SVC00460
    WHEN(21) FREE P->RETE_ARG;                                         SVC00470
    WHEN(22) FREE P->RETE_RTN;                                         SVC00480
    WHEN(23) FREE P->RETE_ARG1;                                        SVC00490
    WHEN(24) FREE P->RETE_RTN1;                                        SVC00500
    WHEN(25) FREE P->RETN_ARG;                                         SVC00510
    WHEN(26) FREE P->RETN_RTN;                                         SVC00520
    WHEN(27) FREE P->RETN_RTN1;                                        SVC00530
    WHEN(28) FREE P->UPDE_ARG;                                         SVC00540
    WHEN(29) FREE P->UPDE_RTN;                                         SVC00550
```

```
     WHEN(30) FREE P->UPDN_ARG;                                    SVC00560
     WHEN(31) FREE P->UPDN_RTN;                                    SVC00570
     WHEN(32) FREE P->VINIT_ARG;                                   SVC00580
     WHEN(33) FREE P->VINIT_RTN;                                   SVC00590
     WHEN(34) FREE P->VNME_ARG;                                    SVC00600
     WHEN(35) FREE P->VNME_RTN;                                    SVC00610
     WHEN(36) FREE P->SHWE_ARG;                                    SVC00620
     WHEN(37) FREE P->SHWE_RTN;                                    SVC00630
     WHEN(38) FREE P->SHWE_RTN1;                                   SVC00640
     WHEN(39) FREE P->SHWE_RTN2;                                   SVC00650
                                                                   SVC00660
     OTHERWISE DO; PUT SKIP LIST ('***SVC3 WARNING',CBTP);         SVC00670
      SIGNAL ERROR; END;                                           SVC00680
     END;                                                          SVC00690
   P=P1;                                                           SVC00700
END;                                                               SVC00710
   /*                                                              SVC00720
IF FDEBUG.LEVO THEN PUT EDIT('   **** SVC3: END') (A);            SVC00730
  */                                                               SVC00740
RETURN;                                                            SVC00750
END /*SVC3*/;                                                     SVC00760
```

```
TBEG: PROC (PROC_ADDR,P);                                              TBE00010
                                                                       TBE00020
 /**** A SERVICE ROUTINE TO STRIF OFF MSG TOKEN FROM A CALLING ***      TBE00030
  *** ARGUMENT LIST; IT IS USED BY A T-PROC WHEN IT IS INVOKED ***/     TBE00040
                                                                       TBE00050
/**** PROC_ADDR CONTAINS THE PROC_ADDR OF THE CALLING PROCEDURE, ***    TBE00060
 ***  P WILL PONT TO THE FIRST TOKEN OF THE CALLING ARGUMENT LIST. */   TBE00070
                                                                       TBE00080
 DCL P PTR;                                                            TBE00090
 %INCLUDE USERS,VPX,FDEBUG, ESVCS;                                     TBE00100
PROC_ADDR=THISVP->VP.WAIT.MSG->MSG.RTN_ADDR;                           TBE00110
P= THISVP->VP.WAIT.MSG->MSG.PTR;                                       TBE00120
FREE THISVP->VP.WAIT.MSG->MSG; /*FREE MSG*/                            TBE00130
RETURN;                                                                TBE00140
END;                                                                   TBE00150
                                                                       TBE00160
```

```
TCALL:PROC (PROC_NAME,I,RP,TP) RECURSIVE ;                              TCA00010
 /*** A SERVICE ROUTINE FOR CROSS-LEVEL PROCEDURE CALLS*/              TCA00020
 /*** PROCE_NAME IS THE NAME OF THE PROCEDURE TO BE INVOKED; ***       TCA00030
 **** I IS THE BOX ID TO WAIT ON; RP POINTS TO THE BEGINNING ***       TCA00040
 **** TOKEN OF THE CALLING ARGUMENT LIST;                             TCA00050
 **** TP POINTS TO THE BEGINNING TOKEN OF THE RETURN ARGUMENT ***     TCA00060
 **** LIST WHEN THE CALLED PROCEDURE RETURNS *****************/       TCA00070
                                                                      TCA00080
DCL FTCALL(4) BIT(1) STATIC EXT;     /* FTCALL IS DEBUG TRACE BIG */  TCA00090
DCL FTIME BIT(1) EXT;              /* FTIME IS TIMING REPORT OPTION */ TCA00100
%INCLUDE FSTRUC;                                                      TCA00110
                                                                      TCA00120
DCL (RP,P,TP) PTR; DCL NULL BUILTIN, I FIXED BIN;                     TCA00130
%INCLUDE USERS,VPX,FDEBUG, ESVCS;                                     TCA00140
DCL RTN_CODE FIXED BIN;                                               TCA00150
                                                                      TCA00160
DO ENUM = 1 TO 18;                    /* FIND ENUM AND LEVEL       */ TCA00170
   IF ARCH.PROCNAME(ENUM) = PROC_NAME THEN LEAVE;                     TCA00180
   END;                                                               TCA00190
I=ARCH.LEVEL(ENUM);                                                   TCA00200
                                                                      TCA00210
/********************* TCALL TRACE OPTION *************************/  TCA00220
IF FTCALL(I)            /* IF TCALL TRACE BIT FOR THAT LEVEL IS ON*/  TCA00230
THEN PUT SKIP EDIT ('TCALL: ',PROC_NAME)(A,A);                        TCA00240
/**********************************************************************/ TCA00250
                                                                      TCA00260
/********************* TIMING REPORT OPTION ***********************/  TCA00270
DCL (BEGTIME,ENUM,LOTIME CTL) FIXED BIN(31);                          TCA00280
DCL RTIMER ENTRY RETURNS(FIXED BIN(31));                              TCA00290
DCL (ECOUNT(50),ELO(50),ETIME(50)) FIXED BIN(31) EXT INIT((50)0);     TCA00300
IF FTIME THEN DO;                                                     TCA00310
ALLOCATE LOTIME; LOTIME=0;                                            TCA00320
BEGTIME=RTIMER;         END;                                          TCA00330
/**********************************************************************/ TCA00340
                                                                      TCA00350
  ALLOCATE MSG;                                                       TCA00360
  MSG.PTR=RP; /*CHAIN TO ROOT PTR OF ARG LIST*/                       TCA00370
  MSG.RTN_ADDR.LEVEL=THISVP->VP.LEVEL;                                TCA00380
  MSG.RTN_ADDR.VPID=THISVP->VP.VPID;                                  TCA00390
  MSG.RTN_ADDR.BOXID=1;                                               TCA00400
  CALL SVCS1 (PROC_NAME,P,RTN_CODE);                                  TCA00410
  IF RTN_CODE ^=0 THEN PUT SKIP LIST ('ERROR IN SVC SEND TO ',        TCA00420
        PROC_NAME);                                                   TCA00430
                                                                      TCA00440
  CALL WAIT (1);                                                      TCA00450
  TP=THISVP->VP.WAIT.MSG->MSG.PTR;        /* CALLED PROC RETURNS */   TCA00460
                                                                      TCA00470
 /********************* TIMING REPORT OPTION *********************/   TCA00480
 IF FTIME THEN DO;                                                    TCA00490
 BEGTIME=RTIMER-BEGTIME;                                              TCA00500
 ETIME(ENUM)=ETIME(ENUM)+BEGTIME;/* ACCUMULATE TOTAL TIME */         TCA00510
 ECOUNT(ENUM)=ECOUNT(ENUM)+1;/* ACCUMULATE COUNT */                  TCA00520
 ELO(ENUM)=ELO(ENUM)+LOTIME;/* ACCUMULATE LOWER LEVEL PROC TIME */   TCA00530
 FREE LOTIME; IF ALLOCATION(LOTIME)^= 0 THEN LOTIME=LOTIME+BEGTIME;   TCA00540
 END;                                                                 TCA00550
```

```
 /*********************************************************************/  TCA00560
                                                                          TCA00570
   /******************** TCALL TRACE ********************************/  TCA00580
   IF FTCALL (I) THEN PUT EDIT ('*** ',PROC_NAME,' RETURNS') (A,A,A);    TCA00590
   /*****************************************************************/  TCA00600
                                                                          TCA00610
   FREE THISVP->VP.WAIT.MSG->MSG;                                         TCA00620
 RETURN;                                                                  TCA00630
 END /*TCALL*/;                                                           TCA00640
```

```
 TRTN: PROC(PROC_ADDR, RTP);                                              TRT00010
                                                                          TRT00020
 /**** A SERVICE ROUTINE USED BY A T-PROC WHEN IT FINISHES PROCESSING     TRT00030
  **** IT ADDS A MSG TOKEN TO A RETURN ARGUMENT LIST AND CALLS SVCS2       TRT00040
 ***    AND THEN FINISHES ITSELF ****/                                    TRT00050
                                                                          TRT00060
 /*** PROC_ADDR CONTAINS THE ADDRS OF THE CALLING PROCEDURE TO RETURN     TRT00070
  *** THE RETURN MSG TO; RTP POINTS TO THE RETURN ARGUMENT LIST ***/       TRT00080
                                                                          TRT00090
                                                                          TRT00100
  %INCLUDE USERS,VPX,ESVCS,FDEBUG;                                        TRT00110
  DCL (P,RTP) PTR, RTN_CODE FIXED BIN;                                    TRT00120
  DCL 1 T BASED (P),                                                      TRT00130
      2 LEN FIXED BIN (15);                                               TRT00140
                                                                          TRT00150
   ALLOCATE MSG;                                                          TRT00160
   MSG.PTR=RTP; /*CHAIN TO ROOT PTR OF ARG LIST*/                         TRT00170
   MSG.RTN_ADDR.LEVEL=THISVP->VP.LEVEL; /*RTN_ADDR NOT IMPORTANT**/       TRT00180
   MSG.RTN_ADDR.VPID=THISVP->VP.VPID;                                     TRT00190
   MSG.RTN_ADDR.BOXID=1;  /*ALWAYS USE BOX 1*/                           TRT00200
   CALL SVCS2 (PROC_ADDR,P,RTN_CODE);                                     TRT00210
   CALL FINISH;  /*COMMIT SUICIDE*/              %INCLUDE HEX;           TRT00220
  END /*TRTN*/;                                                           TRT00230
```

```
UPCT: PROC(PSETID,PINFO,RTN_CODE);                                    UPC00010
 /* A SUBROUTINE WHICH ACCEPTS THE PSETID AND A DECODED PINFO         UPC00020
    AND REPLACES THE PCAT ENTRY */                                    UPC00030
                                                                      UPC00040
%INCLUDE DPCAT,FDEBUG,DBEU;                                           UPC00050
%INCLUDE ESUB1;                                                       UPC00060
DCL P PTR, ADDR BUILTIN, RTN_CODE FIXED BIN;                          UPC00070
DCL PSETID BIT (32);                                                  UPC00080
DCL BIT_STRING BIT(144) BASED;                                        UPC00090
                                                                      UPC00100
CALL RET1 (PSETID,P,RTN_CODE);                                        UPC00110
BEU.DATA=ADDR(PINFO)->BIT_STRING;                                     UPC00120
CALL REP1 (PSETID,P,RTN_CODE);                                        UPC00130
FREE BEU;                                                             UPC00140
RTN_CODE=0;                                                           UPC00150
                                                                      UPC00160
RETURN;                                                               UPC00170
END;                                                                  UPC00180
```

```
UPDE:PROC;                                                              UPD00010
                                                                        UPD00020
 /*T-PROC TO PROCESS UPDE **/                                           UPD00030
%INCLUDE AUPDE,FDEBUG,SERVICE,AUPDN,ARETE;                              UPD00040
%INCLUDE DECAT;                                                         UPD00050
%INCLUDE EGECT,EGACT,EVTPE;                                             UPD00060
%INCLUDE EALLA; /* USES S-PROC ALLA TO OBTAIN ALL ATTRS */             UPD00070
                                                                        UPD00080
DCL ANAMES(25) CHAR (8);                                                UPD00090
DCL (N,NODE) FIXED BIN;                                                 UPD00100
DCL (P,TP,RP,LP,TP1,RTP,EP,EP1)PTR, NULL BUILTIN;                       UPD00110
DCL(RTN_CODE,I) FIXED BIN, NUM FIXED BIN (31);                          UPD00120
DCL BDATA BIT (320), CDATA CHAR (40), DLEN FIXED BIN;                   UPD00130
DCL TO_BIT FIXED BIN INIT (1), TO_CHAR, FIXED BIN INIT (2);             UPD00140
DCL TO_BIT_CHK FIXED BIN INIT (3);                                      UPD00150
DCL (ENAME,ANAME,ENAME1) CHAR (8);                                      UPD00160
DCL 1 MAINFO (25), /*AN ARRAY OF AINFO TO STORE RETRIEVED ACAT ENT*/    UPD00170
      2 BSETID BIT (32),  /*BSETID OF THIS ATTRIBUTE**/                 UPD00180
      2 PSETID BIT (32),  /*PSETID OF TARGET NODE**/                    UPD00190
      2 ENAME CHAR (8),/*IF TARGET IS A ENTITY SET*/                    UPD00200
      2 MAX FIXED BIN (31),                                             UPD00210
      2 MIN FIXED BIN (31),                                             UPD00220
      2 MLEN FIXED BIN,                                                 UPD00230
      2 FUNC CHAR(1),                                                   UPD00240
      2 TYPE CHAR (1),                                                  UPD00250
      2 VTYPE CHAR (1);                                                 UPD00260
  DCL ROOT_OP FIXED BIN;                                                UPD00270
                                                                        UPD00280
/*BEGIN PROCESSING**/                                                   UPD00290
CALL TBEG (PROC_ADDR,P);                                                UPD00300
ALLOCATE UPDE_RTN SET (RTP);                                            UPD00310
RTP->UPDE_RTN.RTN_CODE=0; /*INIT TO O.K.*/                              UPD00320
ENAME=UPDE_ARG.NAME;  ROOT_OP=UPDE_ARG.OP; /* REMEMBER ROOT OP */       UPD00330
MAINFO.ENAME(1)=ENAME; /*INIT AINFO ARRAY'S FIRST TOKEN*/               UPD00340
                                                                        UPD00350
/**ALLOCATE ROOT NODE**/                                                UPD00360
 CALL GECT (ENAME,EINFO,RTN_CODE); /*GECT GETS ENTITY CAT ENTRY */      UPD00370
 IF RTN_CODE ^= 0                                                       UPD00380
 THEN DO;                                                               UPD00390
    RTP->UPDE_RTN.RTN_CODE = 100; /**100 IS NO SUCH ENTITY**/           UPD00400
    CALL TRTN (PROC_ADDR,RTP);    /**RETURNS**/                        UPD00410
    RETURN;                                                             UPD00420
    END;                                                                UPD00430
                                                                        UPD00440
 ALLOCATE UPDN_ARG SET (TP);                                            UPD00450
 RP,LP=TP;                                                              UPD00460
 TP->UPDN_ARG.OP=UPDE_ARG.OP;                                           UPD00470
 TP->UPDN_ARG.PSETID=EINFO;                                             UPD00480
                                                                        UPD00490
/*FOLLOW UPDE TREE **/                                                  UPD00500
DO P=UPDE_ARG.PTR REPEAT P->UPDE_ARG.PTR WHILE (P^=NULL);               UPD00510
   ALLOCATE UPDN_ARG SET (TP); /* ALLOCATE UPDN TOKENS */               UPD00520
   LP->UPDN_ARG.PTR=TP; LP=TP;                                          UPD00530
   TP->UPDN_ARG=UPDE_ARG, BY NAME;                                      UPD00540
   TP->UPDN_ARG.CBTP=RP->UPDN_ARG.CBTP; /*FIX UP CBTP*/                 UPD00550
```

257

```
   TP->UPDN_ARG.PTR = NULL;                                                  UPD00560
   NODE=UPDE_ARG.NODE;                                                       UPD00570
   ENAME1=MAINFO.ENAME(UPDE_ARG.PARENT);                                     UPD00580
   ANAME=UPDE_ARG.NAME;                                                      UPD00590
   CALL GACT (ANAME,ENAME1,AINFO,RTN_CODE); /*GACT GETS ATTR CAT ENT*/       UPD00600
   IF RTN_CODE^=0                                                            UPD00610
   THEN DO;                                                                  UPD00620
      RTP->UPDE_RTN.RTN_CODE=NODE+100; /*100+ IS NO SUCH ATTRIBUTE***/       UPD00630
      CALL SVC3 (RP); /*FREE UP UPDN ALLOCATION UP TO NOW*/                  UPD00640
      CALL TRTN (PROC_ADDR, RTP); /*RETURNS**/                             UPD00650
      RETURN;                                                                UPD00660
      END;                                                                   UPD00670
   MAINFO(NODE)=AINFO; /*ENTER INTO AINFO ARRAY*/                            UPD00680
                                                                             UPD00690
                                                                             UPD00700
  /*TRANSFORM UPDE OP CODE INTO PROPER UPDN OP CODE*/                        UPD00710
   IF AINFO.TYPE='V' & UPDE_ARG.PARENT =1                                    UPD00720
   THEN DO;                                                                  UPD00730
      SELECT(UPDE_ARG.OP);                                                   UPD00740
         WHEN (UPDEOP.IST)  TP->UPDN_ARG.OP=UPDNOP.CI;                       UPD00750
         WHEN (UPDEOP.DEL)  TP->UPDN_ARG.OP=UPDNOP.DD;                       UPD00760
         WHEN (UPDEOP.REP)  TP->UPDN_ARG.OP=UPDNOP.REPP;                     UPD00770
         OTHERWISE;                                                          UPD00780
         END /*SELECT*/;                                                     UPD00790
      END /* IF V */;                                                       UPD00800
   IF UPDE_ARG.PARENT ^= 1 THEN  /* FOR ALL ATTR'S NOT DIRECTLY              UPD00810
                                    RELATED TO ROOT NODE */                  UPD00820
         TP->UPDN_ARG.OP=UPDNOP.ID;                                          UPD00830
                                                                             UPD00840
  /*VERIFY UPDATE DATA ***/                                                  UPD00850
   IF AINFO.TYPE='V' & UPDE_ARG.DLEN^=0                                      UPD00860
   THEN DO;/*WHEN THERE IS VAL*/                                             UPD00870
                                                                             UPD00880
     /*CHECK FOR UNIQUENESS OF KEY*/                                         UPD00890
      IF AINFO.FUNC='K'                                                      UPD00900
      THEN DO;                                                               UPD00910
         SELECT (TP->UPDN_ARG.OP);                                           UPD00920
            WHEN(UPDNOP.CI,UPDNOP.REPP) DO;                                  UPD00930
               ALLOCATE RETE_ARG SET (EP);                                   UPD00940
               EP->RETE_ARG.NAME=ENAME; /*IN CURRENT VERSION                 UPD00950
                        PARENT MUST BE ROOT NODE*/                           UPD00960
               EP->RETE_ARG.GET = RETEGET.ANY;                               UPD00970
               ALLOCATE RETE_ARG SET(EP1);                                   UPD00980
               EP->RETE_ARG.PTR=EP1;                                         UPD00990
               EP1->RETE_ARG.NAME=UPDE_ARG.NAME;                             UPD01000
               EP1->RETE_ARG.PARENT=1;                                       UPD01010
               EP1->RETE_ARG.NODE=2;                                         UPD01020
               EP1->RETE_ARG.N=1; /*1 PRED*/                                 UPD01030
               EP1->RETE_ARG.PRED.CDATA=UPDE_ARG.CDATA; /*COPY CDATA*/       UPD01040
               EP1->RETE_ARG.PRED.DLEN=UPDE_ARG.DLEN;                        UPD01050
               EP1->RETE_ARG.PRED.OP=RETEOP.EQ;                              UPD01060
               CALL TCALL ('RETE',1,EP,EP1);                                 UPD01070
               IF EP1->RETE_RTN.RTN_CODE^=0 | EP1->RETE_RTN.N^=0             UPD01080
               THEN DO; /*PROBLEM DATA, KEY VIOLATION*/                      UPD01090
                  IF FDEBUG.LEV1                                             UPD01100
```

```
                    THEN PUT SKIP LIST ('(UPDE) KEY VIOLATION');          UPDO1110
                    RTP->UPDE_RTN.RTN_CODE=300+UPDE_ARG.NODE;             UPDO1120
                     /*** RTN CODE 300+ IS VIOLATION OF KEY CONSTRAINT*/  UPDO1130
                    CALL SVC3 (EP); CALL SVC3 (EP1); CALL SVC3 (RP);      UPDO1140
                    CALL TRTN (PROC_ADDR,RTP);                            UPDO1150
                    RETURN;                                               UPDO1160
                    END /*RTN_CODE CHK*/;                                 UPDO1170
                                                                          UPDO1180
                  END /*IST OR REP*/;                                     UPDO1190
                OTHERWISE;                                                UPDO1200
                END /*SELECT*/;                                           UPDO1210
                                                                          UPDO1220
             END /*FUNC='K'*/;                                            UPDO1230
                                                                          UPDO1240
          /*CHECK DATA TYPE AND NUMERIC LIMITS*/                        UPDO1250
           CDATA=UPDE_ARG.CDATA;                                          UPDO1260
           DLEN=UPDE_ARG.DLEN;                                            UPDO1270
           CALL VTPE (AINFO,TO_BIT_CHK,BDATA,CDATA,DLEN,RTN_CODE);        UPDO1280
           IF RTN_CODE^=0                                                 UPDO1290
           THEN DO; /*PROBLEMS**/                                         UPDO1300
              RTP->UPDE_RTN.RTN_CODE=NODE; /*1+ IS ILLEGAL NODE DATA*/    UPDO1310
              CALL SVC3(RP); /*FREE UP UPDN ALLOCATION UP TO NOW*/        UPDO1320
              CALL TRTN (PROC_ADDR, RTP); /*RETURNS*/                    UPDO1330
              RETURN;                                                     UPDO1340
              END /*RTN_CODE CHK*/;                                       UPDO1350
           END /*VERIFY DATA*/;                                           UPDO1360
                                                                          UPDO1370
                                                                          UPDO1380
     /*REPLACE ANAME WITH PSETID AND BSETID AND FIX UP DATA**/           UPDO1390
       TP->UPDN_ARG.PSETID=AINFO.PSETID;                                  UPDO1400
       TP->UPDN_ARG.BSETID=AINFO.BSETID;                                  UPDO1410
       IF AINFO.TYPE='V' & UPDE_ARG.DLEN^=0                               UPDO1420
       THEN DO;                                                           UPDO1430
          TP->UPDN_ARG.DLEN=DLEN;                                         UPDO1440
          TP->UPDN_ARG.DATA=BDATA;                                        UPDO1450
          END;                                                            UPDO1460
       END /*DO P=UPDE_ARG.PTR...*/;                                      UPDO1470
                                                                          UPDO1480
     /* FIX UP THE DELETE ENTITY (DEE) OPERATION TO INCLUDE DELETION     UPDO1490
      OF ALL ITS ATTRIBUTES */                                            UPDO1500
                                                                          UPDO1510
    IF ROOT_OP = UPDEOP.DEE                                               UPDO1520
    THEN DO;                                                              UPDO1530
       CALL ALLA (ENAME, ANAMES, N, RTN_CODE);                            UPDO1540
       NODE = LP->UPDN_ARG.NODE; /* LP POINTS TO THE LAST TOKEN           UPDO1550
                IN THE UPDN_ARG CHAIN ALLCATED BY CALLER */               UPDO1560
                                                                          UPDO1570
       DO I = 1 TO N;                                                     UPDO1580
          CALL GACT (ANAMES (I), ENAME, AINFO, RTN_CODE);                 UPDO1590
          ALLOCATE UPDN_ARG;                                              UPDO1600
          LP->UPDN_ARG.PTR= P;LP = P;                                     UPDO1610
          UPDN_ARG.PSETID = AINFO.PSETID;                                 UPDO1620
          UPDN_ARG.BSETID = AINFO.BSETID;                                 UPDO1630
          UPDN_ARG.DLEN = 0;                                              UPDO1640
          UPDN_ARG.PARENT = 1; /* ALWAYS ROOT NODE */                     UPDO1650
```

```
      UPDN_ARG.NODE = NODE + I;                                          UPD01660
      IF AINFO.TYPE = 'V'                                                UPD01670
         THEN UPDN_ARG.OP = UPDNOP.DD;                                   UPD01680
         ELSE UPDN_ARG.OP=UPDNOP.DEL;                                    UPD01690
      END /* DO I */;                                                   UPD01700
   END /* IF DEE */;                                                    UPD01710
                                                                         UPD01720
/*CALL UP UPDN **/                                                       UPD01730
CALL TCALL ('UPDN',1,RP,TP);                                             UPD01740
                                                                         UPD01750
/*CHECK RETURNS*/                                                        UPD01760
I=TP->UPDN_RTN.RTN_CODE;                                                 UPD01770
IF I^=0 THEN                                                             UPD01780
   RTP->UPDE_RTN.RTN_CODE=I+200; /*200+ IS RTN CODE FROM UPDN**/         UPD01790
                                                                         UPD01800
 /*RETURNS***/                                                           UPD01810
CALL SVC3 (RP);                                                          UPD01820
CALL SVC3 (TP);                                                          UPD01830
CALL TRTN (PROC_ADDR, RTP);                                              UPD01840
RETURN;                                                                  UPD01850
                                                                         UPD01860
END /*UPDE*/;                                                           UPD01870
```

```
UPDN:PROC;                                                              UPD00010
/*************************************************************************  UPD00020
*                                                                       *  UPD00030
*                    MODULE    DESCRIPTION                            *  UPD00040
*************************************************************************/  UPD00050
/*************************************************************************  UPD00060
*****    PURPOSE:                                                         UPD00070
*****     THIS MODULE IS THE T-PROC AT THE N-ARY LEVEL WHICH              UPD00080
*****     PERFORMS UPDATES IN THE DATABASE; IT ACCEPTS UPDATE             UPD00090
*****     REQUESTS IN A TREE FORM (UPDN TREE), RESOLVES PREDICATES        UPD00100
*****     ON OCCURENCES TO BE UPDATED, AND CALLS SUBROUTINES TO           UPD00110
*****     PERFORM CRT,DEE AND MOD TYPES OF UPDATES.                       UPD00120
*************************************************************************  UPD00130
*****    METHOD:                                                          UPD00140
*****     1. MAKE A PASS THROUGH THE UPDN TREE TO FILL UP A TEMP          UPD00150
*****        DATA STRUCTURE WHICH IS USED TO REMEMBER ALL NODES           UPD00160
*****        WHICH HAVE ID CHILD NODES;                                   UPD00170
*****     2. USE TEMP, WORKING FROM BOTTOM OF THE TREE TO TOP,            UPD00180
*****        RESOLVE ALL PREDICATES (I.E. ID OPS) ON THE OCCURENCES       UPD00190
*****        OF DATA ELEMENTS TO BE UPDATED; REDUCE THE ORIGNAL           UPD00200
*****        UPDN TREE TO A 2-LEVEL UPDN TREE WITHOUT ID OPS;             UPD00210
*****      3. PASS THE REDUCED TREE TO SUBROUTINES CRTN,DELN OR           UPD00220
*****        MODN DEPENDING ON UPDN.OP AT THE ROOT NODE.                  UPD00230
*************************************************************************  UPD00240
*****    INPUT PARAMETERS:                                                UPD00250
*****       AS INDICATED IN UPDN_ARG;                                     UPD00260
*************************************************************************  UPD00270
*****    OUTPUT PARAMETERS:                                               UPD00280
*****       AS INDICATED IN UPDN_RTN;                                     UPD00290
*************************************************************************  UPD00300
*****    CALLING ARGUMENTS:                                             */ UPD00310
%INCLUDE AUPDN;                                                         UPD00320
/****    CALLS PROCEDURES:                                                UPD00330
*****     INTER-LEVEL T-PROC: NONE;                                       UPD00340
*****     INTRA-LEVEL T-PROC: RETN;                                     */ UPD00350
%INCLUDE ARETN;                                                         UPD00360
/****     INTRA-LEVEL S-PROC:  MODN,CRTN,DELN;                          */ UPD00370
%INCLUDE EMODN,ECRTN,EDELN;                                             UPD00380
/****     CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES:    */ UPD00390
%INCLUDE SERVICE,FDEBUG;                                                UPD00400
/*************************************************************************/ UPD00410
                                                                        UPD00420
 DCL (P,TP,RP,RTP,LP,MP) PTR;                                           UPD00430
 DCL RTN_CODE FIXED BIN;                                                UPD00440
 DCL NULL BUILTIN;                                                      UPD00450
 DCL 1 TEMP (20),  /*TEMP STRUCTURE, PARENT NODE# CAN BE UP TO 20*/     UPD00460
      2 K FIXED BIN,                                                    UPD00470
      2 IDN (5) FIXED BIN,                                              UPD00480
      2 IDP (5) PTR;                                                    UPD00490
 DCL(I,J,N) FIXED BIN;                                                  UPD00500
 DCL OFF_NODE(25) BIT(1) INIT ((25)('0'B));                             UPD00510
 DCL NPTR(25) PTR;                                                      UPD00520
                                                                        UPD00530
 /*BEGIN PROCESSING*/                                                   UPD00540
     CALL TBEG (PROC_ADDR,MP);                                          UPD00550
```

```
                                                                        UPD00560
      ALLOCATE UPDN_RTN SET (RTP);                                      UPD00570
      LP=NULL;                                                          UPD00580
                                                                        UPD00590
   /*TREE REDUCTION--REMOVE ALL 'ID' NODES*/                           UPD00600
      DO N = 1 TO 20; /*INITIALIZE TEMP STRUCTURE*/                     UPD00610
         TEMP.K(N) = 0;                                                 UPD00620
         DO J = 1 TO 5;                                                 UPD00630
            TEMP.IDN(N,J) = 0;                                          UPD00640
            TEMP.IDP(N,J) = NULL;                                       UPD00650
            END;                                                        UPD00660
         END;                                                           UPD00670
                                                                        UPD00680
  /***************** UPDN REQUEST TRACE ****************************/   UPD00690
  DCL FFFN BIT (1) EXT;                                                 UPD00700
  IF FFFN THEN DO;                                                      UPD00710
      PUT SKIP EDIT('NODE *PSETID* PARENT OP DLEN *BSETID* DATA')(A);   UPD00720
                                                                        UPD00730
      DO P=MP REPEAT P->UPDN_ARG.PTR WHILE (P^=NULL);                   UPD00740
      PUT SKIP EDIT(UPDN_ARG.NODE,HEX4(ADDR(UPDN_ARG.PSETID)->BP),      UPD00750
      UPDN_ARG.PARENT,UPDN_ARG.OP,                                      UPD00760
 UPDN_ARG.DLEN,'  ')(F(4),X(1),A(8),F(4),F(6),F(4),A);                  UPD00770
 IF UPDN_ARG.PARENT^=0 THEN                                             UPD00780
  PUT EDIT(HEX4(ADDR(UPDN_ARG.BSETID)->BP),' ')                         UPD00790
                    (A(8),A);                                           UPD00800
  ELSE PUT EDIT('        ')(A);                                         UPD00810
      DO II=1 TO UPDN_ARG.DLEN/4;DCL II FIXED BIN;                      UPD00820
         PUT EDIT(HEX4(ADDR(UPDN_ARG.DATA)->PAA(II)))(A(8));            UPD00830
   %INCLUDE HEX; DCL BP PTR BASED;                                      UPD00840
         END;                          DCL PAA(10) PTR BASED;         UPD00850
      END;  END;                                                        UPD00860
 /*********************************************************************/ UPD00870
                                                                        UPD00880
  /*FIRST MAKE A PASS THROUGH UPDN ARG TREE TO FILL UP TEMP STRUCTURE*/ UPD00890
      DO P=MP REPEAT P->UPDN_ARG.PTR WHILE (P^=NULL);                   UPD00900
                                                                        UPD00910
         NPTR(UPDN_ARG.NODE) = P; /* SAVE POINTER OF NODE TOKEN */      UPD00920
         IF UPDN_ARG.OP = UPDNOP.ID                                     UPD00930
         THEN DO;                                                       UPD00940
            N = UPDN_ARG.PARENT; /* INDEX TEMP BY PARENT NODE # */     UPD00950
            IF N > 20 /* TEMP ARRAY OVERFLOW */                        UPD00960
            THEN DO;                                                    UPD00970
               RTP->UPDN_RTN.RTN_CODE=1;                                UPD00980
               IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(UPDN) ERR RTN '     UPD00990
                           ,RTP->UPDN_RTN.RTN_CODE);                    UPD01000
               GO TO RTN;                                               UPD01010
               END;                                                     UPD01020
            ELSE;                                                       UPD01030
            TEMP.K(N) = TEMP.K(N) + 1;                                  UPD01040
            IF TEMP.K(N) > 5                                            UPD01050
            THEN DO; /* TEMP IDN OVERFLOW */                            UPD01060
               RTP->UPDN_RTN.RTN_CODE=2;                                UPD01070
               IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(UPDN) ERR RTN '     UPD01080
                           ,RTP->UPDN_RTN.RTN_CODE);                    UPD01090
               GO TO RTN;                                               UPD01100
```

```
            END;                                                          UPDO1110
         ELSE;                                                            UPDO1120
         TEMP.IDN(N,TEMP.K(N)) = UPDN_ARG.NODE;                           UPDO1130
         TEMP.IDP(N,TEMP.K(N)) = P;                                       UPDO1140
         IF LP=NULL                                                       UPDO1150
         THEN DO; /* ILLEGAL ROOT */                                     UPDO1160
            RTP->UPDN_RTN.RTN_CODE=3;                                     UPDO1170
            IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(UPDN) ERR RTN '          UPDO1180
                        , RTP->UPDN_RTN.RTN_CODE);                        UPDO1190
            GO TO RTN;                                                    UPDO1200
            END;                                                          UPDO1210
         ELSE;                                                            UPDO1220
         OFF_NODE(UPDN_ARG.NODE) = '1'B; /* MARK NODES OFF TREE */        UPDO1230
         END /* IF UPDN_ARG.OP */;                                        UPDO1240
      ELSE LP=P;                                                          UPDO1250
   END /* DO P = P */;                                                   UPDO1260
                                                                         UPDO1270
  /*FOR EACH PARENT NODE IN TEMP CALL RETN TO 'PIN POINT' THE            UPDO1280
    OCCURRENCE OF THE NODE*****/                                         UPDO1290
  DO N = 20 TO 1 BY -1; /*WORKS FROM THE END NODES UP*/                  UPDO1300
    IF TEMP.K(N) ^= 0                                                    UPDO1310
    THEN DO;                                                             UPDO1320
       I = 0; /*NO PRED FOR ROOT NODE*/  ALLOCATE RETN_ARG;              UPDO1330
       TP,RP = P;                                                        UPDO1340
       RETN_ARG.GET = RETNGET.ALL;                                       UPDO1350
       RETN_ARG.PSETID = NPTR(N)->UPDN_ARG.PSETID;                       UPDO1360
       DO J = 1 TO TEMP.K(N); /*FOR EACH OF ITS CHILD NODE*/             UPDO1370
          I = 1; ALLOCATE RETN_ARG;                                      UPDO1380
          TP->RETN_ARG.PTR = P; TP = P;                                  UPDO1390
          RETN_ARG.NODE = J + 1;                                         UPDO1400
          RETN_ARG.PSETID  = IDP(N,J)->UPDN_ARG.PSETID;                  UPDO1410
          RETN_ARG.BSETID  = IDP(N,J)->UPDN_ARG.BSETID;                  UPDO1420
          RETN_ARG.PRED.DLEN(1) = IDP(N,J)->UPDN_ARG.DLEN;               UPDO1430
          RETN_ARG.PRED.DATA(1) = IDP(N,J)->UPDN_ARG.DATA;               UPDO1440
          RETN_ARG.PARENT = 1;                                           UPDO1450
          RETN_ARG.N = 1;                                                UPDO1460
          RETN_ARG.GET = RETNGET.NO;                                     UPDO1470
          RETN_ARG.PRED.CN(1) = 1;                                       UPDO1480
          RETN_ARG.PRED.OP(1) = RETNOP.EQ;                               UPDO1490
          END /* J */;                                                  UPDO1500
    /****** CALL UP RETN *******/                                        UPDO1510
  CALL TCALL ('RETN',1,RP,TP);                                           UPDO1520
   CALL SVC3(RP); /*FREE CALLING ARG LIST*/                              UPDO1530
   IF TP->RETN_RTN.RTN_CODE ^=0 THEN DO;                                 UPDO1540
     /******RTN CODE 4 IS RETN CALL ERROR IN RETN *********/             UPDO1550
              RTP->UPDN_RTN.RTN_CODE=4;                                  UPDO1560
              IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(UPDN) ERR RTN '       UPDO1570
                        ,RTP->UPDN_RTN.RTN_CODE);                        UPDO1580
              CALL SVC3 (TP); /*FREE RTN MSG CHAIN*/                     UPDO1590
              GO TO RTN;                                                 UPDO1600
              END;                                                       UPDO1610
    /*CHECK NUMBER OF ELEMENTS RETURNED***/                            UPDO1620
    IF TP->RETN_RTN.N^=1 THEN DO ;                                       UPDO1630
     /******RTN CODE 10+N REPRESENTS THE NODE NUMBER OF A  *******      UPDO1640
     ******* PARENT NODE WHICH HAS AN ILLEGAL IDENTIFICATION TREE*/     UPDO1650
```

952

```
                RTP->UPDN_RTN.RTN_CODE=N+10;                                    UPD01660
                IF FDEBUG.LEV2  THEN PUT SKIP LIST ('(UPDN) ERR RTN '           UPD01670
                          ,RTP->UPDN_RTN.RTN_CODE);                             UPD01680
                CALL SVC3(TP); /*FREE RTN MSG CHAIN*/                           UPD01690
                GO TO RTN;                                                      UPD01700
                END;                                                            UPD01710
   P=TP->RETN_RTN.PTR; /*GET RETN DATA*/                                        UPD01720
   NPTR(N)->UPDN_ARG.DLEN=P->RETN_RTN1.DLEN;                                    UPD01730
   NPTR(N)->UPDN_ARG.DATA=P->RETN_RTN1.DATA;                                    UPD01740
   CALL SVC3(TP); /*FREE RTN MSG LIST*/                                         UPD01750
    END /*IF TEMP.K*/;                                                          UPD01760
    END /**N*/;                                                                 UPD01770
                                                                                UPD01780
   /*IN CURRENT VERSION BY NOW THE UPDN TREE HAS BEEN REDUCED TO A             UPD01790
     2-LEVEL TREE WITHOUT ID NODES *****/                                       UPD01800
   /**PERFORM UPDATES****/    P=MP;                                            UPD01810
   SELECT (UPDN_ARG.OP);                                                        UPD01820
     WHEN (UPDNOP.CRT) CALL CRTN (P,OFF_NODE,RTN_CODE);                         UPD01830
     WHEN (UPDNOP.DEE) CALL DELN (P, OFF_NODE,RTN_CODE);                        UPD01840
     WHEN (UPDNOP.MOD) CALL MODN (P, OFF_NODE,RTN_CODE);                        UPD01850
     OTHERWISE DO ; /*RTN CODE 31 IS ILLEGAL ROOT NODE OP*********/             UPD01860
                RTP->UPDN_RTN.RTN_CODE=31;                                      UPD01870
                IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(UPDN) ERR RTN '            UPD01880
                          , RTP->UPDN_RTN.RTN_CODE);                            UPD01890
                GO TO RTN;                                                      UPD01900
                END;                                                            UPD01910
      END /*SELECT**/;                                                          UPD01920
                                                                                UPD01930
      /*****CHECK SUBROUTINE RTN CODE *******/                                 UPD01940
      IF RTN_CODE^=0 THEN DO; /*RTN CODE 40+ IS FROM SUBR CALL***/              UPD01950
      RTP->UPDN_RTN.RTN_CODE=RTN_CODE+40;                                       UPD01960
      IF FDEBUG.LEV2 THEN PUT SKIP LIST ('(UPDN) ERR RTN ',                     UPD01970
                     RTP->UPDN_RTN.RTN_CODE);                                   UPD01980
      END;                                                                      UPD01990
                                                                                UPD02000
      /**SET UP RTN MSG*****/                                                  UPD02010
  RTN: CALL TRTN(PROC_ADDR, RTP);                                               UPD02020
                                                                                UPD02030
 END /*UPDN*/;                                                                  UPD02040
```

```
UPE1:PROC (OPE,T,OP,RTN_CODE);                                          DUM00010
                                                                        DUM00020
/***  SUBROUTINE TO GET DATA FROM USER AND BY MAKING USE OF             DUM00030
 ***** THE T,OPE,OP DATA STRUCTURES PASSED TO IT FROM THE CALLER        DUM00040
 ****** ABLE TO GENERATE PROPER CALL TO UPDE AT THE NEXT LEVEL  **/     DUM00050
                                                                        DUM00060
%INCLUDE AUPDE,SERVICE,ELEX;                                            DUM00070
DCL 1 T,                /*DATA STRUCTURE OF NAMES */                    DUM00080
     2 ENAME CHAR(8),                                                   DUM00090
     2 TN FIXED BIN, /*NUMBER OF ATTR*/                                 DUM00100
     2 T1 (25),                                                         DUM00110
       3 PARENT FIXED BIN,                                              DUM00120
       3 ANAME CHAR (8);                                                DUM00130
DCL RTN_CODE FIXED BIN;                                                 DUM00140
DCL (OP(25), OPE) CHAR(1);                                              DUM00150
DCL (P,RP,LP,TP) PTR, NULL BUILTIN;                                     DUM00160
DCL CODE FIXED BIN, K FIXED BIN;                                        DUM00170
DCL NA(25) BIT(1);                                                      DUM00180
DCL 1 TOKEN (25) ,                                                      DUM00190
      2 D CHAR(40),                                                     DUM00200
      2 L FIXED BIN; /*LENGTH OF TOKEN*/                                DUM00210
DCL N FIXED BIN , /*SERVE AS TOKEN COUNT*/ I FIXED BIN;                 DUM00220
                                                                        DUM00230
/*INITIALIZE NA STRUCTURE*/                                             DUM00240
DO I=1 TO 25;                                                           DUM00250
   NA(I)='0'B;                                                          DUM00260
   END;                                                                 DUM00270
DO I=1 TO TN;                                                           DUM00280
   IF T1.PARENT(I)^=0 THEN NA(T1.PARENT(I))='1'B;                       DUM00290
   END;                                                                 DUM00300
                                                                        DUM00310
/*BEGIN PROCESSING*/                                                    DUM00320
DO WHILE ('1'B); /*DO FOREVER*/                                         DUM00330
                                                                        DUM00340
   /*ALLOCATE ROOT NODE**/                                             DUM00350
   ALLOCATE UPDE_ARG;                                                   DUM00360
   LP,RP=P;                                                             DUM00370
   UPDE_ARG.NAME=T.ENAME;                                               DUM00380
   SELECT (OPE);                                                        DUM00390
   WHEN('M') UPDE_ARG.OP=UPDEOP.MOD;                                    DUM00400
   WHEN ('C') UPDE_ARG.OP=UPDEOP.CRT;                                   DUM00410
   WHEN('D') UPDE_ARG.OP=UPDEOP.DEE;                                    DUM00420
   OTHERWISE DO;                                                        DUM00430
      RTN_CODE=1; /****** 1 -- ILLEGAL ESET OP ****/                    DUM00440
      PUT SKIP LIST ('ILLEGAL SEMANTICS OF MODIFICATION STATEMENT');    DUM00450
      CALL SVC3(RP);                                                    DUM00460
      RETURN;                                                           DUM00470
      END;                                                              DUM00480
   END /*SELECT*/;                                                      DUM00490
   K=0; /*INIT DATA COUNT*/                                             DUM00500
                                                                        DUM00510
   A:                                                                   DUM00520
   CALL LEX (N,TOKEN,0,RTN_CODE);                                       DUM00530
   IF RTN_CODE^=0                                                       DUM00540
   THEN DO;                                                             DUM00550
```

```
      PUT SKIP LIST ('ILLEGAL SYNTAX, PLEASE RE-ENTER LINE');          DUM00560
      GO TO A; END;                                                    DUM00570
                                                                       DUM00580
   IF N=0 THEN LEAVE; /*NO MORE DATA*/                                 DUM00590
                                                                       DUM00600
   /*FILL UP UPDE REQUEST*/                                            DUM00610
   DO I=1 TO TN;                                                       DUM00620
      ALLOCATE UPDE_ARG;                                               DUM00630
      LP->UPDE_ARG.PTR=P;   LP=P;                                      DUM00640
      UPDE_ARG.NODE=I+1; /*ASSIGN NODE NUMBER*/                        DUM00650
      UPDE_ARG.PARENT=T1.PARENT(I)+1;                                  DUM00660
      UPDE_ARG.NAME=T1.ANAME(I);                                       DUM00670
      SELECT (OP(I));                                                  DUM00680
         WHEN ('I') UPDE_ARG.OP=UPDEOP.ID;                             DUM00690
         WHEN ('N') UPDE_ARG.OP=UPDEOP.IST;                            DUM00700
         WHEN ('D') UPDE_ARG.OP=UPDEOP.DEL;                            DUM00710
         WHEN ('R') UPDE_ARG.OP=UPDEOP.REP;                            DUM00720
         OTHERWISE DO;                                                 DUM00730
            RTN_CODE=2;                                                DUM00740
            PUT SKIP LIST ('ILLEGAL SEMANTICS OF MODIFICATION STMT.'); DUM00750
            CALL SVC3(RP);                                             DUM00760
            RETURN;                                                    DUM00770
            END;                                                       DUM00780
         END /*SELECT*/;                                               DUM00790
                                                                       DUM00800
      IF NA(I)='0'B THEN DO; /*FILL IN DATA*/                          DUM00810
         K=K+1;                                                        DUM00820
         IF K>N THEN DO;                                               DUM00830
            PUT SKIP LIST ('INSUFFICIENT DATA; REENTER LINE');         DUM00840
            CALL SVC3 (RP);                                            DUM00850
            GO TO END;                                                 DUM00860
            END;                                                       DUM00870
         UPDE_ARG.CDATA=TOKEN.D(K);                                    DUM00880
         UPDE_ARG.DLEN=TOKEN.L(K);                                     DUM00890
         END;                                                          DUM00900
      END /*DO I*/;                                                    DUM00910
                                                                       DUM00920
   /*ALL NODES ARE ALLOCATED*/                                         DUM00930
   CALL TCALL ('UPDE',1,RP,TP);                                        DUM00940
   CALL SVC3 (RP); /*FREE ARG LIST*/                                   DUM00950
   CODE=TP->UPDE_RTN.RTN_CODE;                                         DUM00960
   IF CODE^=0                                                          DUM00970
   THEN DO;         /* CHECK UPDE RTN CODE: IGNORES 100-200 RANGE */   DUM00980
      IF CODE > 300                                                    DUM00990
      THEN PUT SKIP LIST ('KEY VIOLATION');                            DUM01000
      ELSE IF CODE > 200                                               DUM01010
      THEN PUT SKIP EDIT ('RETN RTN_CODE ',CODE-200)(A,F(3));          DUM01020
      ELSE PUT SKIP EDIT ('ILLEGAL DATA FOR ATTRIBUTE ',               DUM01030
           T.T1.ANAME(CODE-1))(A,A);                                   DUM01040
      PUT SKIP LIST ('DATA ENTERED IGNORED');                          DUM01050
      END /* RTN CODE CHECK */;                                       DUM01060
                                                                       DUM01070
   CALL SVC3 (TP);   /*FREE RETURN LIST*/                              DUM01080
   END:                                                                DUM01090
   END /*DO WHILE*/;                                                   DUM01100
```

157

```
RETURN;                                                    DUMO1110
END /*UPE1*/;                                              DUMO1120
```

```
USER:PROC;                                                              USE00010
/*************************************************************************USE00020
*                                                                       *USE00030
*                    MODULE    DESCRIPTION                            *USE00040
*************************************************************************/USE00050
/*************************************************************************USE00060
*****   PURPOSE:                                                        USE00070
*****   A T-PROC FOR THE USER TO SELECT SUBSYSTEMS AFTER LOGGIN IN      USE00080
*************************************************************************USE00090
*****   CALLS PROCEDURES:                                               USE00100
*****   DBA, BNV, RELV:  INTRA-LEVEL S-PROC REPRESENTING DBA MODE,      USE00110
*****    BASE DATA MODEL VIEW, AND RELATIONAL DATA MODEL VIEW.          USE00120
*****    PRESENTLY, THE LATTER TWO ARE NOT AVAILABLE.                   USE00130
*************************************************************************/USE00140
                                                                        USE00150
%INCLUDE EDBA.SERVICE;  /* ENTRY DCLS OF SUBSYSTEMS*/                   USE00160
DCL NAME CHAR(7),P PTR;                                                 USE00170
                                                                        USE00180
/*BEGIN PROCESSING*/                                                    USE00190
CALL TBEG(PROC_ADDR,P);      /* JUST FOR CONVENTION */                  USE00200
                                                                        USE00210
DO WHILE ('1'B);                                                        USE00220
  PUT SKIP LIST ('SUBSYSTEMS: DBA, BV(BASE VIEW) OR RV(RELATIONAL VIEW)USE00230
?');                                                                    USE00240
  GET EDIT (NAME) (A(7));                                               USE00250
  IF NAME=(7)' ' THEN LEAVE;                                            USE00260
  SELECT (NAME);                                                        USE00270
    WHEN ('DBA','D')CALL DBA;                                           USE00280
    WHEN ('BV','B') CALL BNV;                                           USE00290
    WHEN ('RV','R')CALL RELV;                                           USE00300
    OTHERWISE PUT SKIP EDIT (NAME,'IS NOT A VALID COMMAND') (A,A);      USE00310
  END;                                                                  USE00320
END;                                                                    USE00330
/*IF NO SUBSYSTEM COMMAND IS GIVEN, THEN END SESSION*/                  USE00340
PUT SKIP LIST ('-- END OF USER SESSION--');                             USE00350
                                                                        USE00360
/*************************************************************************/USE00370
BNV: PROC;                                                              USE00380
PUT SKIP LIST ('BASE DATA MODEL VIEW SUBSYSTEM NOT AVAILABLE.');        USE00390
RETURN;                                                                 USE00400
END;                                                                    USE00410
                                                                        USE00420
RELV: PROC;                                                             USE00430
PUT SKIP LIST ('RELATIONAL DATA MODEL VIEW SUBSYSTEM NOT AVAILABLE.');  USE00440
RETURN;                                                                 USE00450
END;                                                                    USE00460
/*************************************************************************/USE00470
CALL TRTN (PROC_ADDR,P);                                                USE00480
END /*USER*/;                                                          USE00490
```

```
VERIF: PROC(PINFO,P,VOP,BDATA,RTN_CODE,YES);                          VER00010
                                                                      VER00020
  /*A S-PROC IN THE N-ARY LEVEL TO VERIFY AND CONVERT DATA ***/       VER00030
                                                                      VER00040
%INCLUDE DPCAT,DBEU;                                                  VER00050
%INCLUDE EPCT,ESUB1;                                                  VER00060
DCL BDATA BIT (*) VAR ALIGNED,                                        VER00070
 VOP BIT(8) ALIGNED, RTN_CODE FIXED BIN, YES BIT(1) ALIGNED;          VER00080
DCL PTYPE CHAR (1), BDATA1 BIT (320) VARYING;                         VER00090
DCL (NUM,NUM1) FIXED BIN (31);                                        VER00100
DCL 1 OP ALIGNED, /* SAME AS RETNOP */                               VER00110
     2 EQ BIT (8) INIT ('00000000'B),                                 VER00120
     2 GT BIT (8) INIT ('00000010'B),                                 VER00130
     2 LT BIT (8) INIT ('00000001'B);                                 VER00140
 DCL P PTR, NULL BUILTIN;                                             VER00150
 DCL UNSPEC BUILTIN;                                                  VER00160
                                                                      VER00170
/*BEGIN PROCESSING*/                                                  VER00180
 YES='0'B; /*INIT TO NO*/                                             VER00190
                                                                      VER00200
 PTYPE=PINFO.PTYPE;                                                   VER00210
                                                                      VER00220
 SELECT (PTYPE);                                                      VER00230
                                                                      VER00240
    WHEN('N') DO; /*CONVERSION*/                                      VER00250
  /*  PUT SKIP LIST ('(VERIF), WHEN N');     */                      VER00260
      UNSPEC(NUM)=BDATA;                                              VER00270
      UNSPEC (NUM1)=BEU.DATA;                                         VER00280
      SELECT (VOP);                                                   VER00290
                                                                      VER00300
         WHEN (OP.EQ) IF NUM1=NUM THEN YES='1'B;                      VER00310
         WHEN(OP.GT) IF NUM1>NUM THEN YES='1'B;                       VER00320
         WHEN (OP.LT) IF NUM1<NUM THEN YES='1'B;                      VER00330
         OTHERWISE DO;                                                VER00340
            RTN_CODE=3; /****3 IS ILLEGAL OP********/                 VER00350
            RETURN;                                                   VER00360
            END;                                                      VER00370
                                                                      VER00380
       END /*SELECT OP*/;                                             VER00390
     END /*WHEN N*/;                                                  VER00400
                                                                      VER00410
   OTHERWISE DO;  /*NO CONVERSION*/                                   VER00420
      BDATA1=BEU.DATA;                                                VER00430
      SELECT (VOP);                                                   VER00440
        WHEN (OP.EQ) IF BDATA1=BDATA THEN YES='1'B;                   VER00450
        WHEN (OP.GT) IF BDATA1>BDATA THEN YES='1'B;                   VER00460
        WHEN (OP.LT) IF BDATA1<BDATA THEN YES='1'B;                   VER00470
        OTHERWISE DO;                                                 VER00480
           RTN_CODE=3; /***3 IS ILLEGAL OP*/                          VER00490
           RETURN;                                                    VER00500
           END;                                                       VER00510
        END /*SELECT OP*/;                                            VER00520
     END /*OTHERWISE*/;                                               VER00530
   END /*SELECT PTYPE*/;                                              VER00540
 END/*VERIF*/;                                                        VER00550
```

```
VINIT:PROC;                                                          VIN00010
/*********************************************************************  VIN00020
*                                                                   *  VIN00030
*                    MODULE    DESCRIPTION                          *  VIN00040
*********************************************************************/ VIN00050
/*********************************************************************  VIN00060
*****   PURPOSE: INITIALIZES THE ENTITY LEVEL EITHER FROM AN          VIN00070
*****    EXISTING FILE OR FROM SCRATCH.                               VIN00080
*********************************************************************  VIN00090
*****   METHOD:                                                       VIN00100
*****   1. DETERMINE WHETHER IT IS A FILE INIT OR A NEW INIT;         VIN00110
*****   2. IF NEW INIT, CALL DEFP (THROUGH DFP1) AND DEFB (THROUGH    VIN00120
*****      DFB1) TO CREATE PSETS AND BSETS USED IN STORING CATALOGUE  VIN00130
*****      ENTRIES AT THIS LEVEL; THE PSETID'S AND BSETID'S RETURNED  VIN00140
*****      ARE SAVED IN A STATIC STRUCTURE CALLED KEY, TO BE          VIN00150
*****      SHARED BY RELEVANT MODULES AT THIS LEVEL.                  VIN00160
*****   3. KEY IS INSERTED IN TO AN ANCHOR PSET BY CALLING UPDN;      VIN00170
*****   4. PSETID OF ANCHOR PSET IS PASSED TO THE N-ARY LEVEL         VIN00180
*****      IN NINIT (OP=STK) CALL.                                    VIN00190
*****   5. IF FILE INIT THEN  OBTAIN THE PSETID  OF THE ANCHOR        VIN00200
*****      PSET FROM NINIT_RTN AND IN TURN RETRIEVE ALL KEYS.         VIN00210
*********************************************************************  VIN00220
*****   INPUT PARAMETERS:                                             VIN00230
*****    AS INDICATED IN VINIT_ARG;                                   VIN00240
*********************************************************************  VIN00250
*****   OUTPUT PARAMETERS:                                            VIN00260
*****    AS INDICATED IN VINIT_RTN;                                   VIN00270
*********************************************************************  VIN00280
*****    CALLING ARGUMENTS:                                         */ VIN00290
%INCLUDE AVINI;                                                      VIN00300
/****    CALLS PROCEDURES:                                            VIN00310
*****    INTER-LEVEL T-PROC: NINIT,RETN,UPDN;                       */ VIN00320
%INCLUDE ANINI,ARETN,AUPDN;                                          VIN00330
/****    INTRA-LEVEL T-PROC: NONE;                                    VIN00340
*****    INTRA-LEVEL S-PROC: DFP1,DFB1;                             */ VIN00350
%INCLUDE EDFP1, EDFB1;                                               VIN00360
/****    CONTROL STRUCTURE, PANEL MANAGER AND DEBUGGING FACILITIES: */ VIN00370
%INCLUDE SERVICE,FDEBUG;                                             VIN00380
/****    EXTERNAL STATIC VARIABLES TO HOLD STATIC DATA: KEY         */ VIN00390
%INCLUDE DKEY;                                                       VIN00400
*********************************************************************/ VIN00410
                                                                     VIN00420
DCL (P,TP,RP,MP,RTP) PTR;                                            VIN00430
DCL NAME CHAR(8), I FIXED BIN;                                       VIN00440
DCL NULL BUILTIN;                                                    VIN00450
DCL RTN_CODE FIXED BIN (15);                                         VIN00460
DCL 1 PC(7), /*DATA FOR DEF OF 7 PSETS IN CATALOGUE; SEQUENCE IS     VIN00470
              ESET,ENM,EINFO,ASET,ANM AINFO,KEY*/                    VIN00480
     2 PTYPE CHAR (1) INIT ('X','C','B','X','C','B','B'),           VIN00490
     2 PLEN FIXED BIN INIT (0,8,4,0,8,29,4),                         VIN00500
     2 TYPE CHAR(1) INIT ('E','V','V','E','V','V','V'),              VIN00510
     2 PSETID BIT (32);                                              VIN00520
DCL 1 BC(5), /*DATA FOR DEF OF 5 BSETS IN CATALOGUE; SEQUENCE IS     VIN00530
              ENAME,EINFO,ESET,ANAME,AINFO*/                         VIN00540
     2 FUNC CHAR(1) INIT ('S','S','M','S','S'),                      VIN00550
```

```
      2 PSETID (2) BIT(32),                                          VIN00560
      2 RTN_CODE FIXED BIN,                                          VIN00570
      2 BSETID BIT (32);                                             VIN00580
 DCL AKEY BIT(32);                                                   VIN00590
                                                                     VIN00600
 /*BEGIN PROCESSING*/                                               VIN00610
      CALL TBEG (PROC_ADDR,MP);                                      VIN00620
                                                                     VIN00630
      ALLOCATE VINIT_RTN SET (RTP);                                  VIN00640
      RTP->VINIT_RTN.RTN_CODE=0; /*INITIALIZE RTN CODE TO O.K.*/     VIN00650
   /*SET UP MSG TO CALL NINT */                                      VIN00660
      NAME=MP->VINIT_ARG.FNAME; /*GET FILE NAME*/                   VIN00670
      ALLOCATE NINIT_ARG;                                            VIN00680
      NINIT_ARG.FNAME=NAME;                                          VIN00690
      NINIT_ARG.OP='I'; /*INITIALIZE NEXT LEVEL*/                    VIN00700
      CALL TCALL ('NINIT', 1,P,TP);                                  VIN00710
      CALL SVC3 (P); /*FREE CALLING ARG*/                            VIN00720
                                                                     VIN00730
   IF TP->NINIT_RTN.RTN_CODE ^=0 THEN DO;                            VIN00740
         RTP->VINIT_RTN.RTN_CODE=1; /*SET RTN CODE INVALID*/         VIN00750
         CALL SVC3(TP); /*FREE RTN MSG*/                             VIN00760
         GO TO RTN; /*GO TO END*/                                    VIN00770
         END;  /*END OF STANDARD MSG PROCESSING*/                    VIN00780
                                                                     VIN00790
   /*PROCESS INITIALIZATION OF CURRENT LEVEL*/                       VIN00800
    IF NAME=(8)' ' THEN /*'NEW' INIT*/                               VIN00810
    DO;  /*NEW INIT: DEFINE CATALOGUES*/                            VIN00820
                                                                     VIN00830
       CALL SVC3 (TP); /*NINIT_RTN NO LONGER NEEDED*/                VIN00840
       DO I=1 TO 7; /*DEFINE 7 PSETS*/                               VIN00850
       CALL DFP1(PC.PTYPE(I), PC.PLEN(I), PC.TYPE(I), RTN_CODE,      VIN00860
            PC.PSETID(I));                                           VIN00870
       IF RTN_CODE ^=0 THEN DO;                                      VIN00880
          RTP->VINIT_RTN.RTN_CODE =1;                                VIN00890
          IF FDEBUG.LEV1 THEN PUT SKIP LIST ('VINIT: DFP1 ABEND');   VIN00900
          GO TO RTN;                                                 VIN00910
          END;                                                       VIN00920
       IF I=7 THEN AKEY=PC.PSETID(I);                                VIN00930
        ELSE KEY(I) = PC.PSETID(I);                                  VIN00940
       END /*DEFINE 7 PSETS*/;                                       VIN00950
                                                                     VIN00960
       /*SET PSETID FOR BC STRUCTURE*/                               VIN00970
       BC.PSETID(1,1)=KEY(1);                                        VIN00980
       BC.PSETID(1,2)=KEY(2);                                        VIN00990
       BC.PSETID(2,1)=KEY(1);                                        VIN01000
       BC.PSETID(2,2)=KEY(3);                                        VIN01010
       BC.PSETID(3,1)=KEY(4);                                        VIN01020
       BC.PSETID(3,2)=KEY(1);                                        VIN01030
       BC.PSETID(4,1)=KEY(4);                                        VIN01040
       BC.PSETID(4,2)=KEY(5);                                        VIN01050
       BC.PSETID(5,1)=KEY(4);                                        VIN01060
       BC.PSETID(5,2)=KEY(6);                                        VIN01070
       DO I=1 TO 5; /*DEFINE 5 BSETS*/                               VIN01080
       CALL DFB1(BC.FUNC(I),BC.PSETID(I,1),BC.PSETID(I,2),           VIN01090
        BC.RTN_CODE(I),BC.BSETID(I));                                VIN01100
```

```
      IF BC.RTN_CODE (I) ^=0 THEN DO;                                  VINO1110
            RTP->VINIT_RTN.RTN_CODE=1;                                 VINO1120
            GO TO RTN;END;                                             VINO1130
      KEY(I+6)=BC.BSETID(I);                                           VINO1140
      END /*DEFINE 5 BSETS*/;                                          VINO1150
                                                                       VINO1160
  /*STORE 11 KEYS INTO THE ANCHOR PSET*/                               VINO1170
 DO I=1 TO 11;                                                         VINO1180
     ALLOCATE UPDN_ARG;                                                VINO1190
     UPDN_ARG.PSETID=AKEY;                                             VINO1200
     UPDN_ARG.OP=UPDNOP.CRT;                                           VINO1210
     UPDN_ARG.DLEN=4;                                                  VINO1220
     UPDN_ARG.DATA=KEY(I);                                             VINO1230
  CALL TCALL ('UPDN',1,P,TP);                                          VINO1240
  END/*I*/;                                                            VINO1250
                                                                       VINO1260
  CALL SVC3(P); /*FREE CALLING ARG LIST*/                              VINO1270
  IF TP-> UPDN_RTN.RTN_CODE ^=0 THEN DO;                               VINO1280
       RTP->VINIT_RTN.RTN_CODE=1; CALL SVC3(TP);                       VINO1290
       GO TO RTN;                                                      VINO1300
       END;                                                            VINO1310
                                                                       VINO1320
  CALL SVC3(TP); /*FREE RTN MSG CHAIN*/                                VINO1330
                                                                       VINO1340
 /*CALL UP THE NEXT LEVEL TO STORE KEY TO THE ANCHOR SET*/             VINO1350
  ALLOCATE NINIT_ARG;                                                  VINO1360
  NINIT_ARG.KEY=AKEY;                                                  VINO1370
  NINIT_ARG.OP='S'; /*ASKS NINIT TO STORE KEY*/                        VINO1380
  CALL TCALL ('NINIT',1,P,TP);                                         VINO1390
                                                                       VINO1400
  CALL SVC3(P); /*FREE CALLING ARG LIST*/                              VINO1410
  IF TP->NINIT_RTN.RTN_CODE ^=0 THEN DO;                               VINO1420
     RTP->VINIT_RTN.RTN_CODE=1; CALL SVC3 (TP);                        VINO1430
     GO TO RTN;                                                        VINO1440
     END;                                                              VINO1450
  CALL SVC3(TP); /*FREE RTN MSG LIST*/                                 VINO1460
                                                                       VINO1470
  /*** BOOTSTRAP CATALUGES HERE *******/                              VINO1480
 CALL EBOT;/*EBOT  IS A SUBROUTINE WHICH BOOTSTRAP CAT ENTRIES*/       VINO1490
                                                                       VINO1500
 END /*NEW INIT PROCESSING*/;                                          VINO1510
                                                                       VINO1520
ELSE DO; /*FILE INIT*/                                                 VINO1530
 /*OBTAIN ANCHOR KEY*/                                                 VINO1540
                                                                       VINO1550
  AKEY=TP->NINIT_RTN.KEY;                                              VINO1560
  CALL SVC3(TP);                                                       VINO1570
 /*OBTAIN THE REST OF THE 11 KEYS*/                                    VINO1580
    ALLOCATE RETN_ARG;                                                 VINO1590
    RETN_ARG.PSETID=AKEY;                                              VINO1600
    RETN_ARG.GET=RETNGET.ALL;                                          VINO1610
                                                                       VINO1620
 CALL TCALL ('RETN',1,P,TP);                                           VINO1630
  CALL SVC3(P);  /*FREE CALLING ARG*/                                  VINO1640
                                                                       VINO1650
```

```
   IF TP->RETN_RTN.RTN_CODE ^=0 THEN DO;                                VIN01660
      IF FDEBUG.LEV1 THEN PUT SKIP LIST ('INSIDE FILE INIT, JUST        VIN01670
   RETURNED FROM RETN, AND RTN_CODE IS NOT O.K.');                      VIN01680
      RTP->VINIT_RTN.RTN_CODE =1; CALL SVC3(TP);                        VIN01690
      GO TO RTN;                                                        VIN01700
      END;                                                              VIN01710
                                                                        VIN01720
     /*GET KEYS*/                                                       VIN01730
   P=TP->RETN_RTN.PTR;                                                  VIN01740
   DO I=1 TO 11;                                                        VIN01750
     KEY(12-I)=RETN_RTN1.DATA;                                          VIN01760
     P=RETN_RTN1.PTR ;                                                  VIN01770
     END;                                                               VIN01780
   CALL SVC3(TP); /*FREE RTN MSG LIST*/                                 VIN01790
                                                                        VIN01800
   END /*FILE INIT*/;                                                   VIN01810
                                                                        VIN01820
 RTN: CALL TRTN (PROC_ADDR,RTP);                                        VIN01830
 END /*VINIT*/;                                                         VIN01840
                                                                        VIN01850
```

```
VNME: PROC;                                                              VNM00010
                                                                         VNM00020
/* T-PROC AT THE ENTITY LEVEL, INVOKED BY DMB (BASE DATA MANIPULATION)   VNM00030
 SUBSYSTEM AT THE USER INTERFACE LEVEL, TO CHECK FOR LEGALITY OF NAMES   VNM00040
 STATED BY USER BEFORE ACCEPTING DATA */                                 VNM00050
                                                                         VNM00060
%INCLUDE SERVICE, AVNME, EGECT, EGACT, DECAT;                            VNM00070
DCL ENAME CHAR (8);                                                      VNM00080
DCL (P,RTP) PTR, RTN_CODE FIXED BIN, NULL BUILTIN;                       VNM00090
DCL  PARENT_ENAME(25) CHAR (8);  /* TEMP STRUCTURE */                    VNM00100
DCL I FIXED BIN (31);                                                    VNM00110
                                                                         VNM00120
/* BEGIN */                                                              VNM00130
CALL TBEG (PROC_ADDR,P);                                                 VNM00140
ALLOCATE VNME_RTN SET (RTP);                                             VNM00150
                                                                         VNM00160
CALL GECT (T.ENAME,EINFO, RTN_CODE);                                     VNM00170
IF RTN_CODE ^=0 THEN DO;                                                 VNM00180
   RTP->VNME_RTN.N=-1; /*** PROBLEM WITH ENTITY SET NAME *****/          VNM00190
   GO TO RTN;                                                            VNM00200
   END;                                                                  VNM00210
                                                                         VNM00220
/*VALIDATE ATTRIBUTES */                                                 VNM00230
DO I = 1 TO T.TN;                                                        VNM00240
   IF T.PARENT(I)=0 THEN ENAME=T.ENAME;                                  VNM00250
   ELSE ENAME = PARENT_ENAME (T.PARENT (I));                             VNM00260
   CALL GACT (T.ANAME (I), ENAME, AINFO, RTN_CODE);                      VNM00270
   IF RTN_CODE ^= 0 THEN DO;                                             VNM00280
      RTP->VNME_RTN.N=I; /*0+ IS PROBLEM WITH ATTRIBUTE NAME *****/      VNM00290
      GO TO RTN;                                                         VNM00300
      END;                                                               VNM00310
   IF AINFO.TYPE='E' THEN PARENT_ENAME (I)=AINFO.ENAME; /*REMEMBER       VNM00320
                TARGET ESET NAME */                                      VNM00330
   END /* DO I */ ;                                                      VNM00340
                                                                         VNM00350
  RTP->VNME_RTN.N=0; /* NO PROBLEM */                                    VNM00360
 RTN: CALL TRTN (PROC_ADDR, RTP);                                        VNM00370
 RETURN;                                                                 VNM00380
 END /* VNME */;                                                        VNM00390
```

```
VNM1: PROC (T, RTN_CODE, N);                                          DUM00010
/**SUBROUTINE TO SET UP CALL TO VNME**/                               DUM00020
%INCLUDE AVNME,SERVICE,FDEBUG;                                        DUM00030
DCL (RTN_CODE,N) FIXED BIN;                                           DUM00040
DCL 1 T,   /*DATA STRUCTURE OF NAMES TO BE VERIFIED      */           DUM00050
      2 ENAME CHAR(8),                                                DUM00060
      2 TN FIXED BIN, /*NUMBER OF ATTR*/                              DUM00070
      2 T1 (25),                                                      DUM00080
        3 PARENT FIXED BIN,                                           DUM00090
        3 ANAME CHAR (8);                                             DUM00100
DCL (TP,P)PTR, NULL BUILTIN;                                          DUM00110
                                                                      DUM00120
ALLOCATE VNME_ARG;                                                    DUM00130
VNME_ARG.T=T;                                                         DUM00140
                                                                      DUM00150
CALL TCALL('VNME',1,P,TP);                                            DUM00160
CALL SVC3(P);                                                         DUM00170
RTN_CODE= TP->VNME_RTN.RTN_CODE;       /* RTN_CODE NOT SIGNIFICANT */ DUM00180
N=TP->VNME_RTN.N;                                                     DUM00190
IF N ^= 0                              /* NAME ERROR */               DUM00200
THEN DO;                                                              DUM00210
   IF N = -1 THEN PUT SKIP LIST ('ILLEGAL ENTITY SET NAME');          DUM00220
   ELSE PUT SKIP EDIT ('ILLEGAL ATTRIBUTE NAME ',T.T1.ANAME(N))(A,A); DUM00230
   END;                                                               DUM00240
                                                                      DUM00250
RETURN; END;                                                          DUM00260
```

892

```
VTPE: PROC (AINFO, MODE, BDATA, CDATA, DLEN, RTN_CODE);                  VTP00010
                                                                         VTP00020
/*A S-PROC AT THE ENTITY LEVEL TO CHECK FOR DATA TYPE VALIDITY AND       VTP00030
 PERFORMS CONVERSION FROM CHAR TO BIT OR VICE VERSA DEPENDING ON         VTP00040
 THE MODE GIVING*/                                                       VTP00050
                                                                         VTP00060
%INCLUDE DECAT; /*FOR AINFO TEMPLATE*/                                   VTP00070
%INCLUDE FDEBUG;                                                         VTP00080
DCL MODE FIXED BIN;                                                      VTP00090
DCL TO_CHAR FIXED BIN INIT (2), TO_BIT FIXED BIN INIT (1);               VTP00100
DCL TO_BIT_CHK FIXED BIN INIT (3);                                       VTP00110
DCL CDATA CHAR (40), BDATA BIT (320), (DLEN, RTN_CODE) FIXED BIN;        VTP00120
DCL NUM FIXED BIN (31); /*TO HOLD CONVERTED FIXED BIN NUMERIC DATA*/     VTP00130
DCL VERIFY BUILTIN; DCL BIN BUILTIN;                                     VTP00140
DCL DATA CHAR (40) VARYING;                                              VTP00150
DCL (UNSPEC,SUBSTR)BUILTIN, I FIXED BIN;                                 VTP00160
                                                                         VTP00170
/*BEGIN PROCESSING***/                                                   VTP00180
RTN_CODE=0;/*INIT TO O.K.*/                                              VTP00190
SELECT (MODE);                                                           VTP00200
                                                                         VTP00210
WHEN (TO_BIT, TO_BIT_CHK) DO;                                            VTP00220
/*FROM CHAR STRING TO BIT STRING*/                                       VTP00230
                                                                         VTP00240
 IF AINFO.TYPE='V' THEN DO; /*FOR V TYPE NODE ONLY*/                     VTP00250
                                                                         VTP00260
   SELECT (AINFO.VTYPE);                                                 VTP00270
                                                                         VTP00280
      WHEN ('N') DO; /*NUMERIC DATA*/                                    VTP00290
         DATA=SUBSTR(CDATA,1,DLEN);                                      VTP00300
         I=VERIFY(DATA,'0123456789 ');                                   VTP00310
         IF I^=0 THEN DO;                                                VTP00320
            RTN_CODE=1; /*ILLEGAL NUM DATA*/                             VTP00330
            RETURN;                                                      VTP00340
            END;                                                         VTP00350
         IF DATA =(40)' ' THEN NUM=0;                                    VTP00360
         ELSE NUM=DATA; /* CONVERSION */                                 VTP00370
                                                                         VTP00380
            /*CHECK FOR LIMITS**/                                        VTP00390
            IF MODE=TO_BIT_CHK THEN DO;                                  VTP00400
               IF NUM<AINFO.MAX & NUM>AINFO.MIN THEN LEAVE;              VTP00410
               ELSE DO; /*VIOLATING LIMITS*/                             VTP00420
                 RTN_CODE=5; /****5 IS LIMIT VIOLATIONS********/         VTP00430
                 RETURN;                                                 VTP00440
                 END;                                                    VTP00450
            END;                                                         VTP00460
                                                                         VTP00470
            SUBSTR(BDATA,1,32)=UNSPEC (NUM); /*TO BIT STRING*/           VTP00480
            DLEN=4; /*4 BYTES LONG*/                                     VTP00490
      END /*WHEN 'N'*/;                                                 VTP00500
                                                                         VTP00510
      OTHERWISE  DO;                                                     VTP00520
             BDATA=UNSPEC(SUBSTR(CDATA,1,AINFO.MLEN));                   VTP00530
             DLEN = AINFO.MLEN;  END;                                    VTP00540
                                                                         VTP00550
```

```
      END /*SELECT VTYPE*/;                                          VTP00560
     END /*TYPE*/;                                                   VTP00570
   END /*TO_BIT MODE*/;                                              VTP00580
                                                                     VTP00590
 WHEN (TO_CHAR) DO;                                                  VTP00600
                                                                     VTP00610
  IF AINFO.TYPE='V' THEN DO;                                         VTP00620
    SELECT (AINFO.VTYPE);                                            VTP00630
                                                                     VTP00640
      WHEN ('N') DO;                                                 VTP00650
         UNSPEC (NUM)=SUBSTR(BDATA,1,32); /*GET DATA*/               VTP00660
         PUT STRING(CDATA) EDIT(NUM)(F(AINFO.MLEN));                 VTP00670
         DLEN=AINFO.MLEN;                                            VTP00680
         END;                                                        VTP00690
                                                                     VTP00700
      OTHERWISE UNSPEC(CDATA) = BDATA;                               VTP00710
                                                                     VTP00720
      END /*SELECT VTYPE*/;                                          VTP00730
   END /*TYPE*/;                                                     VTP00740
 END /*MODE TO_CHAR*/;                                               VTP00750
                                                                     VTP00760
END /*SLECT MODE*/;                                                  VTP00770
                                                                     VTP00780
END /*VTPE*/;                                                        VTP00790
```


DA
FIL