LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

WRDC-TR-90-8027 V01/1

DOCUMENT IDENTIFICATION

Nov 1990

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DTIC ELECTED

DATE ACCESSIONED

DTIC

DATE RETURNED

A-1

DISTRIBUTION STAMP

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC
GEOMETRIC MODELING APPLICATIONS INTERFACE PROGRAMS

GMAP/PDDI SYSTEM COMPONENT PRODUCT SPECIFICATION (AS BUILT)

United Technologies Corporation
Pratt and Whitney
Government Products Division
P.O. Box 9600
West Palm Beach, Florida 33410-9600

NOVEMBER 1990

Final Report For Period August 1985 - March 1989

Approved for public release; distribution is unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Charles R. Gilman
Project Manager

Walter H. Reimann, Chief
Computer-Integrated Mfg. Branch

FOR THE COMMANDER

Bruce A. Rasmussen
Chief, Integration Technology Division
Manufacturing Technology Directorate

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTT, WPAFB, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.
This "As-Built" Product Specification establishes the "as-built" Computer Program Configuration Item (CPCI) identified as the GMAP/PDDI System Components under U.S. Air Force Contract F33615-85-C-5122. It includes descriptions of the structure, functions, language, database requirements, interfaces, and quality assurance provisions of the primary GMAP system components: the System Translator, Model Access Software with Name/Value Interface, and the Schema Manager.
18. Subject Terms (Continued)
   Product Life Cycle
   Engineering
   Manufacturing
   Interface
   Exchange Format
   CAD
   CAM
   CIM
   IBIS
   RFC
   System Translator
   Schema Manager
   Model Access Software
   Name/Value Interface
FOREWORD

This As-built Product Specification, divided into four volumes, covers work performed under Air Force Contract F33615-85-C-5122, Geometric Modeling Applications Interface Program (GMAP), covering the period 1 August 1985 to 31 March 1989. The document addresses the GMAP/PDDI System Components developed or enhanced under this contract which is sponsored by the Computer Integrated Manufacturing Branch, Materials Laboratory, Air Force Systems Command, Wright Air Force Base, Ohio 45433-6533. The GMAP Project Manager for the Air Force is Mr. Charles Gilman.

The primary contractor is Pratt & Whitney, an operating unit of United Technologies Corporation. Mr. Richard Lopatka is managing the GMAP project at Pratt & Whitney. Ms. Linda Phillips is the Program Integrator. Mr. John Hamill is the Deputy Program Manager.

McDonnell Aircraft Company was the subcontractor responsible for the PDDI System Component work. Mr. Jerry Weiss is the GMAP Program Manager at McDonnell Aircraft and Mr. Herb Ryan is the Deputy Program Manager.

Volume I of this document provides the Scope, References, and Detail Design of the GMAP/PDDI system components.

NOTE: The number and date in the upper right corner of each page in this document indicate that it has been prepared in accordance to the ICAM CM Life Cycle Documentation requirements for a Configuration Item (CI).
# TABLE OF CONTENTS

**Volume I**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SECTION 1.</strong> SCOPE</td>
<td>1-1</td>
</tr>
<tr>
<td>1.1 Identification</td>
<td>1-1</td>
</tr>
<tr>
<td>1.2 Functional Summary</td>
<td>1-1</td>
</tr>
<tr>
<td>1.3 Approach</td>
<td>1-2</td>
</tr>
<tr>
<td><strong>SECTION 2.</strong> REFERENCES</td>
<td>2-1</td>
</tr>
<tr>
<td>2.1 Reference Documents</td>
<td>2-1</td>
</tr>
<tr>
<td>2.1.1 Military</td>
<td>2-1</td>
</tr>
<tr>
<td>2.1.2 Commercial</td>
<td>2-4</td>
</tr>
<tr>
<td>2.1.3 Standards Organizations</td>
<td>2-5</td>
</tr>
<tr>
<td>2.2 Terms and Acronyms</td>
<td>2-6</td>
</tr>
<tr>
<td>2.2.1 Glossary A -- Terms Used In GMAP</td>
<td>2-6</td>
</tr>
<tr>
<td>2.2.2 Glossary B -- Terms Used In IDEF0 Diagrams</td>
<td>2-19</td>
</tr>
<tr>
<td>2.2.3 Acronyms Used In GMAP</td>
<td>2-24</td>
</tr>
<tr>
<td><strong>SECTION 3.</strong> DETAIL DESIGN</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1 System Overview</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1.1 Physical Schemas</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1.2 Software Packages</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2 IDEF0 Function Models</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2.1 Schema Manager</td>
<td>3-2</td>
</tr>
<tr>
<td>3.2.1.1 A-0 Manage Schema Data</td>
<td>3-3</td>
</tr>
<tr>
<td>3.2.1.2 AO - Manage Schema Data</td>
<td>3-3</td>
</tr>
<tr>
<td>3.2.2 Model Access Software</td>
<td>3-7</td>
</tr>
<tr>
<td>3.2.2.1 A-0 Perform Model Access</td>
<td>3-7</td>
</tr>
<tr>
<td>3.2.2.2 AO Perform Model Access</td>
<td>3-7</td>
</tr>
<tr>
<td>3.2.3 System Translator</td>
<td>3-10</td>
</tr>
<tr>
<td>3.2.3.1 AO Exchange PDD</td>
<td>3-10</td>
</tr>
<tr>
<td>3.2.3.2 A1 Preprocess PDD</td>
<td>3-10</td>
</tr>
<tr>
<td>3.2.3.3 A12 Create Data Section</td>
<td>3-13</td>
</tr>
<tr>
<td>3.2.3.4 A2 Postprocess PDD</td>
<td>3-13</td>
</tr>
<tr>
<td>3.2.3.5 A23 Process Data Section</td>
<td>3-16</td>
</tr>
<tr>
<td>3.2.3.6 A24 Resolve Forward References</td>
<td>3-16</td>
</tr>
<tr>
<td>3.3 Application Interfaces</td>
<td>3-19</td>
</tr>
<tr>
<td>3.3.1 Interfaces Between GMAP/PDDI</td>
<td>3-19</td>
</tr>
<tr>
<td>3.3.1.1 Application Program/Working Form</td>
<td>3-19</td>
</tr>
<tr>
<td>3.3.1.2 User Interface/System Translator</td>
<td>3-20</td>
</tr>
<tr>
<td>3.3.1.3 Working Form/Exchange Format</td>
<td>3-22</td>
</tr>
<tr>
<td>3.3.1.4 Working Form/Database Management System</td>
<td>3-22</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

3.3.1.5 Exchange Format .................................. 3-22
3.3.1.6 Schema Manager .................................. 3-23
3.3.2 Interfaces Between Applications .................. 3-23
3.3.2.1 Working Form as Exchange Mechanism .......... 3-24
3.3.2.2 Working Form as the Native Form ............ 3-24
3.3.2.3 Exchange Format and Working Form .......... 3-24
3.3.2.4 Transfer Using the Working Form, Exchange Format, and Direct Translator to the Exchange Format .......... 3-28
3.4 Program Interrupts ................................ 3-29
3.4.1 Schema Manager .................................. 3-29
3.4.2 Model Access Software with Name Value Interface ................................ 3-31
3.5 Timing and Sequencing Description ................. 3-33
3.6 Data Dictionary .................................. 3-34
3.6.1 Schema Manager .................................. 3-34
3.6.2 Model Access Software Data Dictionary ........ 3-34
3.6.3 Name Value Interface Data Dictionary ............ 3-64
3.6.4 System Translator Data Dictionary ............... 3-104
3.7 Object Code Creation .............................. 3-107
3.7.1 Schema Manager .................................. 3-107
3.7.2 Model Access Software ........................... 3-107
3.7.3 System Translator ............................... 3-107
3.7.3.1 Use Portable Higher Order Language ........ 3-107
3.7.3.2 Use Model Access Software ................. 3-107
3.7.3.3 Interface to Exchange Format .............. 3-108
3.7.3.4 Interface to Native System .................. 3-108
3.8 Adaptation Data .................................. 3-108
3.9 Detail Design Description ........................ 3-108
3.9.1 Schema Manager Hierarchy ....................... 3-108
3.9.2 Model Access Software Hierarchy ............... 3-108
3.9.3 Name/Value Interface Hierarchy ............... 3-108
3.9.4 System Translator ............................... 3-108
3.9.4.1 Postprocessor ............................... 3-108
3.9.4.2 Preprocessor ............................... 3-170
## TABLE OF CONTENTS (Continued)

### Volume II

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10</td>
<td>Routine Listings</td>
<td>3-171</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Schema Manager</td>
<td>3-171</td>
</tr>
<tr>
<td>3.10.1.1</td>
<td>Index</td>
<td>3-171</td>
</tr>
<tr>
<td>3.10.1.2</td>
<td>Listings</td>
<td>3-178</td>
</tr>
</tbody>
</table>

### Volume III

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10.2</td>
<td>Model Access Software</td>
<td>3-625</td>
</tr>
<tr>
<td>3.10.2.1</td>
<td>Index</td>
<td>3-625</td>
</tr>
<tr>
<td>3.10.2.2</td>
<td>Listings</td>
<td>3-631</td>
</tr>
<tr>
<td>3.10.3</td>
<td>N/VI</td>
<td>3-913</td>
</tr>
<tr>
<td>3.10.3.1</td>
<td>Index</td>
<td>3-913</td>
</tr>
<tr>
<td>3.10.3.2</td>
<td>Listings</td>
<td>3-914</td>
</tr>
</tbody>
</table>

### Volume IV

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10.4</td>
<td>System Translator</td>
<td>3-979</td>
</tr>
<tr>
<td>3.10.4.1</td>
<td>Index</td>
<td>3-979</td>
</tr>
<tr>
<td>3.10.4.2</td>
<td>Listings</td>
<td>3-980</td>
</tr>
</tbody>
</table>

### SECTION 4 QUALITY ASSURANCE

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Quality Assurance (QA) Requirements</td>
<td>4-1</td>
</tr>
</tbody>
</table>
## LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>GMAP/PDDI Architecture</td>
<td>3-2</td>
</tr>
<tr>
<td>3-2</td>
<td>Conceptual Schema Model</td>
<td>3-4</td>
</tr>
<tr>
<td>3-3</td>
<td>IDEF0 A-0 – Manage Schema Data</td>
<td>3-5</td>
</tr>
<tr>
<td>3-4</td>
<td>IDEF0 A0 – Manage Schema Data</td>
<td>3-6</td>
</tr>
<tr>
<td>3-5</td>
<td>IDEF0 A-0 Perform Model Access</td>
<td>3-8</td>
</tr>
<tr>
<td>3-6</td>
<td>IDEF0 A0 Perform Model Access</td>
<td>3-9</td>
</tr>
<tr>
<td>3-7</td>
<td>IDEF0 A0 – Exchange PDD</td>
<td>3-11</td>
</tr>
<tr>
<td>3-8</td>
<td>IDEF0 A1 – Preprocess PDD</td>
<td>3-12</td>
</tr>
<tr>
<td>3-9</td>
<td>IDEF0 A12 – Create Data Section</td>
<td>3-14</td>
</tr>
<tr>
<td>3-10</td>
<td>IDEF0 A2 – Postprocess PDD</td>
<td>3-15</td>
</tr>
<tr>
<td>3-11</td>
<td>IDEF0 A23 – Process Data Section</td>
<td>3-17</td>
</tr>
<tr>
<td>3-12</td>
<td>IDEF0 A24 – Resolve Forward References</td>
<td>3-18</td>
</tr>
<tr>
<td>3-13</td>
<td>Interfaces of the GMAP/PDDI Exchange System</td>
<td>3-21</td>
</tr>
<tr>
<td>3-14</td>
<td>Exchange of Dissimilar Native Representations Through the Working Form</td>
<td>3-25</td>
</tr>
<tr>
<td>3-15</td>
<td>Transfer of Similar Native Representations Based on the Working Form</td>
<td>3-26</td>
</tr>
<tr>
<td>3-16</td>
<td>Transfer of Dissimilar Native Representations Using the Working Form</td>
<td>3-27</td>
</tr>
<tr>
<td>3-17</td>
<td>Transfer of Dissimilar Native Representations Using the Working Form, Exchange Format and Direct Translator to the Exchange Format</td>
<td>3-28</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>MAS Internal Return Codes</td>
<td>3-31</td>
</tr>
</tbody>
</table>
SECTION 1

SCOPE

1.1 Identification

This Product Specification describes the "as built" Computer Program Configuration Item (CPCI) identified as the GMAP/PDDI System Components. It includes descriptions of the structure, functions, language, database requirements, interfaces and quality assurance provisions of the primary system components: the System Translator, the Model Access Software with Name Value Interface, and the Schema Manager. This CPCI is part of the Geometric Modeling Applications Interface Program (GMAP), Project 5602.

The Product Definition Data Interface Program, Project 5601, Contract F33615-82-C-5036, provided the foundation for GMAP. The goal of the Product Definition Data Interface Program was to develop computer based communications to replace the traditional engineering drawings used between engineering and manufacturing environments. GMAP refines and extends this product definition data to include data needed to support applications throughout the entire product life cycle. The overall objectives of GMAP are to identify, establish, and demonstrate the use of computerized product definition data in the engineering, manufacturing, and logistics support of complex structural components.

1.2 Functional Summary

This Product Specification is a reference document for programming personnel who maintain and enhance GMAP/PDDI software. It provides decomposition, or detailing, of the system design. The primary GMAP/PDDI System Components are the System Translator, the Model Access Software with Name Value Interface, and the Schema Manager.

The System Translator is a software package that is used to transmit Product Definition Data between dissimilar CAD/CAM systems. The System Translator's main functions are to convert the Product Definition Data from a Working Form physical structure on a sending system to an Exchange Format physical structure, or from an Exchange Format physical structure to a Working Form physical structure on a receiving system. The System Translator encodes Product Definition Data into the Exchange Format independent of the sending application. The sending system needs to translate its Product Definition Data into the GMAP Conceptual Schema and create a Working Form of the Product Definition Data to be sent. Similarly, the receiving system needs to translate the GMAP Conceptual Schema Product Definition Data into its native representation. The System Translator provides exchange file conversion for the sending and receiving systems.
The Model Access Software provides the utility functions through which the PDDI System Translator can convert the exchange format data file into a random access Working Form for the target CAD/CAM system. Once in this Working Form, the data can be accessed or manipulated by the application programs through subroutine calls to the Model Access Software.

As part of the Model Access Software, the Name/Value Interface frees applications programmers from the need to be concerned with the physical location of attribute values for entities within the Working Form. The applications programmers need only know the attribute name and data type, from the physical schema definition, to use the attribute value. The benefits derived from this approach include: the ability to alter the physical data structure without impact on the source code of the applications programs, removal from the applications programmers of the requirement to program and maintain attribute data structures and access algorithms, and concentration of efficiency concerns at the system level.

The Schema Manager is a software tool that will be used to manage all aspects of the creation and interrogation of the Conceptual Schema. It will create an "active" information model of the Conceptual Schema. This information model becomes the master representation of the Conceptual Schema, and will be used to generate a physical schema. The physical schema will be expressed in forms suitable for compile-time and run-time binding.

1.3 Approach

This Product Specification consists of the four volumes. Volume I contains:

- Section 1 - An introduction, describing the function of the primary GMAP/PDDI system components at a high-level, and summarizing this document.
- Section 2 - Lists reference documents and terms and acronyms used in this document.
- Sections 3 through 3.9 - Provides a System Overview, IDEF0 Function models, application interfaces, program interrupts and other design details.

Volume II contains:

- Section 3.10.1 - the routine listings for the Schema Manager.
Volume III contains:

- Sections 3.10.2 and 3.10.3 - the routine listings for the Model Access Software and Name/Value Interface.

Volume IV contains:

- Sections 3.10.4 and 4 - the routine listings for the System Translator and Quality Assurance provisions.
SECTION 2
REFERENCES

2.1 Reference Documents

The following technical reports, specifications, standards, and other documents have been referred to or are relevant to this As-built System Components Product Specification.

2.1.1 Military:


Interim Technical Report No. 4 (ITR560240004U)
"Geometric Modeling Applications Interface Program" November 1986


Interim Technical Report No. 5 (ITR560240005U)
"Geometric Modeling Applications Interface Program" January 1987
(Period 1 August 1986 - 31 October 1986).


Interim Technical Report No. 6 (ITR560240006U)
"Geometric Modeling Applications Interface Program" May 1987
(Period 1 November 1986 - 31 January 1987).

Geometric Modeling Applications Interface Program (GMAP) System Specification (Volumes I-IV), CI SS560240001U, July 1987

Interim Technical Report No. 7 (ITR560240007U)
"Geometric Modeling Applications Interface Program," August 1987
(Period 1 February 1987 - 30 April 1987).


Geometric Modeling Applications Interface Program (GMAP) to Retirement for Cause Interface Unit Test Plan, CI UTP560240011U, December 1987.


Geometric Modeling Applications Interface Program (GMAP) to Retirement for Cause Interface Unit Test Report, CI UTR560240011U, November 1988.


2-3


Demonstration Model Descriptions for Geometric Modeling Applications Interface Program (GMAP), CI TTD560240001U, July 1989.


2.1.2 Commercial


2.1.3 Standards Organizations

ANSI Y14.5M, Dimensioning and Tolerancing.


2-5
2.2 Terms and Acronyms

Glossary A defines terms frequently used in GMAP that may be included in this Product Specification. Some reference notes applicable to these definitions are presented after the glossary. A second glossary, defining terms used in the IDEF0 diagrams found in Section 3.0, is presented as Glossary B. A list of acronyms and abbreviations used in GMAP is also included in this section.

2.2.1 Glossary A -- Terms Used in GMAP

Accept/Reject/Incomplete Notice -- A display on the cell computer that indicates the final status of the engine disk.

Accept = Acceptable within tolerance specified by engine manufacturer
Reject = Rejected because of flaw(s) outside the range of acceptable tolerances
Incomplete = Part cannot be inspected

Access Software -- A set of routines for creating, managing and querying an incore Working Form model.

Angular -- An angular size tolerance is used to tolerance the size of an angular feature independent of its angular location along an arc.

Application -- A method of producing a specific result.

Application Request -- A request initiated by an application program, either through batch or interactive processing, which will interrogate the model through the PDDI Access Software to obtain or operate on specific information regarding the model and its components or elements.

Application Requested Data -- The data which fulfills the application's original request and which is in the proper format and readable by the application.

Architecture -- A design or orderly arrangement.

ASCII -- American Standard Code for Information Interchange.
As-Is -- The present condition.

Attribute -- A quality or characteristics element of any entity having a name and a value.

B-Spline -- A spline defined by a control polygon, B-spline basis functions, and an associated knot vector. A Bezier curve is a special case of a B-spline; a nurbs is the most general case of a B-spline.

Bezier Curve -- A type of curve defined by a set of vertices called a control polygon and a set of basis functions. The basis functions are known as Bernstein polynomials. K vertices define a curve of order K-1.

Binding -- Establishing specific physical references to data structures for an application program; may be performed at compile time or at run time.

Blend -- A smooth, continuous transition from one surface to another.

Boundary Representation -- A topology imposed on 3-D geometric entities to yield a general solid model. That model describes an object by describing its boundary area.

Body of Revolution (BOR) Representation -- A topology in which an object is represented as the volume swept by a curve rotated about a line. This is a boundary representation in which the curve represents the surface area of the object.

Bounded Geometry -- Geometry that has limits defined by its mathematical domain or range.

Calibration Block Parameters (Scale Factors) -- Nondestructive test parameters used to adjust a specific cell. These parameters are obtained from the calibration blocks located at each cell.

Circumferential -- A circumferential tolerance specifies the tolerance zone within which the average diameter of a circular feature must lie. The average diameter is the actual circumference divided by pi (3.14159). A circumferential tolerance is a specific example of a peripheral or perimeter tolerance for a general curve.

Class -- A collection of entities that are alike in some manner.

CLIST -- IBM Command lists.

Composite Curve -- A group of curve segments that are C0 continuous.
Compound Feature Representation -- An enumerative feature representation in which at least one component is itself a feature. For example, a bolt hole circle might be represented as a list of individual hole features.

Concentricity (Generic) -- A concentricity tolerance specifies a cylindrical tolerance zone within which the axis of a feature must lie, where the axis of the zone coincides with the axis of the datum.

Conceptual Schema -- Formally specified global view that is processing independent, covering information requirements and formulation of independent information structures. A neutral view of data, usually represented in terms of entities and relations.

Conic -- A quadratic curve represented in the most general case by the equation:

\[ Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \]

A conic may be a circle, line, ellipse, parabola, or a hyperbola depending on the coefficients, A, B, C, D, E, and F.

Constraints (Generic) -- An assertion to explicitly specify data meaning or semantics.¹ (Notes appear at the end of this section.)

Context-Free Grammar -- The syntax of the language gives a precise specification of the data without interpretation of it.

Constituent -- A specific instance of an entity that is used in the definition of some other entity.

Data Dictionary -- A catalog of all data elements in a design, giving their name, definition, format, source, and usage. May also include data types and value limits.

Defining Airfoil Sections -- A planar or conical section that depicts an airfoil profile. Defining airfoil sections are those that meet aerodynamic requirements. Other intermediate sections are added for Manufacturing purposes.

Dimension -- A part dimension is a quantifiable value expressing size, form, or location.

Domain -- The set of values permissible in a given context.

Dynamic Allocation -- The allocation (and de-allocation) of memory resources as required by the application. The opposite is static allocation where a fixed size segment of memory is available to the application.
Eddy Current Cell -- Hardware used to perform an Eddy current inspection operation (surface flaws).

Eddy Current Inspection -- An inspection method used to detect internal potential flaws on a disk. It is based on the principle of sending electromagnetic signals to a target area on a part and detecting/interpreting reflection (Eddy current) from the target.

Eddy Current Scan Plan -- An interpreter code program controlling the Eddy current inspection of a particular geometry.

Eddy Current/Ultrasonic Flaw Data Printout -- A printout containing size and location information about specific flaw(s) (both critical and noncritical) associated with a particular part.

Entity -- A description of a person, place, or thing, about which information is kept.

External Reference -- A reference to some quantity of data that exists somewhere outside the scope of the immediate body of information.

Feature -- A part feature in the dimensioning and tolerancing context is a feature in the sense of ANSI Y14.5M, that is, a physical component portion of a part, such as a surface, hole, slot, and so on that is used in a tolerancing situation. In the dimensioning and tolerancing context, a feature consists of individual or groups of basic shape elements used to define the physical shape of an item. This general dimensioning and tolerancing use of features is to be distinguished from Features. The word "features" alone implies dimensioning and tolerancing features. The term "form feature" is described below.

Feature Pattern -- A geometric pattern of occurrences of similar form features, for example, a circular pattern of scallops, a rectangular array of holes.

Feature Representation (Generic) -- A description of a form feature within the context of a geometric model.

Feature Type -- A name applied to a form feature that is suggestive of its shape and size, for example, hole, slot, web.

Feature of Size (Generic) -- A feature of size provides a geometric location capable of being referenced for use with datums and tolerances. A feature of size can be a GMAP feature, or other referenceable shape elements of a part model that are symmetric about a point, line, plane, axis, curve, and so on. When a feature of size is used in a relationship with a tolerance or datum, its feature of symmetry is the implied reference.
Flat Pattern Representation (Extrusion Representation) -- A topology in which an object is represented as the volume swept by a planar polygon moving in a direction normal to its plane. The polygon may have internal polygon represent the surface area of the object.

Flaw Characteristics -- Location, length, width, depth, and nondestructive test parameters associated with a specific flaw.

Flaw Data Packet -- Packet containing nonevaluated flaw data. Note that the packet can contain zero flaws.

Flaw Orientation -- The direction of the major characteristic of the flaw with respect of the part coordinate system. (See the notes section at the end of this glossary.)

Flaw Suspect Location -- The coordinate location of a possible flaw detected during a survey mode inspection (six-axis position of ultrasonic cell, seven-axis position of Eddy current cell).

Form Feature -- A portion of a part's geometry that is useful to regard as an entity. In a boundary representation context, this is a subset of the part's surface area.

Form Tolerance -- Form tolerances are used to control the form of model features. A form tolerance specifies the amount that an actual features form may vary from nominal. Form tolerance include straightness tolerance, flatness tolerance, roundness/circularity tolerance, cylindricity tolerance, perpendicularity tolerance, parallelism tolerance, angularity tolerance, profile-of-a-line tolerance, profile-of-a-surface tolerance, circular-runout tolerance, true-direction tolerance, and mismatch tolerance.

Functionality -- (1) To show that the configuration item has fulfilled the specified requirements. (2) The receiving and sending systems can operate on the entity in the same manner with the same results within a pre-defined tolerance.

Function Modeling -- A description of a system in terms of a hierarchy of functions or activities, each level decomposing higher ones into greater detail. Functions are named by verbs; nouns related are declared as inputs, controls, outputs, and mechanisms.

Geometric Element (Generic) -- An instance of a geometric entity.

Geometric Group -- A group of geometric entities with a name.

Geometric Model -- A part description in terms of its underlying geometric elements. The model may be a wireframe, surface, or solid model.
Geometric Pattern -- A circular or rectangular pattern of geometric entities.

Group Technology Code -- An alphanumeric string identifying significant characteristics of a product, enabling group technology applications. Also known as Part Classification Code.

Include File -- PASCAL source code from another file or library included on the compilation of a PASCAL source file.

Input Data -- That information which the application needs to supply in order to interrogate or operate on the model. This data may assume only these forms prescribed by the PDL Access Software specification.

Inspection Cycle -- A period for which nondestructive testing inspection requirements are defined.

Inspection Cycle Zone -- An entity that is composed of a unique combination of zone and inspection cycle.

Inspection Module Operator -- Refers to personnel operating RFC cell(s).

Instrument Setting Adjustments -- Nondestructive testing parameter adjustments automatically accomplished via pre- and post-calibration operations. These adjustments have to be accomplished within a predetermined tolerance.

Internal Flaw -- A subsurface anomaly.

Internal Flaw Major Characteristic -- A vector determined by an agreed upon method.

Example (1): The vector of greatest magnitude from the centroid to a boundary of the anomaly.

Example (2): A vector representing the major axis of the minimum ellipsoidal envelope encompassing the anomaly.

Internal Flaw Tolerance -- A unique combination of:

(a) Internal flaw orientation range.
(b) Serviceable internal flaw tolerance limits.
(c) Repairable internal flaw tolerance limits.
Internal Flaw Tolerance Limit -- A unique combination of:

7(a) Maximum diameter.
(b) Maximum depth below surface.
(c) Maximum thickness.

Interpreted Request -- Input data which has been appropriately modified to conform to the PDDI Access Software's internal data representation so that it may be further processed.

Key Attribute -- An attribute or combination of attributes having values that uniquely identify each entity instance.

Laminates Representation (Generic) -- A topology in which an object is represented as layers of flat material of known thickness.

Location Tolerance -- Location tolerances specify the allowable variation in position of model features. Location tolerances include various forms of position tolerancing conventions. These are (true) position, concentricity, alignment, rectilinear location, and angular location.

Logistics Support -- The function of procuring, distributing, maintaining, replacing, and repairing material in support of a delivered product.

Machine Coordinate Positions -- The probe location with respect to machine coordinates.

Machine Preset Data -- Machine coordinate adjustments automatically accomplished via pre- and post-calibration operations. These adjustments have to be accomplished within predetermined tolerance.

Metadata -- Data about data. Defines the physical schema and record formats of the part data.

Metamodel -- A body of data that defines the characteristics of a data model or structure.

Model -- A collection of PDD that is transferable, displayable, accessible, and equivalent to a Part. The internal representation of the application data, as initiated and organized by the user. The model is also referred to as the Working Form.

Model Network Definition -- The set of rules and definitions which outline in detail the data structure whereby higher order entities may be composed of lower order entities, or constituents, and the lower order entities may be constituents of one or more higher order entities.
Native System -- The PDD and applications in a format that is unique to the database of a CAD system.

Nondestructive Testing Parameters -- Parameters used by the Eddy current and ultrasonic instruments (examples: amplitude, phase angle, gain, threshold, and so on).

Nondestructive Testing Personnel -- Personnel responsible for the generation of scan plans and derivation of applicable nondestructive testing instrument settings used in the scan plans.

Nonshape Data -- Produce definition data that cannot be represented by shape elements.

Normal Forms -- Conditions reflecting the degree of refinement and control over the relationships and entities in an information model.

Numerical Control Program (Complete and Proposed) -- Set of program instructions used to generate a probe path.

Orientation Range -- An envelope in which the major flaw characteristic must lie.

Parse -- The process of analyzing input strings (records) to identify fields and to verify that the data has a valid format.

Part Blueprint -- A blueprint provided by the engine manufacturer of a particular F100 engine disk.

Physical Schema -- Internal representation of data; the computer view that includes stored record format and physical ordering of stored records.

PID File -- A PID File is a copy of the Working Form filed to disk for temporary storage. The software that produces this capability (PID Code) is provided as an interim solution while a translator to the native database is in development.

Polynomial Spline -- A parametric spline of order 1, 2, or 3 defined by a set of N+1 points. The spline is CX, CY, or CZ continuous and defined by coefficients such that:

2-13
\[ x(i) = AX(i) + BX(i) \ast S + CX(i) \ast S^2 + DX(i) \ast S^3 \]
\[ y(i) = AY(i) + BY(i) \ast S + CY(i) \ast S^2 + DY(i) \ast S^3 \]
\[ z(i) = AZ(i) + BZ(i) \ast S + CZ(i) \ast S^2 + DZ(i) \ast S^3 \]

and a parameter space \((T_0, T_1, \ldots, T_n)\)

where

\[ T(i) < u < T(i+1) \]
\[ S = u - T(i) \]

Position Tolerance — A position tolerance (true position) specifies a tolerance zone within which the feature may vary in any direction.

Post-processor — A phase of the translator where data is received from the Exchange Format and is converted to the Working Form.

Pre-processor — A phase of the translator where data is taken from the Working Form and is converted to the Exchange Format.

Primitive Constructive Feature Representation (Generic) — A constructive representation that is noncompound and that does not incorporate another feature. Such a representation must consist solely of overt construction information. Representation of a through hole by centerline and diameter is an example.

Probe Blueprint — Blueprint of Eddy current probe supplied by the probe manufacturer.

Product Definition Data — Those data "explicitly representing all required concepts, attributes, and relationships" normally communicated from Design throughout Manufacturing and Logistics Support. The data include both shape and nonshape information required to fully represent a component or assembly so that it can be analyzed, manufactured, inspected, and supported. They enable downstream applications, but do not include process instructions. These data are not always finalized at the design release; the manufacturing process can also add to the product model or generate derived manufacturing product models.

Product Life Cycle — Includes design, analysis, manufacturing, inspection, and product and logistics support of a product.
Product Model -- A computer representation of a product.

Product Support -- The function that interprets customer requests for information and can provide the technical responses to the customer in the form of technical orders and instructions.

Proprietary Part Flaw Data -- Formatted dataset containing proprietary data defining size(s), maximums, and location(s) of critical flaw(s) (dimensional and locational tolerance).

RAW.O File -- A data file that uses a bi-cubic patch surface representation to define the surfaces of an airfoil.

Ready Status -- Go/No-Go decision.

Relation -- A logical association between entities.\(^3\)

Remount Decision -- Decision to remount an engine disk.

Replicate Feature Representation (Generic) -- A description of a feature as being identical to another feature except for location. Mathematically, a replicate feature representation consists of the identification of another (necessarily constructive) feature plus a transformation.

Robot Initialization Parameters -- A set of nondestructive testing parameters used to initialize the robot on an Eddy current or ultrasonic cell.

Rotational Sweep -- A sweep in which the swept curve is rotated about a line (the "centerline" of the sweep).

Ruled Surface (Generic) -- A surface defined by a linear blend of two curves.

Run System -- The Translator subpackage which provides the communication interface between the user and the pre/Post-processors.

Run-Time Subschema -- A subset of the data dictionary information used at run-time by the access software to provide field data and check data.

Scan Plan -- Instructions that drive an inspection; these include inspection area geometry, ordered inspection path points, inspection probe selection, inspection path for each probe, mechanical commands that allow mechanical manipulator positioning, instrument setting, and all the variables needed for signal processing and flaw data acquisition during inspection.
Scan Plan Specifications -- Standards and procedures used in creating Eddy current and ultrasonic scan plans for the RFC system.


Shape -- The physical geometry of a mechanical part, as distinguished from a computer description of that geometry. Where the difference is significant, the attitude is taken that shape is nominal or basic, with shape variations of tolerances grafted thereon.

Shape Data -- Include the geometric, topological description of a product along with the associated dimensional tolerances and feature descriptions.

Single Spatial Probe/Transducer Path -- The starting and ending location of a single probe movement.

Size Tolerance -- Size tolerances specify the allowable variation in size-of-model features, independent of location. Size tolerances include circumferential, rectilinear size, and angular size.

Solid Geometric Model (Shape Representation) -- A computer description of shape. The description may be partial in the sense that not all aspects of part shape are indicated. For example, a body of revolution representation of a turned part may not describe the nonaxisymmetric aspects of part geometry. A solid model must be complete and unambiguous in the sense that it describes a single volume in 3-D space.

Solid Modeling -- The creation of an unambiguous and complete representation of the size and shape of an object.

Source Code -- A computer program written in some language which is processed to produce machine code.

Spline -- A piecewise polynomial of order K, having continuity up to order K-1 at the segment joints.

Squirter Blueprint -- Blueprint of the squirter head that houses the ultrasonic transducer.

Subface -- A subface is a bounded portion of a face. It is defined by an underlying face, exactly one periphery closed curve and zero, one, or more internal closed curves that represent cutouts or holes in the region. The internal closed curve must not touch or intersect each other or the periphery closed curve and must be entirely contained within the periphery closed curve.
Surface Flaw -- A surface anomaly.

Surface Flaw Major Characteristic -- A vector determined by an agreed upon method.

Example: A vector representing the major axis of the minimum elliptical envelope encompassing the anomaly in the plane of the surface.

Surface Flaw Tolerance -- A unique combination of:

(a) Surface flaw orientation range.
(b) Serviceable surface flaw tolerance limits.
(c) Repairable surface flaw tolerance limits.

Surface Flaw Tolerance Limit -- A unique combination of:

(a) Maximum length.
(b) Maximum width.
(c) Maximum depth.

Sweep Surface -- Surfaces formed by extruding or revolving a planar profile in space.

Syntax -- Grammar: A set of rules for forming meaningful phrases and sentences from words in a vocabulary.

System Computer -- VAX 11/780 and supporting peripheral hardware.

System Constraints -- Those hardware and software environmental constraints which will be imposed upon the PDDI Access Software that will limit its implementation and application. An example of such constraints might be the particular compiler used to compile the PDDI Access Software package.

To-Be -- The future condition possible, given a proposed capability.

Tolerance (Generic) -- The total amount by which something may vary. For mechanical product definition, tolerances can be shape tolerances, weight tolerances, finish tolerances and so on. In the context of GMAP, the term "tolerance" used alone implies shape tolerance. Other forms of tolerance (nonshape) are explicitly stated, for example, "finish tolerance." In a GMAP product model, tolerances occur without dimensions. As in the Product Definition Data Interface Program, model dimensions are implicit in the model geometry. Therefore, application of a tolerance implies a specific underlying dimension or geometric condition.

Topology -- A data structure that assembles geometric entities (points, curves, surfaces) into a solid geometric model.
Transducer Blueprint -- Blueprint of ultrasonic transducer supplied by the transducer manufacturer.

Transfer Data -- The data required to make an exchange of data between systems (i.e., delimiters, record counts, record length, entity counts, numeric precision).

Translator -- A software MECHANISM that is used for passing data between the Exchange Format and Working Form of the PDD.

Ultrasonic Cell -- Hardware used to perform ultrasonic inspection operation (internal flaws).

Ultrasonic Inspection -- An inspection method used to detect surface flaws on a disk. It uses ultrasonic waves through a stream of water to send and collect signals concerning an area targeted for inspection.

Ultrasonic Scan Plan -- Interpreter code program controlling the ultrasonic inspection of a particular geometry.

Unbounded Geometry -- Geometry represented parametrically, without limits, usually by coefficients to a defining equation.

Unigraphics (UG) -- A computer graphics system.

User Function (UFUNC) -- An interface to the UG database.

Working Form -- Product definition data information in machine-dependent data formats; an a memory resident network model.

Zone -- A physical area of the disk composed of zone components.

Zone Component -- A subface, face, or feature that constitutes a zone or element of a zone.

NOTES:


3 Ibid., p. 214.

4 Ibid., p. 211.
2.2.2 Glossary B -- Terms Used In IDEF0 Diagrams

Access Software -- The PDDI MAS.

Allocation Commands -- The commands that will allocate the necessary resources for the translator, including the name of the file.

Application Request -- A request initiated by an application program, either through batch or interactive processing, that will interrogate the model through the PDDI MAS to obtain, or operate on, specific information regarding the model and its components or elements.

Application Requested Data -- That data that now fulfills the application's original request, and which is in the proper format readable by the application.

Class -- A grouping of entities in the schema model for convenience in referring to entities with like characteristics.

Commands -- Instructions and information that require some action by the pre/postprocessors. For example, a command could tell the preprocessor to include the time and date of the translation along with the PDD being sent to the receiving system. In this example, both the date and the time data are included in the command.

The commands that control the "Step Through Processor" function are commands from the computer user. That function produces commands which provide information to the postprocessor, the preprocessor, and set run-time option.

Computer User -- The operator, or database administrator, who will send or receive Exchange Format models.

Conceptual Schema Report -- A human readable listing of the Conceptual Schema model in a format compatible with the EXPRESS information modeling language being developed in conjunction with the STEP data exchange standard.

Data Type -- The form of data within a computer. The basic data types include integer, real, string, logical, enumeration, and pointer. These may also appear in an array of one or more dimensions. A defined data type may be created from a basic data type.

Database Utilities -- Software that is used to locate files within the database and to initialize the database management software.

Encoded Exchange Format Information -- A buffer containing the next indivisible list of characters to be written in the Exchange Format.
Entity Definitions -- The identifier and attribute definitions for an entity. The identifier consists of a name (character string) and kind (integer) both of which must be unique within the schema. Each attribute definition consists of a name (character string), which must be unique within the entity, and a data type.

Entity Kind -- A field that provides the entity type.

Entity Model -- A Working Form representation of an entity definition.

Error Recovery Method -- The method chosen to recover from errors.


Exchange Medium -- The device (e.g., tape, telecommunications, disk), which transports the PDD from the sending system to the receiving system.

Feedback -- The system state and other information that is returned after a software module has been executed. The information may include such items as: the number of entities processed; any errors that affect the translation; and notification that there is no more PDD to translate.

Fetch Next Character Command -- The command to scan for the next character in the Exchange Format file.

Fetch Next Token Command -- The call to the lexical analyzer to fetch the next logical token from the Exchange Format file.

I/O System -- The system that performs the input and output between memory and external storage.

Input Data -- That information which the application needs to supply to interrogate or to operate on the model. This data may assume only those forms prescribed by the PDDI Model Access Software specifications.

Interpreted Request -- Input data that has been appropriately modified to conform to the PDDI Model Access Software's internal data representation standards so that it may be further processed.

Kind Table -- The memory-resident vector of the kind numbers and entity names created from the Kind Table Data.

Kind Table Data -- An external file which contains a vector of kind numbers and entity names. The file is ordered such that all constituents of a given entity fall before it in the vector.
Lexical Token -- The next logically indivisible string of characters that is extracted from the Exchange Format file.

Model -- The internal representation of the application data as initiated and organized by the user. The model is also referred to as the Working Form.

Model Network Definition -- The set of rules and definitions that outline the data structure, where higher order entities may be composed of lower order entities, or constituents, and the lower order entities may be constituents of one or more higher order entities.

Need to use PDD -- What is controlling the transfer of PDD from one computer system to another. It defines which portion, when, where, and how to transfer computer models.

New Statistics -- Information about each translation that is used to track the results of several translations.

Next Char -- The next character in the Exchange Format file. The stream of characters is terminated by a delimiter.

Option Specification -- A selection from the functional options of the Schema Manager software, such as: create a schema model in the Working Form; generate reports about the Conceptual Schema or the physical schema produced from it; save a schema model into a file; retrieve a schema model from a file for review or update.

Options -- The computer-recognizable options that detail the user response for the translator.

PDD -- The Product Definition Data that is to be sent from one system to another. When sending the PDD, the PDD is in an internal form. When receiving the PDD, the PDD is in an exchangeable (tape to telecommunications) form.

PDDI Concepts -- The system architecture established by the PDDI project.

Pascal Include File -- A file of constants and type declarations that can be included in a PASCAL source program to allow compile-time binding to the entity definitions.

Physical Schema -- The physical schema of the entities in the Working Form Model.

Physical Schema Table -- A data structure holding the physical schema.
Postprocessing Environment -- The initialized state that is used by the rest of the post processor.

Prepare PDD -- The required PDD is generated and put into whatever form is necessary for its use.

Receiving System -- The computer system (hardware/software) that requests the PDD.

Request for PDD -- An inquiry to initiate the preparation of the desired PDD. The inquiry may take any form necessary for an engineering or manufacturing application.

Requested Data -- Data that have been internally processed and represent the application's requested data, but remains in an internal representation form prior to reformatting to the application requested form.

Responses -- The signals and data that will generate messages for the computer user.

Results -- Information, a process, or a part that is the output of use of the PDD.

Run Time Information -- Information provided by the user for preprocessing. This includes data for the header section and comments to be inserted into the file.

Run-Time Subschema -- A computer usable form of a physical subschema produced from the Conceptual Schema, and interface routines, to allow run-time access to the entity definitions.

Schema -- Those definitions that describe the content of the data, the relationship between the various elements or components of the data, and the rules that govern how the data may be manipulated.


Sectional Context -- Which section is being parsed now: Header, Declaration, or Data.

Sending System -- The computer system (hardware/software) that prepares the PDD.

Specific Options -- The choices the computer user makes, such as the disk form to use.
**Subschema** -- A grouping of entities in the schema model for the purpose of defining the scope of a physical schema.

**System Constraints** -- The constraints that define the limitations and capabilities by which the computer system must abide. They are implied by the characteristics of the computer system. For example, the available space on the computer for holding the PDD is finite. The amount of space is then a system constraint.

**System Development Procedures** -- The procedures that guide the development of computer software and the user interface to that software.

**System Options** -- The options from which the computer user may choose. The specific options are the system options that the computer user has chosen.

**System Pack Catalog** -- The place where the name of a dataset is paired with its location and physical attributes.

**System Statistics** -- Information about the condition of the translator when a terminal error occurs.

**Translated PDD** -- Product Definition Data that arrive at the receiving system. When sending the PDD, the PDD are in an internal form. When receiving the PDD, the PDD are in an exchangeable (tape or telecommunications) form.

**Translation Procedures** -- Instructions and system options to the computer user that dictate the functions to be performed (for both the computer and the computer user).

**Translation Software** -- Software and other control language such as JCL, which are used by the translator functions to exchange PDD.

**Translation Work Order** -- A run-time order requesting PDD; a command which physically initiates a translation.

**Unvalidated Lexical Token** -- The lexical token before being checked for embeddedness in another delimiter.

**Use PDD** -- The required PDD have been made available and the application which needs the PDD uses the PDD to generate whatever results are desired.

**User Computer System** -- The specific hardware, operating systems, and applications software systems that the user will employ to implement the PDDI Model Access Software.
2.2.3 Acronyms Used In GMAP

ADB --- Application Data Block (also referred to as Attribute Data Block).
AIMS -- Automated IDEF Methodology System.
ANSI -- American National Standards Institute.
ANT -- Abstract of New Technology.
APT -- Automatically Programmed Tools.
ATP -- Automation Technology Products.
BOM -- Bill of Materials.
BOR -- Body of Revolution.
BPI -- Bits per Inch.
BREP -- Boundary Representation.
CAD -- Computer Aided Design.
CAE -- Computer Aided Engineering.
CAEDS -- Computer Aided Engineering Design System.
CALS -- Computer Aided Acquisition and Logistics Support.
CAM -- Computer Aided Manufacturing.
CAM-I -- Computer Aided Manufacturing--International.
CAS -- Cooled Airfoil System.
CDM -- Common Data Model.
CDR -- Critical Design Review.
CDT -- Component Design Technology.
CFSR -- Contract Fund Status Report.
CI -- Configuration Item.
CIM -- Computer Integrated Manufacturing.
CLIST -- IBM command list.
CM -- Configuration Management.
CMM -- Coordinate Measuring Machine.
CWBS -- Contract Work Breakdown Structure.
DBMS -- Data Base Management System.
DCL  --  DEC Command Language.
DDL  --  Data Definition Language.
DEA  --  Digital Equipment Automation.
DEC  --  Digital Equipment Corporation.
DESØ  --  (ICAM) Architecture of Design.
DJR  --  Design Job Request; Drafting Job Request.
DoD  --  Department of Defense.
DS  --  Design Specification.
DSM  --  Design Substantiation Memo.
EBCDIC  --  Extended Binary Coded Decimal Interchange Code (IBM character set).
EC  --  Eddy Current.
ECO  --  Engineering Change Order.
EDM  --  Electrical Discharge Machining.
EF  --  Exchange Format.
EII  --  Engineering Information Index.
EMD  --  Engineering Master Drawing.
EPCS  --  Engine Product Configuration Support.
ESA  --  Engineering Source Approval.
ESP  --  Experimental Solids Proposal.
FEDD  --  For Early Domestic Dissemination.
FEM  --  Finite-Element Modeling.
"OF  --  Factory of the Future.
FOS  --  Feature of Size.
FPIM  --  Fluorescent Penetrant Inspection Module.
GE  --  General Electric.
GMAP  --  Geometric Modeling Applications Interface Program.
GSE  --  Ground Support Equipment.
HCF  --  High-Cycle Fatigue.
IBIS  --  Integrated Blade Inspection System.
IBM  --  International Business Machines.
ICAM -- Integrated Computer Aided Manufacturing.
ICOM -- Input/Control/Output/Mechanism.
ICS -- Information Computer System.
IDEF -- ICAM Definition.
IDEF0 -- IDEF Function Modeling.
IDEF1 -- IDEF Information Modeling.
IDEFIX -- IDEF Extended Information Modeling.
IDEF2 -- IDEF Dynamics Modeling.
IDSS -- Integrated Decision Support System.
IEEE -- Institute of Electrical and Electronics Engineers.
IEN -- Internal Engineering Notice.
IFS -- Interface Specification.
IISS -- Integrated Information Support System.
ILC -- Improved Life Core.
IMS -- Information Management System.
IPGS -- (IBIS) Inspection Plan Generation System.
IRB -- Industry Review Board.
IPIM -- Infrared Inspection Module.
ISO -- International Standards Organization.
ITA -- Intelligent Task Automation.
ITI -- International TechnneGroup Incorporated.
LCF -- Low-Cycle Fatigue.
MAS -- Model Access Software.
MFGØ -- (ICAM) Architecture of Manufacturing.
MRP -- Materials Requirements Planning.
NAD -- Needs Analysis Document.
NBS -- National Bureau of Standards.
N/C -- Numerical Control.
NDE -- Nondestructive Evaluation.
<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDML</td>
<td>Neutral Data Manipulation Language.</td>
</tr>
<tr>
<td>NDT</td>
<td>Nondestructive Test.</td>
</tr>
<tr>
<td>NTSB</td>
<td>National Transportation Safety Board.</td>
</tr>
<tr>
<td>NVI</td>
<td>Name/Value Interface.</td>
</tr>
<tr>
<td>OGP</td>
<td>Optical Gaging Products, Inc.</td>
</tr>
<tr>
<td>PD</td>
<td>Product Data.</td>
</tr>
<tr>
<td>PDD</td>
<td>Product Definition Data.</td>
</tr>
<tr>
<td>PDDI</td>
<td>Product Definition Data Interface Program.</td>
</tr>
<tr>
<td>PDES</td>
<td>Product Data Exchange Specification.</td>
</tr>
<tr>
<td>PDL</td>
<td>Program Design Language.</td>
</tr>
<tr>
<td>PED</td>
<td>Preliminary Engine Design.</td>
</tr>
<tr>
<td>PI</td>
<td>Principal Investigator.</td>
</tr>
<tr>
<td>PID</td>
<td>PDDI Interim Database.</td>
</tr>
<tr>
<td>PIES</td>
<td>Product Information Exchange System.</td>
</tr>
<tr>
<td>PMP/PMS</td>
<td>Program Management Plan/Project Master Schedule.</td>
</tr>
<tr>
<td>PROCAP</td>
<td>Process Capability.</td>
</tr>
<tr>
<td>PS</td>
<td>Product Specification.</td>
</tr>
<tr>
<td>RFC</td>
<td>Retirement for Cause.</td>
</tr>
<tr>
<td>RPM</td>
<td>Revolutions per Minute.</td>
</tr>
<tr>
<td>SA-ALC</td>
<td>San Antonio-Air Logistics Center.</td>
</tr>
<tr>
<td>SD</td>
<td>Scoping Document.</td>
</tr>
<tr>
<td>SDL</td>
<td>Source Data List.</td>
</tr>
<tr>
<td>SDS</td>
<td>System Design Specification.</td>
</tr>
<tr>
<td>SL</td>
<td>Salvage Layout.</td>
</tr>
<tr>
<td>SML</td>
<td>Source Material Log.</td>
</tr>
<tr>
<td>SOA</td>
<td>State-of-the-Art (Survey).</td>
</tr>
<tr>
<td>SOR</td>
<td>Surface of Revolution.</td>
</tr>
<tr>
<td>SPC</td>
<td>Statistical Process Control.</td>
</tr>
<tr>
<td>SPF</td>
<td>System Panel Facility.</td>
</tr>
<tr>
<td>SQA</td>
<td>Software Quality Assurance.</td>
</tr>
</tbody>
</table>
SQAP  --  Software Quality Assurance Plan.
SRD   --  System Requirements Document.
SRL   --  Systems Research Laboratories.
SS    --  System Specification.
STEP  --  Standard for the Exchange of Product Model Data.
STP   --  System Test Plan.
TCTO  --  Time Compliance Technical Order.
TD    --  Technical Data.
TDCR  --  Turbine Design Cost Reduction.
TDR   --  Tool Design Request.
TechMod--  Technology Modernization.
TO    --  Technical Order.
TOP   --  Technical and Office Protocol.
TSO   --  Time-Sharing Option (IBM term).
UFUNC --  User Function.
UG    --  Unigraphics.
UGFM  --  Unigraphics File Manager.
USA   --  Unified System for Airfoils.
USAF  --  United States Air Force.
UTC   --  United Technologies Corporation.
UTP   --  Unit Test Plan.
UTR   --  Unit Test Report.
UTRC  --  United Technologies Research Center.
VAX   --  Virtual Architecture Extended.
VMS   --  Virtual Memory System.
WBS   --  Work Breakdown Structure.
WF    --  Working Form.
WPAFB --  Wright-Patterson Air Force Base.
XIM   --  X-Ray Inspection Module.
SECTION 3
DETAIL DESIGN

3.1 System Overview

The purpose of the GMAP/PDDI software system is to provide a prototype for the communication of complete production definition data (PDD) between dissimilar CAD/CAM Systems. This system will serve as the information interface between engineering and manufacturing functions. It is composed of Model Access Software (MAS) with Name/Value Interface (N/VI), Conceptual Schema, Schema Manager, Exchange Format and the System Translator. The relationship of these components is illustrated in Figure 3-1.

The MAS is a set of callable utility programs that will allow applications to manipulate and query PDD Working Form models. The N/VI frees applications programmers from the need to be concerned with the physical location of attribute values for entities within the Working Form. The Conceptual Schema is a data dictionary that defines the data needed to create a CAD/CAM model. The Schema Manager is a software tool that will be used to manage all aspects of the creation and interrogation of the Conceptual Schema. It will create an "active" information model of the Conceptual Schema. This information model becomes the master representation of the Conceptual Schema, and will be used to generate a physical schema. The Exchange Format is a neutral physical sequential format for passing data between dissimilar systems. GMAP used the PDES draft physical file format as its Exchange Format. The System Translator is the software mechanism for passing this data between the Exchange Format and the Working Form of the PDD.

3.1.1 Physical Schemas

The Working Form physical schema is determined through a data dictionary or PASCAL include files. The Exchange Format physical schema is defined by a data dictionary and the specification for the neutral file format.

3.1.2 Software Packages

The software for the system consists of three packages - MAS with N/VI, Schema Manager, and System Translator.

3.2 IDEF0 Function Models

This section includes IDEF0 function models that describe the functionality or activities of the individual GMAP/PDDI system components. IDEF0 function models are presented for the Schema Manager, the MAS (including N/VI), and the System Translator.
IDEFO Function Models consist of a diagram representing activities, and the corresponding input, controls, output, and mechanisms for a given node. A textual description of the functions at each node accompany each diagram. These are both titled with a node number, such as AO, and a function title. This titling can be used as a method to relate the text to the appropriate IDEFO diagram.

3.2.1 Schema Manager

The Schema Manager will enable the database administrator to create and maintain entity definitions in a Conceptual Schema model (illustrated in Figure 3-2), analyze the defined entities, and generate physical schema from the Conceptual Schema.

3.2.1.1 A-0 Manage Schema Data

Figure 3-3 defines the environment in which the Schema Manager will operate. The purpose of the Schema Manager is to allow the database administrator who is concerned with the entity definitions to model the Conceptual Schema and to produce a physical schema in the form of one or more run-time subschemas, and the corresponding PASCAL language declarations of constants and types in the form of include files.

3.2.1.2 A0 - Manage Schema Data

Figure 3-4 identifies the primary functionality of the Schema Manager software. Given the constraints imposed upon the system, the software will provide the ability to model a Conceptual Schema and generate physical subschemas from the Conceptual Schema.

Function 1: Select Option - This interactive interface will prompt the user for the function to be performed and control the subsequent processing.

Function 2: Model Entity - This interactive interface will prompt the user for the identifier and attribute definitions for an entity. The validity checks required by the entity definition methodology will be performed, and a Working Form representation of the entity definition will be produced.

Function 3: Model Schema - This interactive interface will prompt the user for the subfunctions to be performed and control the subsequent processing. The subfunctions include organizing entity definitions into subschemas and for classes, reviewing or updating definitions, and generating the Conceptual Schema report.
Function 4: Generate Physical Subschemas - This function processes the specified scope of the Conceptual Schema model to assign physical locations to the attributes within each entity, and to generate from the physical subschema either or both of a run-time subschema and a PASCAL include file.

Figure 3-2. Conceptual Schema Model
Figure 3-3. IDEF0 A-0 - Manage Schema Data
Figure 3-4  IDEF0 A0 - Manage Schema Data
3.2.2 Model Access Software

The MAS will enable the application to create, manage or query the Working Form model. A high level IDEF0 model of the MAS follows.

3.2.2.1 A-0 Perform Model Access

Figure 3-5 defines the environment in which the MAS will operate. The purpose of the MAS is to allow the user, through his software applications, to create and process network structured data stored in computer memory. Computer software will be specifically designed and implemented to provide this software functionality.

3.2.2.2 AO Perform Model Access

The purpose of Figure 3-6 is to identify the primary functionality of the MAS. Given the constraints imposed upon the system, the software will be designed to fulfill the application's request to access and process the model data.

Function 1: Interpret Application Request - This function will accept the application's request, with its accompanying input data, and interpret it to initiate the processing of the data to fulfill that request.

Function 2: Process Application Request - This function will take the interpreted data and process it to obtain the information requested by the application. This processing will be accomplished by lower level software, which the MAS will invoke.

Function 3: Format Application Requested Data - This function takes the processed data, which is in an internal format not readable by the application, and structures it into an acceptable final form for application processing.
Figure 3-6. IDEF0 A0 - Perform Model Access
3.2.3 System Translator

The System Translator is a software package that is used to transmit the Product Definition Data (PDD) between systems. The IDEF0 model defines the functions within the translator.

The IDEF0 charts are organized in the following manner:

AO Exchange PDD
  A1 Preprocess PDD
    A12 Create Data Section
  A2 Postprocess PDD
    A23 Process Data Section
    A24 Resolve Forward References

3.2.3.1 AO Exchange PDD

Figure 3-7 defines the functions that allow the exchange of PDD between systems. Any system may be a receiving system or a sending system. Thus, two functions are required of each system. The function of sending PDD is called preprocessing the PDD. The function of receiving PDD is called postprocessing the PDD. The computer user performs one of these functions by running the system.

Function 1: Preprocess PDD - The PDD is collected from the Sending system and formulated into an Exchange Format file.

Function 2: Postprocess PDD - The PDD is collected from the Exchange Format file and formulated into the receiving system internal form.

3.2.3.2 A1 Preprocess PDD (Figure 3-8)

Function 1: Create Header Section - This function constructs the data needed to format and write the header section entities using the run-time information.

Function 2: Create Data Section - This function constructs the data needed to format and write the data section entities using the PDD model.
Figure 3-7. IDEF0 AO - Exchange PDD
Figure 3-8. Al - Preprocess PDD
3.2.3.3 A12 Create Data Section (Figure 3-9)

The PDD is extracted from the Working Form and encoded into the Data Section of the Exchange Format File.

Function 1: Obtain Entity from Working Form - This function extracts data from the Working Form model on an entity-by-entity basis. All the entities of a given kind are processed together. Upon completion, each entity is marked as having been processed.

Function 2: Obtain Data Dictionary for Entity - This function retrieves a Data Dictionary entry from the Data Dictionary library. It defines the number, type, and physical layout of an entity's attributes.

Function 3: Encode Data in Exchange Format - This function formats and writes the entity data to the Exchange Format file. Together, the Data Dictionary and the Exchange Format specifications control which data is processed and how it is processed.

3.2.3.4 A2 Postprocess PDD (Figure 3-10)

The PDD is extracted from the Exchange Format file and placed into the Working Form model.

Function 1: Process Header Section - This function reads in and interprets the information found in the HEADER section of the Exchange Format file. This function will not be expanded.

Function 2: Initialize Working Form - See Function 2 in diagram "A22 File/Retrieve PDD".

Function 3: Process Data Section - This function reads in and interprets the information found in the DATA section of the exchange format file. The majority of the product definition data resides in the DATA section of the Exchange Format file. This information is placed into the Working Form model.

Function 4: Resolve Forward References - This function resolves the constituent entity references that did not take place during the first pass of the exchange format file.
Figure 3-9. IDEF0 A12 - Create Data Section
Figure 3-10. IDEF0 A2 Postprocess PDD
3.2.3.5 **A23 Process Data Section** (Figure 3-11)

Each PDD entity in the DATA section is interpreted, validated, and placed into the Working Form model.

**Function 1: Determine Entity Kind** - This function determines the entity kind number required by the MAS. This information is based on the entity type keyword found in the Exchange Format file and performed on an entity-by-entity basis.

**Function 2: Obtain Data Dictionary for Entity** - This function reads in the Data Dictionary description of an entity kind from the Data Dictionary library. This function manages the internal storage of Data Dictionary descriptions.

**Function 3: Create Shell Entity** - This function creates an empty "shell" entity based upon the Data Dictionary description of the entity of interest. The entity's attribute data block is adequately sized but empty of values. The entity's constituent list is adequately sized but filled with nil entities.

**Function 4: Move Attributes into Entity** - This function reads attribute values from the exchange format file, converts them from character to a binary form if necessary, and places them into the entity's attribute data block or constituent list as appropriate.

3.2.3.6 **A24 Resolve Forward References** (Figure 3-12)

Entity constituent references that were unresolvable when first encountered in the Exchange Format file are resolved following the first pass over the Exchange Format file.
Figure 3-11. IDEF0 A23 Process Data Section
Figure 3-12. IDEFø A24 Resolve Forward References
Function 1: **Obtain Unresolved Reference Entities** - This function gathers all of the unresolved reference entities together for processing. These entities were created during the first processing pass when an entity reference proved unresolvable.

Function 2: **Resolve Reference to Constituent** - This function determines the entity key of the MAS entity that corresponds to the given entity reference.

Function 3: **Replace Reference in Constituent List** - This function replaces the key of the nil entity, in an entity's constituent list, with the key of the true constituent entity. This function applies when an entity is referenced by multiple parent entities.

### 3.3 Application Interfaces

This section addresses the application interfaces employed in the GMAP system. The first part of this section defines, illustrates, and discusses the interfaces between components of the GMAP/PDDI system software used to interface applications. The second part defines the alternative interface scenarios to be employed between applications.

#### 3.3.1 Interfaces Between GMAP/PDDI System Components

The process of creating a Exchange Format description of a model and moving the model to dissimilar CAD/CAM systems involves several interfaces, as shown in Figure 3-13. The letter prefixes in the figure are references to these interfaces, described briefly below and more fully in paragraphs 3.3.1.1 through 3.3.1.6.

A **Application Program/Working Form** - This is the interface through which the application program can create, modify or query the Working Form model.

B **User Interface/System Translator** - This is the interface through which the user can initiate, monitor and direct the execution of the System Translator.

C **Working Form/Exchange Format** - This is the interface through which the Exchange Format PDD model is translated into a Working Form model, or vice versa. This interface is the System Translator Pre and Postprocessors.

D **Working Form/Database Management System** - This is the interface through which PDD models may be stored on secondary memory systems.
**Exchange Format** - This is the medium through which PDD models may be communicated to other computer systems.

**Schema Manager** - This is the interface through which a conceptual schema is input to the system.

### 3.3.1.1 Application Program/Working Form

The MAS will allow the application program to create and access the Working Form model. To accomplish this, well-defined interface routines will be required, so that, by using standard subroutine calls, the following operations can be performed:

- Create an entity
- Delete an entity
- Modify an entity
- Query entities
- Visit entities.

The create/delete capabilities will allow entities to be added or deleted from the Working Form model. The modify capability will provide the ability to add, remove, replace or insert data into an entity. The application can get the users and/or constituents of an entity by using the query capability or using list operations. The visit capability allows operations to be performed on all constituents of entities or entity lists.

In addition, return codes are sent from the MAS to the application program to signal the status of each operation.

### 3.3.1.2 User Interface/System Translator

The interface between the application program and the translator provides a means of communicating user commands to the System Translator and for passing status messages. The interface should consist of menu handler software which will display the command options on the screen, accept the user's choice, identify the choice, transmit the command and process any status messages which are returned. By using the menus, the user will be able to: identify the model(s) that are to be stored into, or retrieved from, the Exchange Format; and initiate the translation process to convert from the Exchange Format PDD to the Working Form PDD, or vice versa.
Figure 3-13. Interfaces of the GMAP/PDDI Exchange System
3.3.1.3 Working Form/Exchange Format

The MAS will serve as the interface between the Working Form and the Translator. The MAS accepts and manages Working Form PDD. To accomplish this, the System Translator software must control, via subroutine calls to the MAS, the creation of the Working Form model from the translated PDD.

Conversely, the System Translator must accept the Working Form PDD from the MAS and create the Exchange Format file. To accomplish this, the System Translator software must control, via subroutine calls to the MAS, queries of the Working Form model to get the required data.

3.3.1.4 Working Form/Native Database Management System

An interface between the Working Form and the database file/retrieve system of the applications should have the necessary conversion routines to transform the PDD between the Working Form and the file structure of the database system. In addition, the interface should provide file and retrieve capabilities during model building and editing operations. Access to the Working Form will be via the MAS.

This capability is useful for applications systems which are used for actual construction of the model. It should not be required for systems which will simply read in the model from the Exchange Format for temporary processing purposes. Additionally, since the interface will be system dependent and not portable, it will be developed for each demonstration system as an internally funded item and will not be a contractual deliverable.

3.3.1.5 Exchange Format

The Exchange Format is a neutral data structure that is used to transfer data between dissimilar systems. It interfaces between the System Translator and its user system. The System Translator interface requirements are to be able to produce and access an Exchange Format file. The production of the file is known as encoding. The accessing of the file is known as decoding. The encode/decode software must accept user interface data and use it to identify the model(s) that are to be stored into or retrieved from the Exchange Format. It must then be able to either store and encode the PDD into the exchange format after obtaining it from the MAS, or it must retrieve and decode the PDD from the exchange format and prepare it for the MAS to create it.

Because the PDD consists of entities that are made of a unique name, attributes of integers, reals, strings, logicals, and constituent lists of entities, this software package must be able to process these items. Also,
since the user requires communication on the status of the job and the condition of the PDD, this software package must be able to provide messages about them.

Therefore, the interface requirements of the System Translator software are:

- To accept user interface data to perform the translation
- To provide user interface data to establish the condition of the translation
- To store/retrieve Exchange Format PDD
- To create/get Working Form PDD.

To accomplish this, the System Translator must use the encode/decode software to process PDD entities with names, attributes, and constituent lists and provide a status about them.

The user interface requirements will be provided in the specification of the detailed design of the Exchange Format. This specification includes a definition of the language, including its syntax, the entity types, including their structure and meaning, and instruction on how to interface to the format. This specification is the STEP File Structure.

3.3.1.6 Schema Manager

The Schema Manager is a software tool used to manage all aspects of the creation and interrogation of the Conceptual Schema. It provides an interface between the Conceptual Schema and the user through the MAS. This interface allows the user to add or modify entities, query, file and retrieve, and display or print the results of the Working Form of the schema.

3.3.2 Interfaces Between Applications

The GMAP system architecture provides a number of alternatives to interfacing dissimilar computer based applications. These alternatives arise due to the different environments for exchange of PDD.

These may be loosely classified as:

- Transfer between dissimilar applications in a homogeneous computer environment
- Transfer between dissimilar applications in a heterogeneous computer environment.
A homogeneous environment is one within the same computer system. A heterogeneous environment is one between different computer systems. In this context, "same" and "different" refer to the physical representation of stored data. "Dissimilar" refers to any two applications which perform the same functions but logically represent information differently.

Each of the alternatives within the above environments are discussed in paragraphs 3.3.2.1 through 3.3.2.4.

3.3.2.1 Working Form as Exchange Mechanism

Transfer between different applications in a homogeneous computer environment can be accomplished by exchanging data through the Working Form. Figure 3-14 illustrates one possible scenario using this approach.

In this example, two dissimilar applications (A and B) each have unique data representations. Data from application A is translated to the GMAP representation and then put in the Working Form using the MAS. Application B also uses the MAS to retrieve the Working Form and then converts the GMAP representation into its native representation. In this example, the Working Form is the transfer vehicle for two dissimilar applications each with unique native representations.

3.3.2.2 Working Form as the Native Form

The scenario illustrated in Figure 3-15 shows the Working Form as the transfer mechanism and as the native representation. In this case, application A uses the GMAP representation as its internal data representation. Application B does the same. Hence, data mappings between applications are not required. This example illustrates two applications residing within the same computer system, based on the same data representation and the same physical storage.

3.3.2.3 Exchange Format and Working Form

The scenario illustrated in Figure 3-16 shows the Exchange Format as the transfer mechanism between the two dissimilar applications A and B, each residing on different computer systems. The Exchange Format is a neutral machine independent representation of the GMAP PDD. In this case, data from application A are converted to the GMAP representation and put in the Working Form using MAS. The System Translator maps the Working Form physical representation to the Exchange Format physical representation. This format is communicated via an exchange file between systems. The transportable System Translator written to run on system 2 converts the exchange file to a Working Form representation on system 2. Application B uses the MAS to retrieve the GMAP represented data from the Working Form for conversion to the native representation of application B.
Figure 3-14. Exchange of Dissimilar Native Representations through the WF
Figure 3-15. Transfer of Similar Native Representations Based on the WF
Figure 3-16. Transfer of Dissimilar Native Representations Using the WF and EF
3.3.2.4 Transfer Using the Working Form, Exchange Format, and Direct Translator to the Exchange Format

The scenario illustrated in Figure 3-17 shows the Exchange Format as the transfer mechanism between the two dissimilar applications A and B, each residing on different computer systems. The Exchange Format is a neutral machine independent representation of the GMAP PDD. In this case, data from application A are converted to the GMAP representation and put in the Working Form using the MAS. The System Translator converts the Working Form physical representation to the Exchange Format physical representation. This format is communicated via an exchange file between systems. Application B retrieves data directly from the Exchange Format.

Figure 3-17. Transfer of Dissimilar Native Representations Using the Working Form, Exchange Format and Direct Translator to the Exchange Format
3.4 Program Interrupts

3.4.1 Schema Manager

EMLN000
.ALARM=NO .HELP=NO

EMLN001 /REENTER SELECTION NUMBER
.ALARM=YES
'THE NUMBER/LETTER SELECTED IS NOT A VALID OPTION'

EMLN002 /NAME NOT UNIQUE
.ALARM=YES
'THE FIRST SIX CHARACTERS OF THE NAME MUST BE UNIQUE'

EMLN003 /BEGIN WITH A CHARACTER
.ALARM=YES
'THE FIRST POSITION OF THE NAME MUST BE A CHARACTER'

EMLN004 /NUMBER MUST BE INTEGER
.ALARM=YES
'THE KIND NUMBER MUST BE AN INTEGER'

EMLN005 /DOES NOT EXIST
.ALARM=YES
'AN ENTITY WITH THE GIVEN KIND NUMBER DOES NOT EXIST'

EMLN006 /ENTER (K) OR (R)
.ALARM=YES
'ENTER (K) FOR KEY OF (R) FOR ROLE'

EMLN007 /ENTER (O) OR (R)
.ALARM=YES
'ENTER (O) FOR OPTIONAL OR (R) FOR REQUIRED'

EMLN008 /ENTER (I) OR (D)
.ALARM=YES
'ENTER (I) FOR INDEPENDENT OF (D) FOR DEPENDENT'

EMLN009 /INVALID TYPE
.ALARM=YES
'THE NUMBER SELECTED IS NOT A VALID TYPE'

EMLN010 /OUT OF RANGE
.ALARM=YES
'THE NUMBER SELECTED IS OUT OF RANGE'

EMLN011 /MUST HAVE ONE MEMBER
.ALARM=YES
'THE ENTITY MUST HAVE AT LEAST ONE CONSTITUENT'

EMLN012 /INVALID OPERATION
.ALARM=YES
'THE OPERATION SELECTED IS NOT VALID'
EMLN013 /NO MEMBERS EXIST' .ALARM=YES
'NO MEMBERS EXIST TO BE DISPLAYED'

EMLN011 /MUST HAVE ONE MEMBER' .ALARM=YES
'THE ENTITY MUST HAVE AT LEAST ONE CONSTITUENT'

EMLN012 /INVALID OPERATION' .ALARM=YES
'THE OPERATION SELECTED IS NOT VALID'

EMLN013 /NO MEMBERS EXIST' .ALARM=YES
'NO MEMBERS EXIST TO BE DISPLAYED'

EMLN014 /NUMBER MUST BE UNIQUE' .ALARM=YES
'THE KIND NUMBER SELECTED IS NOT UNIQUE'

EMLN015 /REENTER THE NAME' .ALARM=YES
'REENTER THE NAME FOR THE ENTITY'

EMLN016 /REENTER THE KIND NUMBER' .ALARM=YES
'REENTER THE KIND NUMBER FOR THE ENTITY'

EMLN017 /REENTER THE HIGH BOUND' .ALARM=YES
'REENTER THE HIGH BOUND FOR THE ARRAY'

EMLN018 /REENTER THE LOW BOUND' .ALARM=YES
'REENTER THE LOW BOUND FOR THE ARRAY'

EMLN019 /NO ENTITIES OF THIS TYPE' .ALARM=YES
'NO ENTITIES EXIST IN THE SCHEMA OF THIS TYPE'

EMLN020 /SELECT ONLY ONE MEMBER' .ALARM=YES
'ONLY ONE MEMBER MAY BE SELECTED FROM THE DISPLAY'

EMLN021 /REENTER ARRAY BOUND(S)' .ALARM=YES
'LOW BOUND CANNOT BE GREATER THAN HIGH BOUND'

EMLN022 /CANNOT UPDATE' .ALARM=YES
'UPDATE A SHARED DEFINED TYPE THROUGH EDIT MENU'

EMLN023 /CANNOT DELETE ENTITY' .ALARM=YES
'THE ENTITY IS USED BY OTHERS FOR THEIR DEFINITION'

EMLN024 /INCOMPLETE MEMBER LIST' .ALARM=YES
'MEMBER LIMIT EXCEEDED. CONTACT PROGRAMMER'

EMLN025 /INVALID NAME/CHARACTER' .ALARM=YES
'NAME CONTAINS A SPACE OR AN INVALID CHARACTER'
3.4.2 Model Access Software with Name Value Interface

The MAS interface routines will acknowledge successful completion or failure of an operation by returning an internal return code (IRC). A return code of 0 will denote successful processing of an application's request. A positive return code will denote a fatal error and a negative return code will denote a warning condition. Table 3-1 provides the types of codes and explanation that will be used.

**TABLE 3-1**

**MAS INTERNAL RETURN CODES**

<table>
<thead>
<tr>
<th>IRC</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful Process</td>
</tr>
<tr>
<td>1</td>
<td>Invalid Entity Kind</td>
</tr>
<tr>
<td>2</td>
<td>Invalid Create</td>
</tr>
<tr>
<td>3</td>
<td>Cant Create List</td>
</tr>
<tr>
<td>4</td>
<td>MAS Init Failed</td>
</tr>
<tr>
<td>5</td>
<td>Invalid Update</td>
</tr>
<tr>
<td>6</td>
<td>Cant Update Entity</td>
</tr>
<tr>
<td>7</td>
<td>Cant Create Entity</td>
</tr>
<tr>
<td>8</td>
<td>Cant Verify Connect</td>
</tr>
<tr>
<td>9</td>
<td>Invalid Connection</td>
</tr>
<tr>
<td>10</td>
<td>Cant Connect</td>
</tr>
<tr>
<td>11</td>
<td>Absent Input</td>
</tr>
<tr>
<td>12</td>
<td>Invalid Get</td>
</tr>
<tr>
<td>13</td>
<td>NDS Operation Complete</td>
</tr>
<tr>
<td>14</td>
<td>Bad List Position</td>
</tr>
<tr>
<td>15</td>
<td>Maximum List Size</td>
</tr>
</tbody>
</table>
### TABLE 3-1 (Continued)

<table>
<thead>
<tr>
<th>IRC</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Bad List Move Count</td>
</tr>
<tr>
<td>17</td>
<td>Bad List Reference</td>
</tr>
<tr>
<td>18</td>
<td>Bad Entity Key</td>
</tr>
<tr>
<td>19</td>
<td>Duplicate Schema</td>
</tr>
<tr>
<td>20</td>
<td>Dump Error</td>
</tr>
<tr>
<td>21</td>
<td>Bad Entity Size</td>
</tr>
<tr>
<td>22</td>
<td>Bad Schema Kind</td>
</tr>
<tr>
<td>23</td>
<td>Proc Code Error</td>
</tr>
<tr>
<td>24</td>
<td>Proc Out of Range</td>
</tr>
<tr>
<td>25</td>
<td>No Match Found</td>
</tr>
<tr>
<td>26</td>
<td>Dups Not Removed</td>
</tr>
<tr>
<td>27</td>
<td>Invalid Delete</td>
</tr>
<tr>
<td>28</td>
<td>Bad Entity on User List</td>
</tr>
<tr>
<td>29</td>
<td>Bad Delete Key</td>
</tr>
<tr>
<td>30</td>
<td>Empty Model</td>
</tr>
<tr>
<td>31</td>
<td>ARG Out of Range</td>
</tr>
<tr>
<td>32</td>
<td>Invalid CRB Position</td>
</tr>
<tr>
<td>33</td>
<td>CRB Entry Found</td>
</tr>
<tr>
<td>34</td>
<td>Invalid Flag Name</td>
</tr>
<tr>
<td>35</td>
<td>Cant Mark Entity Delete</td>
</tr>
<tr>
<td>36</td>
<td>Size Not Large Enough</td>
</tr>
<tr>
<td>37</td>
<td>RTS Not in Working Form</td>
</tr>
<tr>
<td>38</td>
<td>Core Not Available</td>
</tr>
<tr>
<td>39</td>
<td>Not Enough Core for INIT</td>
</tr>
<tr>
<td>40</td>
<td>Absolutely No More Core</td>
</tr>
<tr>
<td>41</td>
<td>MAINIT Already Done</td>
</tr>
<tr>
<td>42</td>
<td>Rule Does Not Match</td>
</tr>
<tr>
<td>43</td>
<td>Entity Not Found List</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IRC</th>
<th>Warning Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>No Such Schema</td>
</tr>
<tr>
<td>-2</td>
<td>Proc Warning Code</td>
</tr>
<tr>
<td>-3</td>
<td>Empty Delete List</td>
</tr>
<tr>
<td>-4</td>
<td>Empty Exception List</td>
</tr>
<tr>
<td>-5</td>
<td>End of List</td>
</tr>
<tr>
<td>-6</td>
<td>No List Created</td>
</tr>
<tr>
<td>-7</td>
<td>Empty Mark List</td>
</tr>
<tr>
<td>-8</td>
<td>Empty Mark and Exception List</td>
</tr>
<tr>
<td>-9</td>
<td>Empty Delete and Exception List</td>
</tr>
<tr>
<td>-10</td>
<td>Empty Mark and Delete List</td>
</tr>
</tbody>
</table>
TABLE 3-1 (Continued)

Name/Value Interface

<table>
<thead>
<tr>
<th>IRC</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Entity kind is not defined in run-time subschema</td>
</tr>
<tr>
<td>2</td>
<td>Attribute name is not defined for the entity in the</td>
</tr>
<tr>
<td></td>
<td>run-time subschema</td>
</tr>
<tr>
<td>3</td>
<td>Entity key is nil</td>
</tr>
<tr>
<td>4</td>
<td>Call to MAS routine MAEXEQ failed</td>
</tr>
<tr>
<td>5</td>
<td>Call to MAS routine MAL failed</td>
</tr>
</tbody>
</table>

IRC Warning Type

-1

-1 Invalid entry in the Data Dictionary was detected during the translation of an attribute; the warning message written to ddname = OUTPUT describes the error in detail

3.5 Timing and Sequencing Description

Figure 3-13 illustrated the interface between all components of the system architecture. The MAS is the key to this architecture. Data can only be created and accessed from the Working Form model by the MAS. The MAS is an integral part of the Schema Manager and exchange systems.

A Conceptual Schema of the data must first be present before models can be created and exchanged. The Schema Manager processes a human readable form (EXPRESS) of the Conceptual Schema to generate physical schema files that the exchange system uses. Forms of the physical schema files generated are the data dictionary and PASCAL include files.

The data dictionary form of the physical schema is an external file accessed at runtime by the pre- and postprocessor to determine the types and locations of data in Working Form memory. The PASCAL include form of the physical schema is a compile-time bound file that defines data for PASCAL routines.
Once the physical schema files are defined for the exchange system, the modeling software can create Working Form models; the data usage applications can get data from the Working Form models; the preprocessor can get data from the Working Form models and convert it into an Exchange Format file; the postprocessor can get data from an Exchange Format file and convert it into a Working Form model; and the N/VI can be used by any of the software to get from and put data into the Working Form models without knowledge of data content and location.

3.6 Data Dictionary

3.6.1 Schema Manager

This section provides the data structures for the Schema Manager. The following index provides an alphabetic listing of the function.

BALTYP - Boundary alignment types

BATTYP - Batch interface constants and types

DDTYP - Data Dictionary constants and types

EDBDEF - Contains the files structure for accessing the Working Form Data Dictionary

LEXTYP - Batch interface constants and types

NVITYP - N/VI constants and types

RTSTYP - Run-Time Subschema constants and types

SCECON - Schema Manager constants

SCETYP - Schema Manager types
(* (BALTYP) *********************************************)
(* CHANGE CONTROL: *)
(* ORIGINATED: 17 SEPTEMBER 1986, M. H. CHOI, DBMA *)
(* REVISED: 15 JANUARY 1988, M. H. CHOI, DBMA *)
(* ADD T_ALIGN AND T_STRUCT POINTER. *)
(* *)
(* ******************************************)*)

TYPE
(* *)
(* *)

T_DW_POINTER = @T_DW_FRAME;
T_DW_FRAME = RECORD
  NEXT : T_DW_POINTER;
  NAME : T_NAME;
  SIZE : INTEGER;
END;
(* *)

T_FW_POINTER = @T_FW_FRAME;
T_FW_FRAME = RECORD
  NEXT : T_FW_POINTER;
  NAME : T_NAME;
  SIZE : INTEGER;
END;
(* *)

T_HW_POINTER = @T_HW_FRAME;
T_HW_FRAME = RECORD
  NEXT : T_HW_POINTER;
  NAME : T_NAME;
  SIZE : INTEGER;
END;
(* *)

T_BY_POINTER = @T_BY_FRAME;
T_BY_FRAME = RECORD
  NEXT : T_BY_POINTER;
  NAME : T_NAME;
  SIZE : INTEGER;
END;
(* *)

T_PTR_POINTER = @T_PTR_FRAME;
T_PTR_FRAME = RECORD
  NEXT : T_PTR_POINTER;
  NAME : T_NAME;
  SIZE : INTEGER;
END;
(* *)
T_LIST_BOUNDARY = RECORD
  NAME : T_NAME;
  OFFSET : INTEGER;
  SIZE : INTEGER;
END;

(*
T_OFFSET_LIST = ARRAY (1..100) OF T_LIST_BOUNDARY;
(*

T_HIGH_BOUND = ARRAY (1..100) OF INTEGER;
T_LOW_BOUND = ARRAY (1..100) OF INTEGER;
(*

T_OPTIONAL_GLOBAL = RECORD
  KEY : ENTKEY;
  POSITION : INTEGER;
END;

(*
T_GLOBAL_FIELD = ARRAY (1..100) OF T_OPTIONAL_GLOBAL;
(*

T_ALIGN = (DW, FW, HW, BY, PNTR, NONE);(*

T_S_Info_Pointer = @T_S_Info_Frame;
T_S_Info_Frame = Record
  Name : T_Name;
  Offset : Integer;
  Size : Integer;
  Next : T_S_Info_Pointer;
END;

(*
T_Struc_Pointer = @T_Struc_Frame;
T_Struc_Frame = Record
  Next : T_Struc_pointer;
  Name : T_Name;
  Defn : T_S_Info_Pointer;
END;

(*
(* END %INCLUDE BALTYP *************************************************************************)

3-36
CONST
FIRST = 1;

TYPE
BLKDATA = RECORD
  NAME : T_NAME;
  KIND : INTEGER;
  NAME_FOUND : BOOLEAN;
  KIND_FOUND : BOOLEAN;
  KEY : ENTKEY;
END;

ENTITY_LIST_PTR = @ENTITY_INFORMATION;

ENTITY_INFORMATION = RECORD
  NAME : T_NAME;
  NUMBER : INTEGER;
  NEXT : ENTITY_LIST_PTR;
END;

LEXSTRING = STRING(MAX_LENGTH_VALUE);

TEXPECTED = RECORD
  ENTRIES : INTEGER;
  TOKEN_VALUE : ARRAY(.1..20.) OF T_TOKEN_VALUE
END;

(* BEGIN %INCLUDE BATTYP **********************************************)
(* $CHANGE CONTROL: **************************************************)
(* ORIGINATED: 20 MARCH 1987, C. H. MOHME *)
(* **********************************************)

(* END %INCLUDE BATTYP **********************************************)

3-37
(* (DDTYP) DATA DICTIONARY CONSTANTS AND TYPES *******************************

(* $CHANGE CONTROL: *)
(* REVISED : 15 MAY 1987, M. H. CHOI, DBMA *)
(* ADDED T_ADB_RECORD AND T_CL_RECORD TO RETURN *)
(* THE DEFINITIONS IN PHYSICAL SCHEMA ORDER *)
(* ORIGINATED: 19 MARCH 1987, M. H. CHOI, DBMA *)
(* *)

*******************************************************************************)

CONST

CONTINUATION_FLAG = 'X';
PHYSICAL_ORDER = 'P';
MAX_ARRAY_REC = 100;

(* RETURN CODE VALUES: *)
KIND_NOT_IN_DATA_DICTIONARY = 1;
ACTUAL_SIZE_GT_SPACE_AVAIL = -1;

(* TYPE *)
T_INX_RECORD = PACKED ARRAY (. 1..80 .) OF CHAR;

(* T_USER_ARRAY = ARRAY (. 1..46 .) OF
PACKED ARRAY (. 1..80 .) OF CHAR; *)

(* T_BOUNDS = RECORD
LBOUND : PACKED ARRAY (. 1..4 .) OF CHAR;
UBOUND : PACKED ARRAY (. 1..4 .) OF CHAR;
END;*)

(* T_ADB_RECORD = RECORD
ADB_KEY : INTEGER;
POSITION : INTEGER;
NO_OF_REC : INTEGER;
END; *)

(* T_CL_RECORD = RECORD
POSITION : INTEGER;
NO_OF_REC : INTEGER;
END; *)

(* T_VARIANT_RECORD = RECORD
CASE INTEGER OF
0 : ( BASIC_RECORD : PACKED ARRAY (. 1..80 .) OF CHAR );
1 : ( BOUNDS : ARRAY (. 1..9 .) OF T_BOUNDS );
2 : ( C_FLAG : CHAR;
NO_OF_KINDS : PACKED ARRAY (. 1..2 .) OF CHAR; *)

3-38
KINDS : ARRAY (1..12) OF PACKED ARRAY (1..6) OF CHAR;

3 : (E_FLAG : CHAR;
   NO_OF_VALUES : PACKED ARRAY (1..2) OF CHAR;
   VALUES : ARRAY (1..4) OF PACKED ARRAY (1..17) OF CHAR);

END;

(*)
T_ADB_ARRAY = ARRAY (1..MAX_ARRAY_REC) OF T_ADB_RECORD;
(*
T_CL_ARRAY = ARRAY (1..MAX_ARRAY_REC) OF T_CL_RECORD;
(*
T_FILE_VARIANT = FILE OF T_VARIANT_RECORD;
(*
T_INX_FILE = FILE OF T_INX_RECORD;
(*
(* END %INCLUDE DDTYP *******************************************
(* CONTAINS THE FILE STRUCTURE FOR ACCESSING THE WORKING FORM *)
(* DATA DICTIONARY *)
(* *)

**(********************************************************************************************)**
(* *)
(* ENDDEF *)
(* *)
(*)

**TYPE**

SCALAR_REC = RECORD
  DNUMB : INTEGER;
  DSCALAR : ARRAY (.1..10.) OF
    PACKED ARRAY (.1..16.) OF CHAR
END;

ENTITY_REC = RECORD
  CLDISP : INTEGER;
  DNUMB : INTEGER;
  DENTITY : ARRAY (.1..36.) OF INTEGER
END;

SUBENT_REC = RECORD
  CLDISP : INTEGER;
  DNUMB : INTEGER;
  DSUBENT : ARRAY (.1..36.) OF INTEGER
END;

BOUND_REC = ARRAY (.1..10.) OF INTEGER;

D_ENTITY = RECORD
  DNAME : PACKED ARRAY (.1..16.) OF CHAR;
  CSORDR : INTEGER;
  DMIN : INTEGER;
  DMAX : INTEGER;
  DDIM : INTEGER;
  DBOUNDS : BOUND_REC;
  DTYPE: INTEGER;
  DSIZE : INTEGER;
  DDISP : INTEGER;
  CASE DTYPE : OF
    5 : (DSCA : SCALAR_REC);
    7 : (DENT : ENTITY_REC);
    8 : (DSEN : SUBENT_REC)
END;

DICTYP = ARRAY (.1..45.) OF D_ENTITY;

3-40
CONST
Final_Token = 'Final';
Found_Addop = 'Addop';
Found_Assignment = 'Assignment';
Found_Comment = 'Comment';
Found_Delimiter = 'Delimiter';
Found_Error = 'Error';
Found_Identifier = 'Identifier';
Found_Integer = 'Integer';
Found_Keyword = 'Keyword';
Found_Literal = 'Literal';
Found_Mulop = 'Mulop';
Found_Pound_Sign = 'Pound_Sign';
Found_Real = 'Real';
Found_Relop = 'Relop';
Found_Unary = 'Unary';
Initial_Token = 'Initial';

(* ARBITRARY SIZE PARAMETERS: *)
Max_Length_Token = 10;
Max_Length_Value = 80;

Type
T_Token = String(Max_Length_Token);
T_Token_Value = String(Max_Length_Value);

(* END %INCLUDE LEXTYP ***********)
(* (NVITYP) NAME/VALUE INTERFACE CONSTANTS AND TYPES ***************)

(* $CHANGE CONTROL: *)
(* REVISED : 15 JULY 1987, M. H. CHOI, DBMA *)
(* ADDED COMPARISON VALUES *)
(* REVISED : 16 SEPTEMBER 1986, M. H. CHOI, DBMA *)
(* CHANGED STRUCTURE OF THE SCHEMA INSTANCE COLLECTOR *)
(* BECAUSE STRUCTURE CHANGED IN MAS TO HANDLE NEW *)
(* DELETE AND COMPRESS RULES *)
(* REVISED : 12 SEPTEMBER 1986, M. H. CHOI, DBMA *)
(* ADDED FIELD TO T_INT_ITEM FOR THE COLL_ADB AND *)
(* MAPROB2 BECAUSE T_INT_ITEM STRUCTURE CHANGED IN MAS*)
(* REVISED : 04 JUNE 1986, M. H. CHOI, DBMA *)
(* ADDED RETURN CODE VALUES FOR FAILED_IN_MAL AND *)
(* FAILURE. *)
(* ORIGINATED: 13 MAY 1986, G. A. WHITE, DBMA *)
(* *)

(* *******************************************************)

CONST

(* ARBITRARY SIZE PARAMETERS: *)
MAX_ARRAY = 100;
MAX_ATTRIBUTE_NAME = 1000;
MAX_ATTRIBUTE_VALUE = 1000;
MAX_CHARS = 1000;
MAX_GROL2 = 8;
MAX_LIST = 4000000;
MAX_RDB_SIZE = 65535;
MAX_WORDS = MAX_CHARS DIV 4;
SCHEMANAME_SIZE = 16;

(* RETURN CODE VALUES: *)
INVALID_ARRAY_ENTITY = 7;
INVALID_SCALAR_VALUE = 6;
FAILED_IN_MAL = 5;
FAILED_IN_MAEXEQ = 4;
NIL_ENTITY_KEY = 3;
ATTRIBUTE_NOT_IN_ENTITY = 2;
KIND_NOT_IN_RUNTIME_SUBSCHEMA = 1;
SUCCESS = 0;
WARNING = -1;
FAILURE = -2;

(* COMPARISON VALUES: *)
EQ = 1;
LT = 2;
GT = 3;
NE = 4;
LE = 5;
GE = 6;

3-42
(* ATTRIBUTE NAME STRING DELIMITERS: *)
END_OF_SEGMENT = '.';
END_OF_STRING = '00'XC;

(* DIMENSION VALUE DELIMITERS: *)
BEGIN_OF_ARRAY = '(';
END_OF_ARRAY = ')';

TYPE
T_ATTRIBUTE_NAME = ARRAY(. 1..MAX_ATTRIBUTE_NAME .) OF CHAR;
T_DIMEN_VALUE = ARRAY(. 1..MAX_ARRAY .) OF INTEGER;
T_DISPLAY_WORD = PACKED ARRAY(. 1..8 .) OF CHAR;
T_HEX_BYTE = PACKED 0..255;
T_HEX_WORD = ARRAY(. 1..4 .) OF T_HEX_BYTE;
T_Word = ARRAY(. 1..4 .) OF Char;
T_LOCATION = (IN_ADB, IN_CL, IN_ENUMERATION, IN_STRUCTURE);
T_SCHEMA_NAME = PACKED ARRAY (. 1..SCHEMA_NAME_SIZE .) OF Char;
T_SELECTOR = PACKED 0..9;
T_INTEGER_1 = PACKED -128..127;
T_INTEGER_2 = PACKED -32768..32767;
T_INTEGER_4 = MININT..MAXINT;
T_VALUE = ARRAY(. 1..MAX_ATTRIBUTE_VALUE .) OF CHAR;
T_Array_Value = ARRAY(. 1..MAX_ATTRIBUTE_VALUE .) OF T_Word;

(* T_ATTRIBUTE_VALUE = RECORD *)
CASE INTEGER OF
0 : ( AS_VARIANT : T_Value );
1 : ( AS_INTEGER_1 : T_Integer_1 );
2 : ( AS_INTEGER_2 : T_Integer_2 );
3 : ( AS_INTEGER_4 : T_Integer_4 );
4 : ( AS_REAL_4 : SHORTREAL );
5 : ( AS_REAL_8 : REAL );
6 : ( AS_LOGICAL : BOOLEAN );
7 : ( AS_ENUMERATION : T_Schema_Name );
8 : ( AS_Array : T_Array_Value );
END;

(* T_VARIANT_VALUE = RECORD *)
CASE INTEGER OF
0 : ( AS_VARIANT : T_VALUE );
1 : ( AS_INTEGER : T_INTEGER_4 );
2 : ( AS_REAL : REAL );
END;

(* ROUTINE = PACKED ARRAY(. 1..8 .) OF CHAR; *)

(* ENTDATA = RECORD *)

3-43
CASE INTEGER OF
  0 : (CHARS : PACKED ARRAY(. 1..MAX_CHARS .) OF CHAR);
  1 : (WORDS : ARRAY(. 1..MAX_WORDS .) OF T_HEX_WORD );
END;

(*       EXT_RET_CODE = INTEGER;       *)
(*       LISTINDEX = 0..MAX_LIST;        *)
(*       LISTPSTN = 0..MAX_LIST;          *)
(*       LISTSIZE = 0..MAX_LIST;           *)
(*       ORD_KIND = 0..MAXINT;             *)
(*       RDBSIZE = PACKED 0..MAX_RDB_SIZE;  *)
(*       T_RULE_ELMNTS = ( COMPRESS, DELETE, USER_DELETE, CNST_DELETE ); *)
(*       T_RULE = SET OF T_RULE_ELMNTS;     *)
(*       T_GROUP = RECORD
            LAST_CNST : RDBSIZE;
            RULE : T_RULE;
        END;                *)
(*       T_GROUP_ARRAY = ARRAY (. 1..MAX_GROUP .) OF T_GROUP; *)
(*       T_SCH_INST_ENT = RECORD
            KIND : ORD_KIND;
            POSITION : LISTPSTN;
            NUM_GROUP: LISTPSTN;
            MIN_CNST : LISTPSTN;
            GROUP : T_GROUP_ARRAY;
        END;             *)
(*       EMTBLOCK = RECORD
            KIND : ORD_KIND;
            SIZE : 0..MAX_CHARS;
            SYSUSE : ARRAY (. 1..4 .) OF BOOLEAN;
            DATA : ENTDATA;
        END;          *)
(*       ENTPNTR = @EMTBLOCK;              *)
(*       LISTPNTR = @T_SYS_LIST;           *)
T_INT_ITEM = RECORD
  RDBEXIST : BOOLEAN;
  MAPROB : BOOLEAN;
  MAPROB2 : BOOLEAN;
  COLL_ADB : ENTPNTR;
  USERS : LISTPNTR;
  CNSTS : LISTPNTR;
  ENPTR : ENTPNTR;
END;

ENTITIES = ( NIL-ENT, INT_ROOT, INT_ITEM, APPL_LIST );

T_ENTITY = RECORD
  FORM : ENTITIES;
  IIT : T_INT_ITEM;
END;

ANYKEY = RECORD
  P : @T_ENTITY;
END;

ENTKEY = ANYKEY;

LISTKEY = ANYKEY;

T_SYS_LIST = RECORD
  SIZE : LISTSIZE;
  LSTLNG : LISTSIZE;
  LIST : ARRAY (.1..MAX_LIST.) OF ENTKEY;
END;

T_DEFN_POINTER = @T_DEFN_FRAME;
T_DEFN_FRAME = RECORD
  NEXT : T_DEFN_POINTER;
  KIND : ORD_KIND;
  DATA_TYPE : INTEGER;
  CASE INTEGER OF
    1, 2, 3, 4 : ( OFFSET : INTEGER;
                  SIZE : INTEGER );
    7, 8 : ( POSITION : INTEGER);
    5 : ( SELECTOR_OFFSET : INTEGER;
          TABLE_OFFSET : INTEGER );
END;
T_NAME_POINTER = @T_NAME_FRAME;
T_NAME_FRAME  = RECORD
   NAME : T_SCHEMA_NAME;
   NEXT : T_NAME_POINTER;
   DEFN : T_DEFN_POINTER;
END;

T_DATAREC    = RECORD
   NAME_ROOT  : T_NAME_POINTER;
   LIST_ROOT  : LISTKEY;
   ATTRIBUTE_VALUE : T_VARIANT_VALUE;
   DIMEN_VALUE : T_DIMEN_VALUE;
   NO_OF_DIMENSIONS : INTEGER;
END;

BLKDATA = T_DATAREC;
(* (RTSTYP) RUN-TIME SUBSCHEMA CONSTANTS AND TYPES *******)
(*                       *)
(* $CHANGE CONTROL: *)
(*   REVISED: 8 SEPTEMBER 1987, M. H. CHOI, DBMA  *)
(*   ADDED MINIMUM OCCURENCE FIELD TO T_ATTRIBUTE *)
(*   ORIGINATED: 17 SEPTEMBER 1986, M. H. CHOI, DBMA *)
(*                       *)
(*****************************************************************)
CONST
(* ARBITRARY SIZE PARAMETERS: *)
MAX_ARRAY_POINTER = 100;
MAX_ATTRIBUTE = 100;
MAX_ENUMERATION = 100;
MAX_ENUM_INDEX = 100;
MAX_ARRAY_LIST = 100;
MAX_ARRAY_INDEX = 100;
MAX_CL_LIST = 100;
MAX_CL_KINDS = 100;

TYPE
T_STRING_8 = PACKED ARRAY (.1..8.) OF CHAR;
T_DATA_TYPE = ( INTEGER_DT, REAL_DT, STRING_DT, LOGICAL_DT,
                ENUM_DT, PNTR_DT, ARRAY_DT );

(* T_ATTRIBUTE = RECORD *)
  NAME : T_SCHEMA_NAME;
  MIN_OCC : INTEGER;
  DATA_TYPE : INTEGER;
  CASE INTEGER OF
    1, 2, 3, 4 : ( OFFSET : INTEGER;
                  SIZE : INTEGER );
    7, 8 : ( POSITION : INTEGER );
    5, 10 : ( SELECTOR_OFFSET : INTEGER;
               TABLE_INX_POSITION : INTEGER );
  END;

(* P_ATTRIBUTE = RECORD *)
  NAME : T_SCHEMA_NAME;
  PSTN : INTEGER;
  NAME_TYPE : CHAR;
  DATA_TYPE : INTEGER;
  CASE INTEGER OF
    1, 2, 3, 4 : ( OFFSET : INTEGER;
                   SIZE : INTEGER );
    7, 8 : ( POSITION : INTEGER );
    5, 10 : ( SELECTOR_OFFSET : INTEGER;
              TABLE_INX_POSITION : INTEGER );
  END;
T_ENUM_INDEX = ARRAY (. 1..MAX_ENUM_INDEX .) OF RECORD
  NO_OF_ENTRIES : INTEGER;
  TABLE_INDEX_POSITION : INTEGER;
END;

(* )
T_ENUMERATION = ARRAY (.1..MAX_ENUMERATION.) OF T_SCHEMA_NAME;
(* )
T_ARRAY_INDEX = ARRAY (. 1..MAX_ARRAY_INDEX .) OF RECORD
  NO_OF_DIMENS : INTEGER;
  TABLE_INDEX_POSITION : INTEGER;
END;

(* )
T_ARRAY_LIST = ARRAY (. 1..MAX_ARRAY_LIST .) OF RECORD
  SIZE : INTEGER;
  LOW_BOUND : INTEGER;
END;

(* )
T_CL_INDEX = ARRAY (. 1..MAX_CL_LIST .) OF RECORD
  NO_OF_CL_KINDS : INTEGER;
  TABLE_INDEX_POSITION : INTEGER;
END;

(* )
T_CL_KINDS = ARRAY (. 1..MAX_CL_KINDS .) OF INTEGER;

(* )
T_SCHEMA_POINTER = @T_SCHEMA;
T_SCHEMA = RECORD
  NAME : T_SCHEMA_NAME;
  KIND : INTEGER;
  ATTRIBUTE_COUNT : INTEGER;
  ENUM_INDEX_OFFSET : INTEGER;
  ENUM_VALUE_OFFSET : INTEGER;
  ARRAY_INDEX_OFFSET : INTEGER;
  ARRAY_LIST_OFFSET : INTEGER;
  CL_INDEX_OFFSET : INTEGER;
  CL_KINDS_OFFSET : INTEGER;
  ATTRIBUTE : ARRAY (. 1..MAX_ATTRIBUTE .) OF T_ATTRIBUTE;
END;

(* )
P_SCHEMA_POINTER = @P_SCHEMA;
P_SCHEMA = RECORD
  NAME : T_SCHEMA_NAME;
  KIND : INTEGER;
  ATTRIBUTE_COUNT : INTEGER;
  ENUM_INDEX_OFFSET : INTEGER;
  ENUM_VALUE_OFFSET : INTEGER;
  ARRAY_INDEX_OFFSET : INTEGER;
  ARRAY_LIST_OFFSET : INTEGER;
  CL_INDEX_OFFSET : INTEGER;
  CL_KINDS_OFFSET : INTEGER;
END;
CL_KINDS_OFFSET : INTEGER;
ATTRIBUTE : ARRAY (1..MAX_ATTRIBUTE) OF P_ATTRIBUTE;
END;

(*
T_RUN_TIME_POINTER = @T_RUN_TIME;
T_RUN_TIME = RECORD
ENTITY : T_SCHEMA;
ENUM_INDEX : T_ENUM_INDEX;
ENUM_VALUE : T_ENUMERATION;
ARRAY_INDEX : T_ARRAY_INDEX;
ARRAY_LIST : T_ARRAY_LIST;
CL_INDEX : T_CL_INDEX;
CL_KINDS : T_CL_KINDS;
END;
(*
P_RUN_TIME_POINTER = @P_RUN_TIME;
P_RUN_TIME = RECORD
ENTITY : P_SCHEMA;
ENUM_INDEX : T_ENUM_INDEX;
ENUM_VALUE : T_ENUMERATION;
ARRAY_INDEX : T_ARRAY_INDEX;
ARRAY_LIST : T_ARRAY_LIST;
CL_INDEX : T_CL_INDEX;
CL_KINDS : T_CL_KINDS;
END;
(*
T_KIND_ADB_POINTER = @T_KIND_ADB;
T_KIND_ADB = RECORD
SYSTEM_AREA : T_SCHEMA;ENT;
RUN_TIME : T_RUN_TIME;
END;
(*
T_ARRAY_LIST_COMPACTOR = RECORD
TABLE : T_ARRAY_LIST;
TABLE_SIZE : INTEGER;
END;
(*
T_ARRAY_INDEX_COMPACTOR = RECORD
TABLE : T_ARRAY_INDEX;
TABLE_SIZE : INTEGER;
END;
(*
T_ENUM_INDEX_COMPACTOR = RECORD
TABLE : T_ENUM_INDEX;
TABLE_SIZE : INTEGER;
END;
(*
T_ENUM_COMPACTOR = RECORD
TABLE : T_ENUMERATION;
3-49
TABLE_SIZE : INTEGER;
END;

(*
T_CL_INX_COMPACTOR = RECORD
 TABLE  : T_CL_INDEX;
 TABLE_SIZE  : INTEGER;
END;
(*)

T_CL_KINDS_COMPACTOR = RECORD
 TABLE  : T_CL_KINDS;
 TABLE_SIZE  : INTEGER;
END;

(*
T_DATA_VALUE = ARRAY (. 1..MAX_ATTRIBUTE_VALUE .) OF CHAR;
(*)

T_VARIANT_POINTER = RECORD
 CASE INTEGER OF
 0 : ( AS_ADD_SIZE  : LISTPSTN );
 1 : ( AS_INTEGER  : INTEGER );
 2 : ( TO_ATTRIBUTE_VALUE  : @T_VALUE );
 3 : ( TO_ENTITY  : ENTPNTR );
 4 : ( TO_ENUMERATION  : @T_ENUMERATION );
 5 : ( TO_SCHEMA  : T_SCHEMA_POINTER );
 6 : ( TO_SELECTOR  : @T_SELECTOR );
 7 : ( TO_ENUM_INDEX  : @T_ENUM_INDEX );
 8 : ( AS_RUN_TIME_POINTER  : T_RUN_TIME_POINTER );
 9 : ( AS_DATA  : @T_DATA_VALUE );
 10 : ( TO_ARRAY_INDEX  : @T_ARRAY_INDEX );
 11 : ( TO_ARRAY_LIST  : @T_ARRAY_LIST );
 12 : ( TO_CL_INDEX  : @T_CL_INDEX );
 13 : ( TO_CL_LIST  : @T_CL_KINDS);
 14 : ( TO_ENTKEY  : ENTPNTR );
 15 : ( TO_ARRAY_VALUE  : @T_Word );
 16 : ( TO_CNSTKEY  : LISTPSTN );
 17 : ( TO_INTEGER_1  : @T_INTEGER_1 );
 18 : ( TO_INTEGER_2  : @T_INTEGER_2 );
 19 : ( TO_INTEGER_4  : @T_INTEGER_4 );
 20 : ( TO_REAL_4  : @SHORTREAL );
 21 : ( TO_REAL_8  : @REAL );
 22 : ( TO_VALUE  : T_VALUE );
 23 : ( AS_RTS_POINTER  : P_RUN_TIME_POINTER );
END;
(*
(*
CONST
 MAX_BUFFER  = SIZEOF ( T_RUN_TIME );
(*
(* END %INCLUDE RTSTYP  ******************************************************)
(* %INCLUDE SCECON *)

(* CONSTANTS USED BY THE SCHEMA MANAGER PACKAGE *)

CONST

(* BLANK CONSTANTS *)

BLANK8 = ' '; BLANK16 = ' '; BLANK30 = ' '; BLANK40 = ' '; BLANK50 = ' '; BLANK_IDENTIFIER = ' '; BLANK_TABLE_VARIABLE = ' ';

(* CONSTANT LENGTHS *)

IDENTIFIER_LENGTH = 16;
UNIQUENESS_LENGTH = 14;
TABLE_VARIABLE_LENGTH = IDENTIFIER_LENGTH + 7;

(* CONSTANT LENGTH OF THE ENTITY ADB *)

ENTITY_ADB CONST_LENGTH = 28;

(* MINIMUMS AND MAXIMUMS *)

MINIMUM_INTEGER_PRECISION = 1;
MINIMUM_REAL_PRECISION = 1;
MINIMUM_STRING_LENGTH = 1;
MAXIMUM_INTEGER_PRECISION = 9;
MAXIMUM_REAL_PRECISION = 16;
MAXIMUM_STRING_LENGTH = 1000;
MINIMUM_ARRAY_BOUND = -99;
MAXIMUM_ARRAY_BOUND = 999;
MAX_ARRAY_SIZE = 1000;

(*-----------------------------------------------*)
(* ENTITY KIND NUMBERS *)
(*-----------------------------------------------*)

PRIMITIVE_KIND = 1000;
INTEGER_KIND = 1001;
REAL_KIND = 1002;
STRING_KIND = 1003;
LOGICAL_KIND = 1004;
ENUMERATION_KIND = 1005;
ENUMERITEM_KIND = 1006;
POINTER_KIND = 1008;
STRUCTURE_KIND = 1009;
ARRAY_KIND = 1010;
SUPERTYPE_KIND = 1016;
CLASS_KIND = 1017;
ENTITY_KIND = 1018;
FIELD_KIND = 1019;
DEFINED_TYPE_KIND = 1020;
SUBSCHEMA_KIND = 1021;
GLOBAL_FIELD_KIND = 1022;
BACKPATCH_KIND = 1023;
UNRESOLVED_KIND = 1024;
SCHEMA_KIND = 1089;

(*-----------------------------------------------*)
(* CONCEPTUAL SCHEMA REPORT CONSTANTS *)
(*-----------------------------------------------*)

(*-----------------------------------------------*)
(* THE MAXIMUM PAGE SIZE *)
(*-----------------------------------------------*)
MAX_PAGE_SIZE = 55;

(*-----------------------------------------------*)
(* RUN-TIME SUBSCHEMA CONSTANTS *)
(*-----------------------------------------------*)

MAX_ATTRIBUTE_VALUE = 1000;
MAX_CHARS = 1000;
MAX_LIST = 4000000;
MAX_RDB_SIZE = 65535;
MAX_WORDS = MAX_CHARS DIV 4;
SCHEMA_NAME_SIZE = 16;

%PRINT OFF
(* %INCLUDE SCETYP *)
(*-------------------------------------------------------------------*)
(* INCLUDES FOR THE SCHEMA EXECUTIVE *)
(*-------------------------------------------------------------------*)

%INCLUDE SCECON;
%INCLUDE APLTYP;

(*-------------------------------------------------------------------*)
(* TYPES USED BY THE SCHEMA MANAGER PACKAGE *)
(*-------------------------------------------------------------------*)

TYPE

(*-------------------------------------------------------------------*)
(* ENTITY PRIMITIVE TYPES *)
(*-------------------------------------------------------------------*)

ENTITY_TYPE = (INT, REEL, STRING, LOGICAL, ARAY, LIS, SETT, ENUM, ENUMITEM, POINTER, STRUCT, DEF_TYP, CLASS, SUPERTYPE, ENTITY, FIELD, GBLFLD, SUBSCHEMA, SCHEMA, BACK_PATCH, UNRESOLVED);

(*-------------------------------------------------------------------*)
(* TRANSACTION TYPES *)
(*-------------------------------------------------------------------*)

TRANS_TYPE = (T_SUBSCMA, T_CLASS, T_ENTITY, T_GBLFLD, T_FIELD, T_STRUCT, T_ARRAY, (* T_LIST, T_SET, *) T_DEF_TYP, T_DEF_TYP2, T_ENUM, T_ENUMITEM, T_INTEGER, T_REAL, T_LOGICAL, T_STRING, T_POINTER, T_FIND_KEY, T_PASS_KEY, T_SUPERTYPE, T_SUPERTYPE2, T_END_SUPERTYPE, T_END_STRUCT, T_END_ENUM, T_END_POINTER, T_END_DEF_TYP, T_END_ENTITY, T_END_CLASS, T_END_SUBSCMA, T_END_GBLFLD, T_END_PROCESSING, T_UNRESOLVED);

(*-------------------------------------------------------------------*)
(* NAME TYPE FOR THE ENTITIES *)
(*-------------------------------------------------------------------*)

T_NAME = PACKED ARRAY(.1..16.) OF CHAR;
CHAR50 = PACKED ARRAY(.1..50.) OF CHAR;
CHAR150 = PACKED ARRAY(.1..150.) OF CHAR;
(*-----------------------------------------------*)
(* ENUMERATED TYPES FOR THE RECORD DECLARATIONS  *)
(*-----------------------------------------------*)

T_STATUS = (FROZEN,UNFRZN);
T_PURPS = (KEY, KOLE);
T_REQ = (REQ, OPT);
T_DEP = (DEP, IND);
F_KEY_REC = RECORD
  REC_KIND : INTEGER;
(* CASE REC_KIND OF  *)
END;

(*-----------------------------------------------*)
(* BACKPATCH RECORD TYPE                        *)
(*-----------------------------------------------*)

BACKPATCH_REC = RECORD
  NAME : T_NAME;
  KIND : INTEGER;
END;

(*-----------------------------------------------*)
(* UNRESOLVED RECORD TYPE                       *)
(*-----------------------------------------------*)

UNRESOLVED_REC = RECORD
  NAME : T_NAME;
  KIND : INTEGER;
END;

(*-----------------------------------------------*)
(* SCHEMA RECORD TYPE                           *)
(*-----------------------------------------------*)

SCHEMA_REC = RECORD
  NAME : T_NAME;
  VERSION : INTEGER;
  DATE: ARRAY(.1..9.) OF CHAR;
  STATUS : T_STATUS
END;
)(* SUBSCHEMA TYPE *)

SSCMARREC = RECORD
  NAME : T_NAME;
  COMMENT : CHAR150;
  ADDB_SIZE : INTEGER;
  PHYSICAL : BOOLEAN;
END;

(* CLASS RECORD TYPE *)

CLASSREC = RECORD
  NAME : T_NAME;
  KIND : INTEGER;
  COMMENT : CHAR150;
END;

(* ENTITY RECORD TYPE *)

ENTITYREC = RECORD
  NAME : T_NAME;
  KIND : INTEGER;
  ADDB_SIZE : INTEGER;
  COMMENT : CHAR150;
  PHYSICAL : BOOLEAN;
END;

(* FIELD RECORD TYPE *)

FIELDREC = RECORD
  NAME : T_NAME;
  POS : INTEGER;
  (* PURPS : T_PURPS; *)
  REQD : T_REQ;
  (* DEPND : T_DEP; *)
  COMMENT : CHAR50;
  CASE INTEGER OF
    0 : (OFFSET : INTEGER;
SIZE : INTEGER);
1: (POSITION : INTEGER);
2: (SELECTOR_OFFSET : INTEGER;
   TABLE_INX_POSITION : INTEGER);
END;

(*---------------------------------------------------------------*)
(* ARRAY_RECORD_TYPE *)
(*---------------------------------------------------------------*)

ARRAY_REC = RECORD
  LO_BOUND : INTEGER;
  HI_BOUND : INTEGER;
  MIN_OCCUR : INTEGER;
  ARRAY_TYPE : ENTITY_TYPE;
  END;

(*---------------------------------------------------------------*)
(* LIST_RECORD_TYPE *)
(*---------------------------------------------------------------*)

(* LIST_REC = RECORD
  MINIMUM : INTEGER;
  MAXIMUM : INTEGER;
  END; *)

(*---------------------------------------------------------------*)
(* SET_RECORD_TYPE *)
(*---------------------------------------------------------------*)

(* SET_REC = RECORD
  MINIMUM : INTEGER;
  MAXIMUM : INTEGER;
  END; *)

(*---------------------------------------------------------------*)
(* ENTITY_ATTRIBUTE_DATA_BLOCK_TYPE *)
(*---------------------------------------------------------------*)

ENTITY_ADB = RECORD
  ENT_KIND : INTEGER;
  LENGTH : INTEGER;
  SYSUSE : INTEGER;
  VERSION : INTEGER;
  SYS_IDENT : INTEGER;
  IDENT : INTEGER;
TYP : ENTITY_TYPE;
DUMMY : ARRAY(.1..3.) OF CHAR;
CASE TYP OF
INT : (I_PRECISION : INTEGER );
REEL : (R_PRECISION : INTEGER );
STRING : (S_LENGTH : INTEGER );
LOGICAL : ();
ARRAY : (ARY : ARAY_REC );
(* LIS : (LST : LIST_REC ); *)
(* SETT : (SETS : SET_REC ); *)
ENUM : ();
ENUMITM : (ENUMITM_NAME : T_NAME );
POINTR : ();
STRUC : ();
DEF_TYP : (DEF_TYP_NAME : T_NAME );
CLASS : (CLS : CLASS_REC );
SUPERTYPE : (SUPERTYPE_NAME : T_NAME );
ENTITY : (ENT : ENTITY_REC );
FIELD : (FLD : FIELD_REC );
GBLFLD : ();
SUBSCHEMA : (SSCMA : SSCMA_REC );
SCHEMA : (SCMA : SCMA_REC );
BACK_PATCH : (BACKPATCH : BACKPATCH_REC );
UNRESOLVED : (UNRESOLVE : UNRESOLVED_REC);
END;

ENTBLOCK = ENTITY_ADB;

/* TRANSACTION TYPE */

TRANSACTION = RECORD
T_TYP : TRANS_TYPE;
CASE T_TYP OF
T_SUBSCMA : (SSCMA : SSCMA_REC );
T_CLASS : (CLS : CLASS_REC );
T_SUPERTYPE,
T_SUPERTYPE2 : (SUPERTYPE_NAME : T_NAME );
T_ENTITY : (ENT : ENTITY_REC );
T_GBLFLD : ();
T_FIELD : (FLD : FIELD_REC );
T_STRUC : ();
T_ARRAY : (ARY : ARAY_REC );
(* T_LIST : (LST : LIST_REC ); *)
(* T_SET : (SETS : SET_REC ); *)

3-57
T_DEF_TYP,
T_DEF_TYP2,
T_END_DEF_TYP : (DEFTYP_NAME : T_NAME);
T_ENUM : ();
T_ENUMITEM : (ENUMITEM_NAME : T_NAME);
T_INTEGER : (I_PRECISION : INTEGER);
T_REAL : (R_PRECISION : INTEGER);
T_LOGICAL : ();
T_STRING : (S_LENGTH : INTEGER);
T_POINTER : ();
T_FIND_KEY : (FNDKEY : F_KEY_REC);
T_PASS_KEY : (PASSKEY : ENTKEY);
T_END_STRUCT : ();
T_END_ENUM : ();
T_END_POINTER : ();
T_END_ENTITY : ();
T_END_CLASS : ();
T_END_SUPERTYPE : ();
T_END_SUBSCMA : ();
T_END_GBLFLD : ();
T_UNRESOLVED : (UNRESOLVE : UNRESOLVED_REC);
END;

(*----------------------------------------------------------*)
(* TRANSACTION STACK TYPE *)
(*----------------------------------------------------------*)

TRANSPT = &T_STACK;

T_STACK = RECORD
    STACKPTR : TRANSPT;
    STACKDATA : TRANSACTION
END;

(*----------------------------------------------------------*)
(* KEY STACK TYPE *)
(*----------------------------------------------------------*)

KEYPT = &K_STACK;

K_STACK = RECORD
    K_STACKPTR : KEYPTR;
    KEY_DATA : LISTKEY
END;

(*----------------------------------------------------------*)
(* ERROR TYPES *)
(*----------------------------------------------------------* )
T_SCE_ERROR =
    (OK,
     DIALOG_FAILED,
     TRANSACTION_PROCESSING_ERROR,
     STACK_UNDERFLOW,
     MAS_ROUTINE_ERROR,
     UNKNOWN_TYPE,
     SIZE_OUT_OF_RANGE);

(*--------------------------------------------------------------------------*)
(* WARNING TYPES *)
(*--------------------------------------------------------------------------*)

T_SCE_WARNING =
    (OKW,
     INVALID_KEY_RETURNED,
     INVALID_CASE_OPTION,
     MAX_ARRAY_SIZE_EXCEEDED,
     CANNOT_UPDATE,
     NO_ARRAY_ENTITY,
     NOT_ENOUGH_ROOM);

(*--------------------------------------------------------------------------*)
(* ERROR RETURN CODE TYPES *)
(*--------------------------------------------------------------------------*)

RETURN_CODE = RECORD
    ERROR : T_SCE_ERROR;
    WARNING : T_SCE_WARNING;
END;

RET_REC = RECORD
    RC : RETURN_CODE;
    ROUT_NAME : STRING(8)
END;

(*--------------------------------------------------------------------------*)
(* CHARACTER STRING TYPES *)
(*--------------------------------------------------------------------------*)

CHAR8  = PACKED ARRAY(.1..8.) OF CHAR;
CHAR9  = PACKED ARRAY(.1..9.) OF CHAR;
CHAR12 = PACKED ARRAY(.1..12.) OF CHAR;
CHAR16 = PACKED ARRAY(.1..16.) OF CHAR;
CHAR23 = PACKED ARRAY(.1..23.) OF CHAR;
CHAR30 = PACKED ARRAY(.1..30.) OF CHAR;
CHAR40 = PACKED ARRAY(.1..40.) OF CHAR;
IDCHAR = PACKED ARRAY (.1..IDENTIFIER_LENGTH.) OF CHAR;
TABCHAR = PACKED ARRAY (.1..TABLE_VARIABLE_LENGTH.) OF CHAR;
CRTABCHAR = PACKED ARRAY (.1..40.) OF CHAR;

(*-------------------------------------------------------------------*)
(*  ARRAY OF ENTITY TYPES                                           *)
(*-------------------------------------------------------------------*)

T_ARRAY16 = ARRAY(.1..MAX_ARRAY_SIZE.) OF CHAR16;
T_ARRAY23 = ARRAY(.1..MAX_ARRAY_SIZE.) OF CHAR23;
T_ARRAYID = ARRAY(.1..MAX_ARRAY_SIZE.) OF IDCHAR;
T_ARRAYTV = ARRAY(.1..MAX_ARRAY_SIZE.) OF TABCHAR;
T_ARRAYRV = ARRAY(.1..MAX_ARRAY_SIZE.) OF CRTABCHAR;

(*-------------------------------------------------------------------*)
(*  MESSAGE TYPE                                                    *)
(*-------------------------------------------------------------------*)

MESSAGE = CHAR8;

(*-------------------------------------------------------------------*)
(*  OPERATION TYPE                                                  *)
(*-------------------------------------------------------------------*)

OPERATIONS = (EDIT,REPORT,S_FILE,EXIT,S_RETURN,CR_SUB,CR_CLASS,
              CR_SUPR,CR_ENT,CR_DEF,CR_GBLFLD,UP_SUB,UP_CLASS,UP_ENT,
              UP_SUPR,UP_GBLFLD,UP_LOCAL,REVIEW,ACCEPT,DISPLAY,
              LIST,NO_MORE,INCLDS,DATA_DIC,RUNTIME,
              CONCEPTUAL_SCHEMA,CROSSREFER,UNDEFINED,
              ADD,REMOVE,DELETE,UPDATE,MULTIPLE_SEL,SAVE,
              RETRIEVE,UP_DEF,REV_ENT,REV_DEF,REV_SUPR,REV_GBLFLD,
              REV_SUB,REV_CLASS,REV_LOCAL,DEFAULT,CR_OPTION_ONE,
              CR_OPTION_TWO,CR_OPTION_THREE,CR_OPTION_FOUR,
              CR_OPTION_FIVE,CR_OPTION_SIX,CR_OPTION_SEVEN,
              CR_OPTION_EIGHT,CR_OPTION_NINE,CR_OPTION_TEN,
              PHYSICAL_SUBSCHEMA);
(* FIELD TYPE INDICATES THE TYPE OF THE USER OF THE FIELD *)
T_FIELDTYPE = (ENTITE,GLOBAL,STRUCTURE,SUPR);

(* USED TO DEFINE THE CONTEXT OF A DATA TYPE CREATION *)
T_CONTEXT = RECORD
  BUILDING_STRUCTURE : BOOLEAN;
  BUILDING_ARRAY : BOOLEAN;
END;

(* FIELD DEFINITION TYPE *)
T_FIELD_DEF = (IN_ADB,IN_KEYBLOCK,IN_CNST_REC);

(* MATCH RECORD INDICATES IF THE NAME AND/OR KIND NUMBER IS IDENTICAL *)
MATCH = RECORD
  NAME : BOOLEAN;
  NUMBER : BOOLEAN
END;

(* WITHIN RECORD INDICATES IF RETURN OR EXIT WAS CHOOSEN WITHIN ANOTHER ROUTINE *)
WITHIN = RECORD
  ROUTINE : BOOLEAN;
  RET : BOOLEAN;
  XIT : BOOLEAN
END;

(* TYPES FOR THE CONCEPTUAL SCHEMA REPORT *)
PAGES = (CLASS_PAGE,
        DEFINED_TYPE_PAGE,
        SUPERTYPE_PAGE,
        ENTITY_PAGE,
        GLOBAL_FIELD_PAGE,
        SUBSCHEMA_PAGE);

HEADING_TYPE = (DEFINITION,
                INDEX);

PAGE_PTR = @PAGE_REC;

PAGE_REC = RECORD
    PAGE_NUMBER : INTEGER;
    NEXT_PAGE   : PAGE_PTR
END;

(*-----------------------------------------*)
(* 'PAGE_PTR' IS USED TO LINK TOGETHER THE PAGE NUMBER OF THOSE  *)
(* PAGES THAT BEGIN A NEW ENTITY, CLASS, OR SUBSCHEMA. WHEN THE   *)
(* INDICES ARE PRINTED OUT, THE CHAIN OF PAGE NUMBERS CREATED IS  *)
(* USED.                                                       *)
(*-------------------------------------------------------------------*)

(*-------------------------------------------------------------------*)
(* TYPES FOR THE RUN-TIME SUBSCHEMA                                  *)
(*-------------------------------------------------------------------*)

T_SCHEMA_NAME      = PACKED ARRAY (. 1..SCHEMA_NAME_SIZE .) OF CHAR;
T_ATTRIBUTE_VALUE  = ARRAY (. 1..MAX_ATTRIBUTE_VALUE .) OF CHAR;
T_HEX_BYTE         = PACKED 0..255;
T_HEX_WORD         = ARRAY (. 1..4 .) OF T_HEX_BYTE;
T_SELECTOR         = PACKED 0..9;
T_INTEGER_1        = PACKED -128..127;
T_INTEGER_2        = PACKED -32768..32767;
T_INTEGER_4        = MININT..MAXINT;
T_VALUE            = ARRAY(.1..MAX_ATTRIBUTE_VALUE.) OF CHAR;
T_WORD             = ARRAY(.1..4.) OF CHAR;

T_PS_RECORD        = RECORD
    NAME          : T_SCHEMA_NAME;
    CS_ORDER      : INTEGER;
    OFFSET        : INTEGER;
END;
T_PS_ORDER = ARRAY (1..100) OF T_PS_RECORD;

ENTPNTR = @ENTBLOCK;
LISTPNTR = @T_SYS_LIST;
RDBSIZE = PACKED 0..MAX_RDB_SIZE;

T_SYS_LIST = RECORD
  SIZE : LISTSIZE;
  LSTLNG : LISTSIZE;
  LIST : ARRAY (1..MAX_LIST) OF ENTKEY;
END;

T_SCH_INST_ENT = RECORD
  KIND : ORD_KIND;
  POSITION : LISTPSTN;
  NUM_GROUP : LISTPSTN;
  MIN_CNST : LISTPSTN;
END;

T_ENTITY_INDEX = ARRAY (1..1000) OF RECORD
  NAME : T_SCHEMA_NAME;
  KIND : INTEGER;
END;
3.6.2 Model Access Software Data Dictionary

The following members of the Model Access Software constitute a PASCAL Include file. These listings are provided on the following pages.

- **APLTYP** - Application type declarations for MAS interface routines
- **ENTBLK** - Definition of an application data block for an entity
- **MASKIND** - Constants for internal routines
- **MASTYP** - Constants and types for MAS interface routines
- **NDSDCL** - Constants and types for MAS internal routines
- **$PCMGIT** - MAS memory management type declaration
- **PCMGIT** - MAS memory management type declaration
- **SCHDCL** - Constants and types for MAS internal routines
- **SCHTYP** - Constants and types for MAS internal routines
- **TVERIFY** - Verifies common types
(* %INCLUDE APLTYP *)

(* APPLICATION TYPE DECLARATIONS FOR USING THE MAS PACKAGE. *)

(*---------------------------------------------------------------*)

TYPE

ANYKEY = INTEGER;
LISTKEY = ANYKEY;
ENTKEY = ANYKEY;
ORD_KIND = INTEGER;
EXT_RET_CODE = INTEGER;
LISTPSTN = INTEGER;
LISTINDX = INTEGER;
LISTSIZE = INTEGER;
ROUTINE = ARRAY(.1..8.) OF CHAR;
NAMTYP = PACKED ARRAY(.1..6.) OF CHAR;

(* END %INCLUDE APLTYP *)
(*) (ENTBLK) DEFINITION OF ENTITY BLOCK TYPE. (*)

%INCLUDE PRINTOFF

(**) (*) MAS VERSION 1 (*)
(*) PACKAGE: ENTITY PACKAGE. (*)
(**) (*) THE ENTITY BLOCK IS DEFINED SEPARATELY TO ALLOW THE UNLIKELY (*)
(*) APPLICATION STRUCTURE OF AN ENTDATA RECORD CONTAINING FIELDS OF *)
(*) TYPES DEFINED IN NUSDCL. IN SUCH A CASE THE APPLICATION SHOULD *)
(*) INCLUDE NUSDCL, THEN DEFINE ENTDATA, THEN INCLUDE ENTBLOCK. *)
(**) (*)

CONST
SYS_MRDGLG = 1;  (* MARK FOR DELETE FLAG BAU *)
SYS_PRCFLG = 2;  (* PROCESS FLAG *)
SYS_ABSFLG = 3;  (* ABSENT/PRESENT FLAG *)
SYS_APLFLG = 4;  (* APPLICATION FLAG *)

TYPE
SYSINDEX=1..4;
ENTBLOCK=RECORD
  KIND:ORD_KIND;
  SIZE:ENTSIZE;
  SYSUSE:ARRAY(.SYSINDEX.) OF BOOLEAN;
  DATA:ENTDATA;
END;

%PRINT ON
(* END %INCLUDE ENTBLK *)
(* %INCLUDE MASKIND. *)
(**)
(* FUNCTION *)
(* CONSTANT DECLARATIONS FOR MAS ENTITY TYPES *)
(* )
(* LANGUAGE *)
(* PASCAL *)
(* )
(* PACKAGE *)
(* CADD EMULATION PACKAGE *)
(* )
(* ARGUMENTS - NONE. *)
(* )
(* )
(* COMMENTS *)
(* 1001 - 1480 ARE CADD ENTITY TYPES *)
(* 1501 - 1508 ARE PDDI ENTITY TYPES (TOPOGRAPHICAL) *)
(* )
(* )
(**)

CONST

HEADER = 1000;
MAS_POINT = 1001;
MAS_LINE = 1002;
MAS_PLANE = 1003;
MAS_ARC = 1004;
MAS_CIRCLE = 1014;
MAS_PC_CURVE = 1005;
MAS_SPHERE = 1006;
MAS_CROSSHAIR = 1007;
MAS_CONIC = 1010;
MAS_ELLIPSE = 1020;
MAS_LOFT_MAT = 1051;
MAS_LOFT_FUN = 1052;
MAS_LOFT_SPC = 1053;
MAS_LOFT_LPA = 1054;
MAS_LOFT_FAC = 1055;
MAS_LOFT_COF = 1056;
MAS_PCPATCH = 1086;
MAS_BOUNDED_PLANE = 1087;
MAS_SURFACE = 1088;
MAS_GROUP = 1089;
MAS_TEXT = 1090;
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAS_COORDINATE_DIM</td>
<td>1180</td>
</tr>
<tr>
<td>MAS_CONVENTIONAL_DIM</td>
<td>1280</td>
</tr>
<tr>
<td>MAS_ANGULAR_DIM</td>
<td>1380</td>
</tr>
<tr>
<td>MAS_RADIUS_DIM</td>
<td>1480</td>
</tr>
<tr>
<td>MAS_VERTEX</td>
<td>1501</td>
</tr>
<tr>
<td>MAS_EDGE</td>
<td>1502</td>
</tr>
<tr>
<td>MAS_LOOP</td>
<td>1504</td>
</tr>
<tr>
<td>MAS_FACE</td>
<td>1505</td>
</tr>
<tr>
<td>MAS_SHELL</td>
<td>1507</td>
</tr>
<tr>
<td>MAS_OBJECT</td>
<td>1508</td>
</tr>
<tr>
<td>DELETE_OP</td>
<td>2010</td>
</tr>
<tr>
<td>REPLACE_OP</td>
<td>2020</td>
</tr>
<tr>
<td>CONNECT_OP</td>
<td>2030</td>
</tr>
<tr>
<td>DISCONNECT_OP</td>
<td>2040</td>
</tr>
<tr>
<td>REPLACE_ATTRIBUTE_OP</td>
<td>2050</td>
</tr>
<tr>
<td>PICK_ENTITY</td>
<td>2060</td>
</tr>
</tbody>
</table>

(* END %INCLUDE MASKIND. *)
(* (MASTYP) CONSTANTS AND TYPES USED IN THE MAS INTERFACE. *)
(* INCLUDE PRINTOFF *)
(* MAS VERSION 1 *)
(* PACKAGE: NETWORK PACKAGE. *)
(* ------------------------*)
(* CONST *)
(* ------------------------*)
(* THE NUMBER OF ROUTINES IN MAS/SCHEMA/NDX. *)
(* ------------------------*)
(* NUM_MAS_ROUTINES=71; *)
(* ------------------------*)
(* EACH INTERFACE ROUTINE MUST BE ASSOCIATED WITH A NUMBER FOR *)
(* STATISTICS GENERATION. *)
(* ------------------------*)

MAEUD_ID=1;
MAEGTK_ID=2;
MAECR_ID=3;
MADMP_ID=4;
MAINIT_ID=5;
MDMP_ID=6;
MAL_ID=7;
MALATC_ID=8;
MALNO_ID=9;
MALGTK_ID=10;
MAEXEQ_ID=11;
MALXEQ_ID=12;
MALK_ID=13;
MALFND_ID=14;
MAEU_ID=15;
MALD_ID=16;
MALRDE_ID=17;
MALCPY_ID=18;
MAEC_ID=19;
MAECI_ID=20;
MAEUI_ID=21;
MAESWT_ID=22;
MAESVL_ID=23;
MALKL_ID=24;
MAED_ID=25;
MAEDT_ID=26;
MALRMI_ID=27;
MALOR_ID=28;
MALRPL_ID=29;
MALAND_ID=30;
MALNOT_ID=31;
MAEA_ID=32;
MAEAII_ID=33;
MALSTF_ID = 34;
MALSTR_ID = 35;
MALRD_ID = 36;
MALDA_ID = 37;
MALDI_ID = 38;
MAEAIV_ID = 39;
MAKILL_ID = 40;
MAEUUK_ID = 41;
MAECIK_ID = 42;
MALINS_ID = 43;
MALREP_ID = 44;
MAEDI_ID = 45;
MAEDTI_ID = 46;
MAECTK_ID = 47;
MAEKND_ID = 48;
MAKXEQ_ID = 49;
MALN_ID = 50;
MAESWA_ID = 51;
MAEUSR_ID = 52;
MAEGKN_ID = 53;
MASMSZ_ID = 54;
MALOCK_ID = 55;
MAECXQID = 56;
MAECQY_ID = 71;

(***)

3-70
(* THE EXTERNAL RETURN CODE INITIALIZATION AND OK VALUE. *)
NORM_RC=0;

(* THE RANGE ON AN ENTITY. *)
MIN_EN=0;
MAX_EN=65535;

(* THE EXTERNAL NIL POINTER VALUE. *)
NIL_PTR=0;

TYPE
    LINE=TEXT;
    PGMNAME=PACKED ARRAY(1..8.) OF CHAR;
    ERRMSG=PACKED ARRAY(1..25.) OF CHAR;

(* EXTERNAL RETURN CODE. *)
EXTRET_CODE=INTEGER;

CONST
    OKRC=0;

(* MAXIMUM VALUE TO BE IN THE OVERFLOW REQUESTED SIZE ARRAY *)
MAX_OVRFLW=10;

(* COMMON AREA FOR FSTART AND FSTOP ID AND ON/OFF FLAG. *)

TYPE

COMMON = RECORD
  ID_FIELD: INTEGER;
  ID_FLAG: INTEGER;
  ID_ERR_PGM: INTEGER;
  ID_ERR_PGMNAME: PGMNAME;
  ID_ERR_CODE: INTEGER;
  ID_ERR_MESSAGE: ERRMSG;

  OVRFLW_COUNT: INTEGER;
  OVRFLW_ENTRY: ARRAY (.1..MAX_OVRFLW.) OF INTEGER;
END;

(**)
%PRINT ON
(* END %INCLUDE MASTYP *)
(* (NDSDCL) NETWORK DATA STRUCTURE TYPES AND CONSTANTS USED BOTH BY BOTH INTERNAL NDS ROUTINES AND EXTERNAL APPLICATIONS. *)

(*INCLUDE PRINTOFF

(*************** NETWORK PACKAGE. ***************)

(MAS VERSION 2)

PACKAGE: NETWORK PACKAGE.

CHANGE CONTROL:

02/11/85 MAS VER 2 B. A. ULMER
ADD CRBPNT TO T_INT_ROOT
ADD MAX_RDB_SIZE
ADD INCREMENT VALUES FOR CONSTITUENT READ BLOCK
ADD T_CNST_RDBLK AND CNST_RDB_ENTRY
ADD 2 NEW ERROR MESSAGES
ADD RDBEXIST TO T_INT_ITEM
DELETE ROOT FROM T_INT_ITEM
DELETE DRLIST FROM T_INT_ITEM
04/23/85 MAS VER 2 E. D. SHREVE
ADD DELTFLG TO T_APLLIST TO MARK LISTS FOR NONDELETE.
ADD PROCESS FLAG (MAPROB) FOR INTERNAL MAS USAGE
05/21/85 MAS VER 2 B. A. ULMER
ADD NO_LIST_CREATED WARNING TO LIST OF WARNINGS
ADD INVALID_FLAG_NAME TO LIST OF ERRORS
06/13/85 MAS VER 2 B. A. ULMER
ADD NAMTYP FOR PARAMETER FOR MAQURY ANS MAUPDT
07/11/85 MAS VER 2 B. A. ULMER
ADD NUM_WARN AND NUM_ERROR FOR CHANGE TO CNVRR AND MSTATUS COMMON
10/24/85 MAS VER 2 B. A. ULMER
ADD RTS_NOT_IN_WORKING_FORM AND SIZE_NOT_LARGE_ENOUGH TO LIST OF ERRORS
03/12/86 MAS VER 2 E. D. SHREVE
ADD MAPROB2 TO THE IIT FOR SORT DELETE ROUTINES
03/21/86 MAS VER 2 B. A. ULMER
ADD CORE_NOT_AVAILABLE, NOT_ENOUGH_CORE_FOR_INIT, AND ABSOLUTELY_NO_MORE_CORE TO ERRORS FOR PROCESSING OF THE "OUT OF SPACE" CONDITION
04/22/86 MAS VER 2 B. A. ULMER
ADD NEW WARNING MESSAGE FOR MAEDTS - EMPTY_MARK_LIST
AND CHANGED THE NUM_WARN TO 7
05/05/86 MAS VER 2 B. A. ULMER
ADD NEW WARNING CODES FOR HANDLING EMPTY OUTPUT LISTS
FOR MAEDTS AND ERROR CODES FOR MAINIT

*)

3-73
MAS VERSION 3

PACKAGE: NETWORK PACKAGE.

06/17/86  MAS VER 3  B. A. ULMER

CHANGE STRUCTURE OF IIT FOR NEW DELETE RULES - ADD TYPE
DEFINITIONS FOR THE SCHEMA INSTANCE COLLECTOR

06/18/86  MAS VER 3  B. A. ULMER

ADD NEW ERROR CODES FOR SETRULS ROUTINE - ADD MAX_GROUP

07/01/86  MAS VER 3  B. A. ULMER

CHANGE LSTNG - T_NDSGM DUE TO NDS NAME CHANGE CONFLICTS

07/22/86  MAS VER 3  B. A. ULMER

CHANGE T_NDS_ERROR - ERROR CODE

CANT_MARK_FOR_DELETE TO SCHEMA_ROOT_NIL - BUG FIX
CANT_MARK_FOR_DELETE TO SCHEMA_ROOT_NIL - BUG FIX

THAT DEALS WITH MAERST

-----------------------------------------------

CONST

-----------------------------------------------

THE NUMBER OF CHARACTERS PER WORD IS MACHINE DEPENDENT. THE
ASSUMPTION HAS BEEN MADE THAT AN INTEGRAL NUMBER OF CHARACTERS
WILL OVERLAY AN INTEGER.

CHARS_PER_WORD = 4;

-----------------------------------------------

THE MAXIMUM SIZE OF KIND IS THE MAXIMUM INTEGER.

-----------------------------------------------

3-74
MAX_ENT_KIND = MAXINT;
(\*
(*) LIST INCREMENTS ARE USED TO CONTROL THE MEMORY VS SPEED
(*) TRADEOFF FOR THE MANAGEMENT OF THEIR CORRESPONDING LIST TYPES.
(*) THEIR VALUE CORRESPONDS TO THE NUMBER OF ENTITIES ADDED TO THE
(*) LIST EACH TIME GROWTH IS REQUIRED AND THE NUMBER OF ENTITIES
(*) REMOVED WHEN LIST COMPRESSION IS APPLICABLE.
*)
(\*)------------------------------------------------------------------------------------
(\*) APPL_LIST_INCR = 10;
ITEM_LIST_INCR = 24;
USER_LIST_INCR = 2;
CNST_LIST_INCR = 2;
LIST_OF_ROOTS_INCR = 10;
LIST_OF_LISTS_INCR = 20;
STACK_OF_LISTS_INCR = 20;
INST.COL_CNST_INCR = 200;
LRG_LIST_INCR_1 = 20;
LRG_LIST_INCR_2 = 100;
LRG_LIST_INCR_3 = 200;
(*) IF LIST PROCESSING CHANGED.  (*)
(\*)------------------------------------------------------------------------------------
(\*) MAXGROUP = 8;
(*)------------------------------------------------------------------------------------
(\*) CRB_REC_LEN = 12;
CNST_RDBLK_INCR = 4;
(*)------------------------------------------------------------------------------------
(\*) MAX_ENT_SIZE = 800000;  (*16777215 40*)
MAX_ENT_WORDS = 200000;  (* 4194301 10*)
(* ------------- TO FORCE VARIABLES TO A HALFWORD OF STORAGE BAU *)

MAX_RDB_SIZE = 65535;

(* EACH ENTRY IN A LIST OCCUPIES ONE WORD. THE LIST ITSELF CONTAINS *)
(* 1 WORD OF OVERHEAD. THE NUMBER OF ENTITIES IN A LIST ARE LIMITED *)
(* TO THE NUMBER OF WORDS ADDRESSABLE - 1 OR (2**22)-1. *)

MAX_LIST_SIZE = 4194302;

%PAGE

TYPE
  SIGNED_INT2 = PACKED -32768..32767;
  UNSIGNED_INT2 = PACKED 0..65535;
  NATURAL2 = PACKED 1..65535;

RDBSIZE = PACKED 0..MAX_RDB_SIZE;

(* THE ORDER TYPE IS USED FOR THE RESULT OF THE APPLICATION DEFINED *)
(* ORDER FUNCTION. THIS FUNCTION IS USED AS INPUT BY THE PROCEDURES *)
(* THAT SORT LISTS. *)

T_ORDER = (N_LESS, N_EQUAL, N_GREATER);
ORD_KIND = 0..MAX_ENT_KIND;

CONST

NIL_KIND = 0;

(* NUMBER OF WARNINGS AND ERRORS BAU 7/11/85*)
NUM_WARN = -10;
NUM_ERROR = 43;
TYPE

(*-*)

(* INTERNAL RETURN CODES. *)

(*-*)

(*-*)

(***)

T_NDS_ERROR =

(OK, * 0 *)
BAD_ENT_KIND, (* 1 *)
INVALID_CREATE, (* 2 *)
CANT_CREATE_LIST, (* 3 *)
MAS_INIT_FAILED, (* 4 *)
INVALID_UPDATE, (* 5 *)
CANT_UPDATE_ENT, (* 6 *)
CANT_CREATE_ENT, (* 7 *)
CANT_VERIFY_CONNECT, (* 8 *)
INVALID_CONNECTION, (* 9 *)
CANT_CONNECT, (* 10 *)
ABSENT_INPUT, (* 11 *)
INVALID_GET, (* 12 *)
NDS_OP_COMPLETE, (* 13 *)
BAD_LIST_POSITION, (* 14 *)
MAXIMUM_LIST_SIZE, (* 15 *)
BAD_LIST_MOVE_COUNT, (* 16 *)
BAD_LIST_REFERENCE, (* 17 *)
BAD_ENT_KEY, (* 18 *)
DUPLICATE_SCH, (* 19 *)
DUMP_ERROR, (* 20 *)
BAD_ENT_SIZE, (* 21 *)
BAD_SCH_KIND, (* 22 *)
PROC_CODE_ERROR, (* 23 *)
PROC_OUT_OF_RANGE, (* 24 *)
NO_MATCH_FOUND, (* 25 *)
DUPS_NOT_REMOVED, (* 26 *)
INVALID_DELETE, (* 27 *)
BAD_ENTITY_ON_USER_LIST, (* 28 *)
BAD_DELETE_KEY, (* 29 *)
EMPTY_MODEL, (* 30 *)
ARG_OUT_OF_RANGE, (* 31 *)
INVALID_CRB_POSITION, (* 32 *)
CRB_ENTRY_NOT_FOUND, (* 33 *)
INVALID_FLAG_NAME, (* 34 *)
SCHEMA_ROOT_NIL, (* 35 *)
SIZE_NOT_LARGE_ENOUGH, (* 36 *)
RTS_NOT_IN_WORKING_FORM, (* 37 *)
CORE_NOT_AVAILABLE, (* 38 *)

3-77
NOT_ENOUGH_CORE_FOR_INIT, (* 39 *)
ABSOLUTELY_NO_MORE_CORE, (* 40 *)
MAINIT_ALREADY_DONE, (* 41 *)
RULE_DOES_NOT_MATCH, (* 42 *)
ENTITY_NOT_FOUND_LIST ); (* 43 *)

(**)
T_NDS_WARNING =
(OKW, (* 0 *)
NO_SUCH_SCH, (* 1 *)
PROC_WARNING_CODE, (* 2 *)
EMPTY_DELETE_LIST, (* 3 *)
EMPTY_EXCEPTION_LIST, (* 4 *)
END_OF_LIST, (* 5 *)
NO_LIST_CREATED, (* 6 *)
EMPTY_MARK_LIST, (* 7 *)
EMPTY_MARK_N_EXCEPTION, (* 8 *)
EMPTY_DELETE_N_EXCEPTION, (* 9 *)
EMPTY_MARK_N_DELETE); (* 10 *)

(**)
RETURN_CODE = RECORD
ERROR : T_NDS_ERROR;
WARNING : T_NDS_WARNING;
END;

(**)
RET_REC = RECORD
RC : RETURN_CODE; ROUT_NAME : STRING(8); 
END;

(*---------------------------------------------------------------------*)
(* NEW DELETE AND COMPRESS RULE STRUCTURES 6/17/86 *)
(*---------------------------------------------------------------------*)
(**)
T_RULEELMNTS = (COMPRESS, DELETE, USER_DELETE, CNST_DELETE);

(**)
T_RULE = SET OF T_RULEELMNTS;

(**)
T_GROUP = RECORD
  LAST_CNST : RDBSIZE;
  RULE : T_RULE;
END;

(**)
T_GROUP_ARRAY = ARRAY (.1..MAX_GROUP.) OF T_GROUP;

(*---------------------------------------------------------------------*)
(* ENTITIES ARE DYNAMIC OBJECTS REFERENCED BY APPLICATION KEYS *)
(* AND WHICH CAN BE REFERENCED BY LISTS. *)
(*---------------------------------------------------------------------*)

3-78
ENTITIES = (NIL_ENT, INT_ROOT, INT_ITEM, APPL_LIST);

(* List pointers are used to connect entities to system lists. *)
(* A system list can never be connected to more than one entity. *)

LISTPNTR = @T_SYS_LIST;

(* The system list position indicates relative position within a system list. It is not a component with the system list type because only system lists connected to application list entities have an associated position. *)

LISTPSN = 0..MAX_LIST_SIZE;

(* Allowable size of systems lists. *)

LISTSIZE = 0..MAX_LIST_SIZE;

(* List index for indexing into system lists. *)

LISTINDX = 1..MAX_LIST_SIZE;

(* Pointer to a T_ENTITY in the network data structure. *)

T_KEY = @T_ENTITY;
ENTKEY = RECORD
P : T_KEY;
END;

(* Size of user entity data in words. *)

ENTSIZE = 0..MAX_ENT_SIZE;
(* SIZE OF APPLICATION DATA IN CHARACTERS. *)
(*-------------------------------------------------------------------*)

ENTCHARSIZE = 0..MAX_ENT_SIZE;

(* MACHINE DEPENDENT INDEX TYPE FOR MOVING USER DATA AS CHARACTER ARRAYS. *)
(*-------------------------------------------------------------------*)

ENTCHARINDX = 1..MAX_ENT_SIZE;

(* MACHINE DEPENDENT INDEX TYPE FOR MOVING USER DATA AS WORDS IN SYSTEMS FORMAT. *)
(*-------------------------------------------------------------------*)

ENTINDX = 1..MAX_Ent_WORDS;

(* USER DATA CAN BE VIEWED AS ARRAY OF CHARACTERS OR INTEGERS. *)
(* POINTER TO DYNAMICALLY AlLOCATED SYSTEM DATA STORAGE AREAS. *)
(*-------------------------------------------------------------------*)

ENTPNTR = @ENTBLOCK;

%PAGE

(* NDS DYNAMIC OBJECTS. *)
(*-------------------------------------------------------------------*)

TYPE

(* POINTER TO CONSTITUENT READ BLOCK BAU*)
(*-------------------------------------------------------------------*)

CRBPNTR = @T_CNST_RDBLK;

(* EACH NDS HAS ONE AND ONLY ONE INTERNAL ROOT. *)
(*-------------------------------------------------------------------*)
T_INT_ROOT = RECORD
  ROOT : ENTKEY;
  SCH_ROOT : ENTKEY;
  CNST_RDBLK : CRBPNTR;
END;

(**)
(*----------------------------------------------------------*)
(* DIRECTED LIST TYPE FOR MAINTAINING DIRECTION AND POSITION IN LISTS*)
(*----------------------------------------------------------*)
(**)
LISTDIR = (FORWARD, REVERSE);
(**)
DIRLST = RECORD
  LIST : LISTPNTR;
  POSITION : LISTPSTN;
  DIRECTION : LISTDIR;
END;

(**)
(*----------------------------------------------------------*)
(* CONSTITUENT READ BLOCK FOR MAINTAINING "POSITION" AND "DIRECTION" *)
(* IN CONSTITUENT LISTS BAU*)
(*----------------------------------------------------------*)
(**)
CNST_RDB_ENTRY = RECORD
  ENT : ENTKEY;
  POSITION : LISTPSTN;
  DIRECTION : LISTDIR;
END;

(**)
T_CNST_RDBLK = RECORD
  SIZE : RDBSIZE;
  CRBLNG : RDBSIZE;
  CNST_RDBLKARY : ARRAY(.RDBSIZE.) OF CNST_RDB_ENTRY;
END;

(**)
(*----------------------------------------------------------*)
(* EACH NDS HAS ONE INTERNAL ITEM FOR EACH ENTBLOCK. *)
(*----------------------------------------------------------*)
(* T_INT_ITEM = RECORD
  RDBEXIST : BOOLEAN;
  MAPROB : BOOLEAN; ADDED 04/23/85 - MAS PROCESS FLAG
  MAPROB2 : BOOLEAN; ADDED 12/03/85 - FOR DELETE EDS
  USERS : LISTPNTR;
  CNSTS : LISTPNTR;
  ENPTR : ENTPNTR;
END; *)

(**)
**T_INT_ITEM**  = RECORD
  RDSEXIST : BOOLEAN;
  MAPROB  : BOOLEAN; (* ADDED 04/23/85 - MAS PROCESS FLAG *)
  MAPROB2 : BOOLEAN; (* ADDED 12/03/85 - FOR DELETE EDS *)
  COLL_ADB : ENTPNTR; (* ADDED 06/17/86 - FOR DELETE AND *)
  (* AND COMPRESS BAU *)
  USERS : LISTPNTR;
  CNSTS : LISTPNTR;
  ENPTR : ENTPNTR;
END;

**T_ENTITY**  = RECORD
  FORM : ENTITIES;
  CASE ENTITIES OF
    NIL_ENT : ();
    INT_ROOT : (IRT : T_INT_ROOT);
    INT_ITEM : (IIT : T_INT_ITEM);
    APPL_LIST : (APL : T_APPL_LIST);
END;

**T_SYS_LIST** = RECORD
  SIZE : LISTSIZE;
  LSTLNAM : LISTSIZE;
  LIST : ARRAY(.LISTINDX.) OF ENTKEY;
END;

**THE SYSTEM LIST IS A VARIABLE LENGTH DYNAMICALLY ALLOCATED**
**THE STRUCTURE CONTAINING A LIST OF ENTITY REFERENCES**
(*-----------------------------------------------*)
(* USER OBJECTS *)
(*-----------------------------------------------*)
(**)
TYPE
  ENTRANCE = RECORD
    LOW : ORD_KIND;
    HIGH : ORD_KIND;
  END;
(**)
NDS  = RECORD
  KEY : ENTKEY;
END;
(**)
LISTKEY = RECORD
  P : T_KEY;
END;
(**)
ANYKEY = RECORD
  CASE INTEGER OF
    0 : (ENTK : ENTKEY);
    1 : (LSTK : LISTKEY);
END;

%PAGE
TYPE
(**)
(* GLOBAL VARIABLE RECORD *)
(*-----------------------------------------------*)
(**)
  T_NDSGVM = RECORD
    LIST_OF_ROOTS : LISTPNTR;
    STACK_OF_LISTS : LISTPNTR; (* MAY NOT BE NEEDED IN FUTURE *)
    (* IF LIST PROCESSING CHANGED. *)
    MODEL_HEAP : T_KEY;
  END;

(*----------BAU 6/13------------------------------------------*)
(**)
  NAMTYP = PACKED ARRAY (.1..6.) OF CHAR;
%PRINT ON
(* END %INCLUDE NDSDCL *)
(*****************************************************************)
(* $PCMG'T - INCLUDE FILE TO DEFINE TYPES FOR SPACE MANAGEMENT PROCS *)
(*****************************************************************)

%PRINT OFF
CONST
  XCB_SIZE = 8;
  $CB_SIZE = 16;

TYPE
  XCBP = @XCB;
  $CBP = @$B

  XCB = RECORD
    FREE: XCBP;
    SIZE: INTEGER;
  END;

  $CB = RECORD
    NEXT: $CBP;
    BIGGEST: INTEGER;
    FIRST: XCB;
  END;

MEMCONV$ = RECORD
  CASE INTEGER OF
    0: (I: INTEGER);
    1: ($: $CBP);
    2: (X: XCBP);
  END;

T_$PCMGR = RECORD
  SIZE: INTEGER;
  PTR: $CBP;
END;

%PRINT ON
CONST
XCB_SIZE = 8;
$CB_SIZE = 16;
OSIZE = 32368; (* 16184 32368 8092 *)

TYPE
XCBP = @XCB;
$CBP = @$CB;

XCB = RECORD
  FREE: XCBP;
  SIZE: INTEGER;
END;

$CB = RECORD
  NEXT: $CBP;
  BIGGEST: INTEGER;
  FIRST: XCB;
END;

MEMCONV$ = RECORD
  CASE INTEGER OF
    0: (I: INTEGER);
    1: ($: $CBP);
    2: (X: XCBP);
  END;

T_$PCMGR = RECORD
  SIZE: INTEGER;
  PTR: $CBP;
  OVERFLOW: $CBP;
  OFLAG: BOOLEAN;
END;

%PRINT ON
(%INCLUDE PRINTOFF

(* MAS VERSION 1 *)
(* NDS SCHEMA OBJECTS *)
(* PACKAGE: SCHEMA PACKAGE. *)
(*--------------------------------*)

(**)
%INCLUDE NDSDCL
%INCLUDE SCHTYP

(* USER DATA CAN BE VIEWED AS ARRAY OF CHARACTERS OR INTEGERS. *)
(*--------------------------------*)

ENTDATA=RECORD
    CASE INTEGER OF
        0:(CHARS:PACKED ARRAY(.ENTCHARINDX.) OF CHAR);
        1:(INTS:ARRAY(.ENTINDEX.) OF INTEGER);
        2:(SCH_ROOTS:T_SCH_ROOT_ENT);
        3:(SCH_INSTS:T_SCH_INST_ENT);
        4:(SCH_CLS:T_SCH_CLS_ENT);
    END (* ENTDATA *);

%INCLUDE ENTBLK;
(**)
%PRINT ON
(* END %INCLUDE SCHDCL *)
(* (SCHTYP) NETWORK DATA STRUCTURE SCHEMA TYPES AND CONSTANTS *)
(*-------------------------------------------------------------------*)
%INCLUDE PRINTOFF
(*-------------------------------------------------------------------*)
(* MAS VERSION 1 *)
(* NDS SCHEMA OBJECTS *)
(* PACKAGE: SCHEMA PACKAGE. *)
(* MAS VERSION 2 *)
(* NDS SCHEMA OBJECTS *)
(* PACKAGE: SCHEMA PACKAGE. *)
(* CHANGED: B. A. ULMER (LJB) DATE: 09/04/85 *)
(* REASON: ADD TWO NEW DELETE RULES (REQUIRES USER(S) TO EXIST-*
(* REQUIRES CONSTITUENT(S) TO EXIST) *)
(* MAS VERSION 3 *)
(* NDS SCHEMA OBJECTS *)
(* PACKAGE: SCHEMA PACKAGE. *)
(* CHANGED: B. A. ULMER DATE: 06/18/86 *)
(* REASON: CHANGE STRUCTURE OF THE SCHEMA INSTANCE COLLECTOR SO AS *)
(* HANDLE NEW DELETE AND COMPRESS RULES *)
(*-------------------------------------------------------------------*)
(**)
CONST
(**)
(* THE SCHEMA FAST ARRAY MUST HAVE A FIXED NUMBER OF ENTRIES IN *)
(* PASCAL. *)
(*-------------------------------------------------------------------*)
(**)
MAX_SCH_FST_SIZE = 1;
(**)
TYPE
(**)
(* THE SCHEMA KIND IS ASSIGNED THE ORDINAL OF A SCALAR IN SCH_KIND. *)
(*-------------------------------------------------------------------*)
(**)
SCH_KIND=(NIL_SCH,SCH_ROOT,SCH_INST,SCH_CLASS);
(**)
(* THE SCHEMA FAST INDEX IS USED FOR INDEXING INTO THE SCHEMA FAST *)
(* ACCESS LIST. *)
(*-------------------------------------------------------------------*)
(**)
SCH_FST_INDEX = 1..MAX_SCH_FST_SIZE;
(**)

3-87
(* THE SCHEMA ROOT IS AN INTERNAL ITEM IN ENTITY FORMAT CONTAINING *)
(* 'USER' DATA REQUIRED TO SEARCH THE ENTITIES IN THE MODEL BY KIND. *)
(*----------------------------------------------------------------*)

(* T_SCH_PSTN ALLOWS SEQUENTIAL STYLE ACCESS OF ENTIRE SCHEMA IN THE *)
(* SAME MANNER AS SEQUENTIAL READ OF AN APPLICATIONS LIST. CONTAINS *)
(* POSITION OF CURRENT SCHEMA INSTANCE ENTITY WITHIN SCHEMA AND *)
(* POSITION OF CURRENT ENTITY WITHIN SCHEMA INSTANCE ENTITY.*)
(*----------------------------------------------------------------*)

T_SCH_PSTN = RECORD
    SCH_PSTN : LISTPSTN;
    ENT_PSTN : LISTPSTN;
END (* T_SCH_PSTN *);

(*----------------------------------------------------------------*)

(* ARRAY SIZES IN THE SCHEMA_ROOT DEFINE THE AREA OCCUPIED BY THE *)
(* FAST AND STANDARD ARRAYS AND THE PORTION OF THAT AREA CONTAINING *)
(* VALID DATA.*)
(*----------------------------------------------------------------*)

T_SCH_ARY_SIZES = RECORD
    FST_ARY_LEN : LISTSIZE;
    FST_ARY_USED_LEN : LISTSIZE;
    STD_ARY_LEN : LISTSIZE;
    STD_ARY_USED_LEN : LISTSIZE;
END (* T_SCH_ARY_SIZES *);

(*----------------------------------------------------------------*)

(* THE FAST ARRAY IS USED FOR A PARTIAL SEARCH FOR THOSE KINDS *)
(* DETERMINED TO BE ACCESSED MORE OFTEN THAN THE USUAL CASE.*)
(*----------------------------------------------------------------*)

T_SCH_FST_ENTRY = RECORD
    KIND : ORDKIND;
    STD_INDEX : LISTINDEX;
END (* T_SCH_FST_ENTRY *);

T_SCH_FST_ARY = ARRAY (.SCH_FST_INDEX.) OF T_SCH_FST_ENTRY;

(*----------------------------------------------------------------*)

(* THE STANDARD ARRAY CONTAINS EACH KIND IN THE SAME SEQUENCE AS ITS *)
(* CORRESPONDING SCHEMA APPEARS IN THE CONSTITUENT LIST OF THE *)
(* SCHEMA_ROOT.*)
(*----------------------------------------------------------------*)
T_SCH_STD_ENTRY = RECORD
  KIND : ORD_KIND;
END (* T_SCH_STD_ENTRY *);
T_SCH_STD_ARY = ARRAY (.LISTINDEX.) OF T_SCH_STD_ENTRY;

(*)
\[\text{THE SCHEMA_ROOT_ENT MAY BE HANDLED AS ANY OTHER ENTDATA BY ROUTINES USING THE STANDARD DEFINITION OF ENTDATA.}\]
(*)
T_SCH_ROOT_ENT = RECORD
  POSITION : T_SCH_PSTN;
  ARY_SIZES : T_SCH_ARY_SIZES;
  FST_ARY : T_SCH_FST_ARY;
  STD_ARY : T_SCH_STD_ARY;
END (* T_SCH_ROOT_ENT *);
SCHRPNTR = @T_SCH_ROOT_ENT;

(*)
\[\text{THE SCHEMA_INST_ENT MAY BE HANDLED AS ANY OTHER ENTDATA BY ROUTINES USING THE STANDARD DEFINITION OF ENTDATA.}\]
(*)
T_SCH_INST_ENT = RECORD
  KIND : ORD_KIND;
  POSITION : LISTSPTN;
  RULE_DEP : BOOLEAN;
  RULE_STRONG : BOOLEAN;
  RULE_REQ_USER : BOOLEAN;
  RULE_REQ_CNST : BOOLEAN;
END; T_SCH_INST_ENT
SCHIPNTR = @T_SCH_INST_ENT;

(*)
\[\text{THE SCHEMA_CLS_ENT MAY BE HANDLED AS ANY OTHER ENTDATA BY ROUTINES USING THE STANDARD DEFINITION OF ENTDATA.}\]
(*)
T_SCH_CLS_ENT = RECORD
  KIND : ORD_KIND;
  POSITION : LISTSPTN;
  NUM_GROUP : LISTSPTN;
  MIN_CNST : LISTSPTN;
  GROUP : T_GROUP_ARRAY;
END (* T_SCH_CLS_ENT *);
SCHIPNTR = @T_SCH_CLS_ENT;

(*)
T_SCH_CLS_ENT = RECORD
    POSITION : LIST_PSN;
    KIND : ORDKIND;
END (* T_SCH_CLS_ENT *);
SCHCPNTR = @T_SCH_CLS_ENT;

(* )
CONST
(**)
    SCH_IN_MIN_SIZE = SIZEOF (T_SCH_INST_ENT);
    SCH_CL_MIN_SIZE = SIZEOF (T_SCH_CLS_ENT);
( )
(* THE SCHEMA ROOT CONTAINS A USER DATA BLOCK WHOSE SIZE DEPENDS 
   UPON THE NUMBER OF KINDS MODELED IN THE CURRENT SCHEMA. ITS 
   MINIMUM SIZE IS THE SIZE OF THE STATIC FIELDS PLUS THE MINIMUM 
   SIZE OF EACH DYNAMIC FIELD. ITS MAXIMUM SIZE IS THE SIZE OF THE 
   STATIC FIELDS PLUS THE MAXIMUM SIZE OF EACH DYNAMIC FIELD 
   MAX NUMBER OF T_SCH_INST_ENT THAT A T_SCH_CLASS_ENT CAN COLLECT 
)(* )
   SCH_RT_MIN_SIZE=SIZEOF(T_SCH_ROOT_ENT)-SIZEOF(T_SCH_STDARY);
   SCH_RT_MAX_SIZE=SIZEOF(T_SCH_ROOT_ENT);
( )
TYPE
    KIND_ARRAY=ARRAY(.I..MAXNUMKIND.) OF ORDKIND;
%PRINT ON
(* END %INCLUDE SCHTYP *)
(* (TVERIFY) VERIFY COMMON TYPE. *)
%INCLUDE PRINTOFF
(*------------------------------------------------------------------------*)
(* MAS VERSION 1 *)
(*------------------------------------------------------------------------*)
(**)
TYPE
VERIFY_COMMON=RECORD
   UPDATE:BOOLEAN;
   CREATE:BOOLEAN;
   GET:BOOLEAN;
   CONNECT:BOOLEAN;
   DELETE:BOOLEAN;
END;
%PRINT ON
(* END %INCLUDE TVERIFY *)
3.6.3 Name/Value Interface Data Dictionary

This section provides the data structures for the Name/Value Interface (N/VI). The following index provides an alphabetical list of the entities.

DDTYPE - Data Dictionary constants and types
NVITYP - N/VI constants and types
RTSTYP - Run-Time Subschema constants and types
(** (DCTYP) DATA DICTIONARY CONSTANTS AND TYPES ****************************)
(** $CHANGE CONTROL: ****************************)
(** REVISION : 15 MAY 1987, M. H. CHO, DBMA ****************************)
(** ADDED T_ADB_RECORD AND T_CL_RECORD TO RETURN ****************************)
(** THE DEFINITIONS IN PHYSICAL SCHEMA ORDER ****************************)
(** ORIGINATED: 19 MARCH 1987, M. H. CHO, DBMA ****************************)
(** ****************************)

CONST
    CONTINUATION_FLAG = 'X';
    PHYSICAL_ORDER = 'P';
    MAX_ARRAY_REC = 100;

(* RETURN CODE VALUES: *)
    KIND_NOT_IN_DATA_DICTIONARY = 1;
    ACTUAL_SIZE_GT_SPACE_AVAIL = -1;

(* TYPE *)
    T_ENTITY_NAME = PACKED ARRAY (. 1..32 .) OF CHAR;
(* )
    T_INX_RECORD = PACKED ARRAY (. 1..80 .) OF CHAR;
(* )
    T_USER_ARRAY = ARRAY (. 1..100 .) OF
                  PACKED ARRAY (. 1..80 .) OF CHAR;
(* )
    T_BOUNDS = RECORD
              LBOUND : PACKED ARRAY (. 1..4 .) OF CHAR;
              UBOUND : PACKED ARRAY (. 1..4 .) OF CHAR;
              END;
(* )
    T_ADB_RECORD = RECORD
                   ADB_KEY : INTEGER;
                   POSITION : INTEGER;
                   NO_OF_REC : INTEGER;
                   END;
(* )
    T_CL_RECORD = RECORD
                   POSITION : INTEGER;
                   NO_OF_REC : INTEGER;
                   END;
(* )
    T_VARIANT_RECORD = RECORD
          CASE INTEGER OF
              0 : ( BASIC_RECORD : PACKED ARRAY (. 1..80 .) OF CHAR );
              1 : ( BOUNDS : ARRAY (. 1..9 .) OF T_BOUNDS );
              2 : ( C_FLAG : CHAR;
                     NO_OF_KINDS : PACKED ARRAY (. 1..2 .) OF CHAR;
          END;

3-93
KINDS : ARRAY (. 1..12 .) OF
        PACKED ARRAY (. 1..6 .) OF CHAR );
3 : ( E_FLAG : CHAR;
    NO_OF_VALUES : PACKED ARRAY (. 1..2 .) OF CHAR;
    VALUES : ARRAY (. 1..4 .) OF
              PACKED ARRAY (. 1..17 .) OF CHAR );
END;
(*
 (* T_ADB_ARRAY = ARRAY (. 1..MAX_ARRAY_REC .) OF T_ADB_RECORD;
 *)
 (* T_CL_ARRAY = ARRAY (. 1..MAX_ARRAY_REC .) OF T_CL_RECORD;
 *)
 (* T_FILE_VARIANT = FILE OF T_VARIANT_RECORD;
 *)
 (* T_INX_FILE = FILE OF T_INX_RECORD;
 *)
(* END %INCLUDE DDTYP ***********************************************)
(? (NVITYP) NAME/VALUE INTERFACE CONSTANTS AND TYPES **********************)

(**************************************************************)

(* $CHANGE CONTROL: *)

(* REVISION: 11 MARCH 1988, M. H. Choi, DBMA *)
(* ADDED DIMEN_COUNT TO T_DATAREC *)
(* DELETED NO_OF_DIMENSION FROM T_DATAREC *)
(* ADDED NO_OF_DIMEN TO T_NAME_FRAME *)

(* REVISION: 15 JULY 1987, M. H. Choi, DBMA *)
(* ADDED COMPARISON_VALUES *)

(* REVISION: 16 SEPTEMBER 1986, M. H. Choi, DBMA *)
(* CHANGED STRUCTURE OF THE SCHEMA INSTANCE COLLECTOR *)
(* BECAUSE STRUCTURE CHANGED IN MAS TO HANDLE NEW *)
(* DELETE AND COMPRESS RULES *)

(* REVISION: 12 SEPTEMBER 1986, M. H. Choi, DBMA *)
(* ADDED FIELD TO T_INT_ITEM FOR THE COLL_ADB AND *)
(* MAPROB2 BECAUSE T_INT_ITEM STRUCTURE CHANGED IN MAS*)

(* REVISION: 04 JUNE 1986, M. H. Choi, DBMA *)
(* ADDED RETURN_CODE VALUES FOR FAILED_IN_MAL AND *)
(* FAILURE. *)

(* ORIGIENATED: 13 MAY 1986, G. A. WHITE, DBMA *)

(* **************************************************************)

CONST

(* ARBITRARY SIZE PARAMETERS: *)

MAX_ARRAY = 100;
MAX_ATTRIBUTE_NAME = 1000;
MAX_ATTRIBUTE_VALUE = 1000;
MAX_CHARS = 1000;
MAX_FIXED_STRING = 132;
MAX_GROUP = 8;
MAX_LIST = 4000000;
MAX_RDB_SIZE = 65535;
MAX_WORDS = MAX_CHARS DIV 4;
SCHEMA_NAME_SIZE = 16;

(* RETURN CODE VALUES: *)

INVALID_ARRAY_ENTITY = 7;
INVALID_SCALAR_VALUE = 6;
FAILED_IN_MAL = 5;
FAILED_IN_MAEXEQ = 4;
NIL_ENTITY_KEY = 3;
ATTRIBUTE_NOT_IN_ENTITY = 2;
KIND_NOT_IN_RUNTIME_SUBSCHEMA = 1;
SUCCESS = 0;
WARNING = -1;
FAILURE = -2;

(* COMPARISON VALUES: *)
EQ = 1;
LT = 2;
GT = 3;
NE = 4;
LE = 5;
GE = 6;

(* ATTRIBUTE NAME STRING DELIMITERS: *)
END_OF_SEGMENT = '\.';
END_OF_STRING = '00'XC;

(* DIMENSION VALUE DELIMITERS: *)
BEGIN_OF_ARRAY = '('.
END_OF_ARRAY = ')';

TYPE
T_ATTRIBUTE_NAME = ARRAY(. 1..MAX_ATTRIBUTE_NAME .) OF CHAR;
T_DIMEN_VALUE = ARRAY(. 1..MAX_DIMENSION .) OF INTEGER;
T_DISPLAY_WORD = PACKED ARRAY(. 1..8 .) OF CHAR;
T_FIXED_STRING = PACKED ARRAY(. 1..MAX_FIXED_STRING .) OF CHAR;
T_HEX_BYTE = PACKED 0..255;
T_HEX_WORD = ARRAY(. 1..4 .) OF T_HEX_BYTE;
T_Word = ARRAY(. 1..4 .) OF Char;
T_LOCATION = (IN_ADB, IN_JL, IN_ENUMERATION, IN_STRUCTURE);
T_SCHEMA_NAME = PACKED ARRAY (. 1..SCHEMA_NAME_SIZE .) OF CHAR;
T_SELECTOR = PACKED 0..0;
T_INTEGER_1 = PACKED -128..127;
T_INTEGER_2 = PACKED -32768..32767;
T_VALUE = ARRAY(. 1..MAX_ATTRIBUTE_VALUE .) OF CHAR;
T_Array_Value = ARRAY(. 1..MAX_ATTRIBUTE_VALUE .) OF T_Word;

(* ROUTINE = PACKED ARRAY(. 1..8 .) OF CHAR; *)

(* ENTDATA = RECORD
CASE INTEGER OF
  0 : ( CHAR : PACKED ARRAY(. 1..MAX_CHARS .) OF CHAR);
  1 : ( WORDS : ARRAY(. 1..MAX_WORDS .) OF T_HEX_WORD );
END; *)

(* EXT_RET_CODE = INTEGER; *)

(* LISTINDEX = 0..MAX_LIST; *)

(* LISTPSTN = 0..MAX_LIST; *)

(* LISTSIZE = 0..MAX_LIST; *)

(* ORD_KIND = 0..MAXINT; *)

(* RDBSIZE = PACKED 0..MAX_RDB_SIZE; *)
T_RULE_ELMNTS = ( COMPRESS, DELETE, USER_DELETE, CNST_DELETE );

(*

T_RULE = SET OF T_RULE_ELMNTS;

(*

T_GROUP = RECORD
    LAST_CNST : RDBSIZE;
    RULE : T_RULE;
END;

(*

T_GROUP_ARRAY = ARRAY (. 1..MAX_GROUP .) OF T_GROUP;

(*

T_SCH_INST_ENT = RECORD
    KIND : ORD_KIND;
    POSITION : LISTPSTN;
    NUM_GROUP : LISTPSTN;
    MIN_CNST : LISTPSTN;
    GROUP : T_GROUP_ARRAY;
END;

(*

ENTBLOCK = RECORD
    KIND : ORD_KIND;
    SIZE : 0..MAX_CHARS;
    SYSUSE : ARRAY (. 1..4 .) OF BOOLEAN;
    DATA : ENTDATA;
END;

(*

ENTPNTR = @ENTBLOCK;

(*

LISTPNTR = @T_SYS_LIST;

(*

T_INT_ITEM = RECORD
    RDBEXIST : BOOLEAN;
    MAPROB : BOOLEAN;
    MAPROB2 : BOOLEAN;
    COLL_ADB : ENTPNTR;
    USERS : LISTPNTR;
    CNSTS : LISTPNTR;
    ENPTR : ENTPNTR;
END;

(*

ENTITIES = ( NIL_ENT, INT_ROOT, INT_ITEM, APPL_LIST );

(*

T_ENTITY = RECORD
    FORM : ENTITIES;
    IIT : T_INT_ITEM;
END;

(*

3-97
ANYKEY = RECORD
  P : @T_ENTITY;
END;
(*
  ENTKEY = ANYKEY;
  (*
  LISTKEY = ANYKEY;
  (*
  T_SYS_LIST = RECORD
  SIZE : LISTSIZE;
   LSTLNG : LISTSIZE;
   LIST : ARRAY (. 1..MAX_LIST .) OF ENTKEY;
END;
(*
T_ATTRIBUTE_VALUE = RECORD
  CASE INTEGER OF
    0 : ( AS_VARIANT : T_Value );
    1 : ( AS_INTEGER_1 : T_Integer_1 );
    2 : ( AS_INTEGER_2 : T_Integer_2 );
    3 : ( AS_INTEGER_4 : Integer );
    4 : ( AS_REAL_4 : SHORTREAL );
    5 : ( AS_REAL_8 : REAL );
    6 : ( AS_FIXED_STRING : T_FIXED_STRING );
    7 : ( AS_LOGICAL : BOOLEAN );
    8 : ( AS_ENUMERATION : T_Schema_Name );
    9 : ( AS_Word : Array (. 1..250 .) of T_HEX_WORD );
    10 : ( AS_Extkey : Entkey );
    11 : ( AS_Array_Word : T_Word );
END;
(*
T_DEFN_POINTER = @T_DEFN_FRAME;
T_DEFN_FRAME = RECORD
  NEXT : T_DEFN_POINTER;
  KIND : ORD_KIND;
  DATA_TYPE : INTEGER;
  CASE INTEGER OF
    1, 2, 3, 4 : ( OFFSET : INTEGER;
    SIZE : INTEGER );
    7, 8 : ( POSITION : INTEGER);
    5 : ( SELECTOR_OFFSET : INTEGER;
    TABLE_OFFSET : INTEGER );
END;
(*
T_NAME_POINTER = @T_NAME_FRAME;
T_NAME_FRAME = RECORD

3-98
NAME : T_SCHEMA_NAME;
No_of_Dimen : Integer;
NEXT : T_NAME_POINTER;
DEFN : T_DEFN_POINTER;
END;

(*
   T_DATAREC       = RECORD
      NAME_ROOT     : T_NAME_POINTER;
      LIST_ROOT     : LISTKEY;
      ATTRIBUTE_VALUE : T_Attribute_Value;
      DIMEN_VALUE   : T_DIMEN_VALUE;
      Dimen_Count   : INTEGER;
   END;
(*
   BLKDATA = T_DATAREC;
(*
(* END %INCLUDE NVITYP *****************************************************)
(* (RTSTYP) RUN-TIME SUBSCHEMA CONSTANTS AND TYPES
**********)

(* $CHANGE CONTROL: *)
(* REVISED: 11 MARCH 1988, M. H. CHOI, DBMA *)
(* ADDED TO_ARRAY TO T_VARIANT_POINTER *)
(* REVISED: 8 SEPTEMBER 1987, M. H. CHOI, DBMA *)
(* ADDED MINIMUM OCCURRENCE FIELD TO T_ATTRIBUTE *)
(* ORIGINATED: 17 SEPTEMBER 1986, M. H. CHOI, DBMA *)

(* ARBITRARY SIZE PARAMETERS: *)
MAX_ARRAY_POINTER = 100;
MAX_ATTRIBUTE = 100;
MAX ENUMERATION = 100;
MAX_ENUM_INDEX = 100;
MAX ARRAY_LIST = 100;
MAX ARRAY_INDEX = 100;
MAX CL_LIST = 100;
MAX CL_KINDS = 100;

TYPE
T_STRING_8 = PACKED ARRAY (.1..8.) OF CHAR;
T_DATA_TYPE = ( INTEGER_DT, REAL_DT, STRING_DT, LOGICAL_DT,
 ENUM_DT, PNTR_DT, ARRAY_DT );

(* T_ATTRIBUTE = RECORD *)
  NAME : T_SCHEMA_NAME;
  MIN OCC : INTEGER;
  DATA_TYPE : INTEGER;
  CASE INTEGER OF
    1, 2, 3, 4 : ( OFFSET : INTEGER;
                  SIZE : INTEGER);
    7, 8 : ( POSITION : INTEGER);
    5, 10 : ( SELECTOR_OFFSET : INTEGER;
               TABLE_INX_POSITION : INTEGER);
END;

(* T_ENUM_INDEX = ARRAY (.1..MAX_ENUM_INDEX .) OF RECORD *)
  NO_OF_ENTRIES : INTEGER;
  TABLE_INX_POSITION : INTEGER;
END;

(* T_ENUMERATION = ARRAY(.1..MAX_ENUMERATION.) OF T_SCHEMA_NAME; *)

(* T ARRAY_INDEX = ARRAY (.1..MAX ARRAY_INDEX .) OF RECORD *)
  NO_OF_DIMENS : INTEGER;
TABLE_INX_POSITION : INTEGER;
END;

(*

T_ARRAY_LIST = ARRAY (1..MAX_ARRAY_LIST) OF RECORD
  SIZE : INTEGER;
  LOW_BOUND : INTEGER;
END;
(*

T_CL_INDEX = ARRAY (1..MAX_CL_LIST) OF RECORD
  NO_OF_CL_KINDS : INTEGER;
  TABLE_INX_POSITION : INTEGER;
END;
(*

T_CL_KINDS = ARRAY(1..MAX_CL_KINDS) OF INTEGER;
(*

T_SCHEMA_POINTER = @T_SCHEMA;
T_SCHEMA = RECORD
  NAME : T_SCHEMA_NAME;
  KIND : INTEGER;
  ATTRIBUTE_COUNT : INTEGER;
  ENUM_INDEX_OFFSET : INTEGER;
  ENUM_VALUE_OFFSET : INTEGER;
  ARRAY_INDEX_OFFSET : INTEGER;
  ARRAY_LIST_OFFSET : INTEGER;
  CL_INDEX_OFFSET : INTEGER;
  CL_KINDS_OFFSET : INTEGER;
  ATTRIBUTE : ARRAY (1..MAX_ATTRIBUTE) OF T_ATTRIBUTE;
END;
(*

T_RUN_TIME_POINTER = @T_RUN_TIME;
T_RUN_TIME = RECORD
  ENTITY : T_SCHEMA;
  ENUM_INDEX : T_ENUM_INDEX;
  ENUM_VALUE : T_ENUMERATION;
  ARRAY_INDEX : T_ARRAY_INDEX;
  ARRAY_LIST : T_ARRAY_LIST;
  CL_INDEX : T_CL_INDEX;
  CL_KINDS : T_CL_KINDS;
END;
(*

T_KIND_ADB_POINTER = @T_KIND_ADB;
T_KIND_ADB = RECORD
  SYSTEM_AREA : T_SCH_INST_ENT;
  RUN_TIME : T_RUN_TIME;
END;
*)

3-101
T_ARRAY_LIST_COMPACTOR = RECORD
  TABLE : T_ARRAY_LIST;
  TABLE_SIZE : INTEGER;
END;

(*

T_ARRAY_INX_COMPACTOR = RECORD
  TABLE : T_ARRAY_INDEX;
  TABLE_SIZE : INTEGER;
END;

(*

T_ENUM_INX_COMPACTOR = RECORD
  TABLE : T_ENUM_INDEX;
  TABLE_SIZE : INTEGER;
END;

(*

T_ENUM_COMPACTOR = RECORD
  TABLE : T_ENUMERATION;
  TABLE_SIZE : INTEGER;
END;

(*

T_CL_INX_COMPACTOR = RECORD
  TABLE : T_CL_INDEX;
  TABLE_SIZE : INTEGER;
END;

(*

T_CL_KINDS_COMPACTOR = RECORD
  TABLE : T_CL_KINDS;
  TABLE_SIZE : INTEGER;
END;

(*

T_DATA_VALUE = ARRAY (. 1..MAX_ATTRIBUTE_VALUE .) OF CHAR;

(*

T_VARIANT_POINTER = RECORD
  CASE INTEGER OF
    0 : ( AS_ADD_SIZE : LISTPSTN );
    1 : ( AS_INTEGER : INTEGER );
    2 : ( TO_ATTRIBUTE_VALUE : @T_VALUE );
    3 : ( TO_ENTITY : ENTPNTR );
    4 : ( TO_ENUMERATION : @T_ENUMERATION );
    5 : ( TO_SCHEMA : T_SCHEMA_POINTER );
    6 : ( TO_SELECTOR : @T_SELECTOR );
    7 : ( TO_ENUM_INDEX : @T_ENUM_INDEX );
    8 : ( AS_RUN_TIME_POINTER : T_RUN_TIME_POINTER );
    9 : ( AS_DATA : @T_DATA_VALUE );
   10 : ( TO_ARRAY_INDEX : @T_ARRAY_INDEX );
   11 : ( TO_ARRAY_LIST : @T_ARRAY_LIST );
   12 : ( TO_CL_INDEX : @T_CL_INDEX );

3-102
13 : ( TO_CL_LIST : @T_CL_KINDS );
14 : ( TO_ENTKEY : ENTKEY );
15 : ( TO_ARRAY_VALUE : @T_Word );
16 : ( TO_CNSTKEY : LISTPTR );
17 : ( TO_INTEGER_1 : @T_INTEGER_1 );
18 : ( TO_INTEGER_2 : @T_INTEGER_2 );
19 : ( TO_INTEGER_4 : @INTEGER );
20 : ( TO_REAL_4 : @SHORTREAL );
21 : ( TO_REAL_8 : @REAL );
22 : ( TO_value : T_VALUE );
23 : ( TO_ARRAY : T_Word );

END;

(*
(*) CONST
MAX_BUFFER = SIZEOF ( T_RUN_TIME );
(*
(*)
(* END %INCLUDE RTSTYP ***********************************************)

3-103
3.6.4 **System Translator Data Dictionary**

This section provides the data structures for the GMAP System Translator. The following index provides a brief description of the data entries function. The entities are listed in alphabetic order.

**EDBDEF** – Contains the file structure for accessing the Working Form Data Dictionary

**PRINT** – Turns output print off
(* EDBDEF *)
(* CONTAINS THE FILE STRUCTURE FOR ACCESSING THE WORKING FORM *)
(* DATA DICTIONARY *)

TYPE

SCALAR_REC = RECORD
  DNUMB: INTEGER;
  DSCALAR: ARRAY (.1..10.) OF
            PACKED ARRAY (.1..16.) OF CHAR
END;

ENTITY_REC = RECORD
  CLDISP: INTEGER;
  DNLMB: INTEGER;
  DENTITY: ARRAY (.1..36.) OF INTEGER
END;

SUBENT_REC = RECORD
  CLDISP: INTEGER;
  DNUMB: INTEGER;
  DSUBENT: ARRAY (.1..36.) OF INTEGER
END;

D_ENTITY = RECORD
  DNAME: PACKED ARRAY (.1..16.) OF CHAR;
  CSORDR: INTEGER;
  DMIN: INTEGER;
  DMAX: INTEGER;
  DTYPE: INTEGER;
  DSIZE: INTEGER;
  DDISP: INTEGER;
  CASE DTYPE OF
    5: (DSCA: SCALAR_REC);
    7: (DENT: ENTITY_REC);
    8: (DSEN: SUBENT_REC)
END;

DICTYP = ARRAY (.1..45.) OF D_ENTITY;

3-105
(*PRINT OFF*)
3.7 Object Code Creation

This section addresses the physical characteristics of the GMAP system components. The required physical characteristics of the Schema Manager, MAS, and the System Translator are discussed in succession.

3.7.1 Schema Manager

The Schema Manager software should be developed in a portable, higher order language, that is applicable to structured top-down programming principles. The interactive interface to the Schema Manager may require the use of system dependent software for terminal dialog management. These functions are widely available as an operating system option (i.e., IBM SPF Dialog Manager, VAX Forms Management system).

3.7.2 Model Access Software

The MAS should be modular to make its integration into existing applications easier. It should also be developed in a portable, higher-order language, which is applicable to structured top-down programming principles, and which can be linked to existing FORTRAN applications. In addition, the MAS should contain a built-in dynamic memory management capability to provide for the WF model.

3.7.3 System Translator

The following physical characteristics are based upon the fact that the System Translator is a software package that must interface with the Working Form and the Exchange Format.

3.7.3.1 Use Portable High Order Language

The System Translator must be developed in a high order language which is readily transportable between dissimilar hardware systems and used with different existing applications. The language should support operation of the System Translator in either the interactive or batch mode. In addition, the System Translator should, to the greatest extent possible, use only standard features of the high order language and use few, if any, extensions, unless the extensions are readily available.

3.7.3.2 Use Model Access Software

The System Translator must be able to interface to software that can place and retrieve in the WF (based upon a functional requirement). This software is known as MAS. This interface must be well-defined and modularized so that applications can interface to the System Translator, or other translators can use portions of this software.
3.7.3.3 **Interface to Exchange Format**

The System Translator must be able to interface to the Exchange Format through the Working Form. The format uses files that are ASCII, or Binary on 1600 BPI unlabeled tapes.

3.7.3.4 **Interface to Native System**

The System Translator must be able to interface to the native system. The native system can provide data by whole models, entities by kind, or by one entity. Thus, the System Translator must be able to process whole models, entities by kind, or one entity at a time.

3.8 **Adaptation Data**

The software that makes the GMAP exchange system different from most other exchange systems is its use of the Model Access Software, and the Working Form format of the exchange file. The GMAP system converts the sequential Exchange Format file data to a non-character/direct access format for the native system translator. It has been shown that using this access method significantly decreases processing time over the sequential method.

The only requirement of an organization in using the GMAP system is developing an interface between their native system translator, and the Working Form model using the MAS.

3.9 **Detail Design Description**

This section provides the routine hierarchy of the GMAP software components. The minimum core requirements for this set of software is 1.0M plus the size of the model. For example, the IBIS blade model required approximately 900K bytes.
3.9.1 Schema Manager Hierarchy

This section provides a cross reference listing for the Schema Manager routines.

**Routine: Refers to:**

**ADDENUM**
- SCELAB
- ISPLNK
- ISPLNKID

**ADDFIELD**
- SCELAB
- ISPLNK
- ISPLNK9
- ISPLNK50
- ISPLNKID

**APPROVE**
- SCELAB
- ISPLNK
- ISPLNKC8
- ISPLNK9
- ISPLNK12
- ISPLNKID

**BATDVR**
- FILRTV
- BATERR
- BATRPT
- BLDCLS
- BLDSUB
- BLEXICAL
<table>
<thead>
<tr>
<th>Routine:</th>
<th>Refers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLRSTK</td>
<td></td>
</tr>
<tr>
<td>DEFCLS</td>
<td></td>
</tr>
<tr>
<td>DEFENT</td>
<td></td>
</tr>
<tr>
<td>DEFGBL</td>
<td></td>
</tr>
<tr>
<td>DEFSUB</td>
<td></td>
</tr>
<tr>
<td>DEFSUP</td>
<td></td>
</tr>
<tr>
<td>DEFTYP</td>
<td></td>
</tr>
<tr>
<td>BATERR</td>
<td>ERRMSG</td>
</tr>
<tr>
<td></td>
<td>LEXICAL</td>
</tr>
<tr>
<td>RESTRING</td>
<td>SCELAB</td>
</tr>
<tr>
<td></td>
<td>ISPLNK</td>
</tr>
<tr>
<td></td>
<td>ISPLNKC8</td>
</tr>
<tr>
<td></td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>D</td>
</tr>
<tr>
<td>REF SUP</td>
<td>SCELAB</td>
</tr>
<tr>
<td></td>
<td>ELDUNRES</td>
</tr>
<tr>
<td></td>
<td>DEFADD</td>
</tr>
<tr>
<td></td>
<td>SCPUSHTR</td>
</tr>
<tr>
<td></td>
<td>MAKXEQ</td>
</tr>
<tr>
<td>REENTITY</td>
<td>SCELAB</td>
</tr>
<tr>
<td></td>
<td>ISPLNK</td>
</tr>
<tr>
<td></td>
<td>ISPLNKC50</td>
</tr>
<tr>
<td></td>
<td>ISPLNKKID</td>
</tr>
</tbody>
</table>
Routine: Refers to:

REENUM
   SCELAB
   ISPLNK

Routine: Refers to:
PSTRSTC
   SCELAB
   MAEC
   MAECTK
   MAEGTK
   MALNO
   MALRD
   MALSTF
   SCXERRCK
   SCXERRCK

Routine: Refers to:
?? ??RSTC
   PSRADB
   PSRARRAY
   PSRCL
   PSRENUM

Routine: Refers to:
PSTRGPF
   SCELAB
   MALK
   MALNO
   MALRD

Routine: Refers to:
PSRENUM

Routine: Refers to:
PSREPORT
   SCELAB
   MAEC
   MAECTK
   MALK
   MALRD
   MALSRT
   MALSTF
   PHYSICAL
   PSORDER
   PSRADB
Routine: Refers to:

CPCI
RSCPCT
RCPEI
RSCPET
PSMASKND
PSTRGF
PSTRSM
PSTRST
SCXERRCK

TF
PHBYFPOS
PHPOSITN
PHWOPPOS
SCXERRCK
SCXERRCK
SCERRCK
SCERRCK
SCERRCK
SCERRCK
SCEXIT

Routine: Refers to:

PHGTST
SCELAB
MALKL
MALRD
MALSTF

Routine: Refers to:

PHBYFPOS
SCELAB
MAEGTK
PHALFLD
Routine: Refers to:

PHGTFLD
PHSRTFLD
PHSRTORD
SCEXIT

Routine: Refers to:

REPORT
SCELAB
ISPLNK

Routine: Refers to:

MREVIEW
SCELAB
ISPLNK

Routine: Refers to:

MUPDATE
SCELAB
MQGETVAL

Routine: Refers to:

MQGTDEFN
MQGETVAL

Routine: Refers to:

MQIAATT
MQCLMU
MQGETDEFN
MQNCLMU
MQNUSRMU

Routine: Refers to:

MMAIN
SCELAB
ISPLNK
Routine: Refers to:

MNEWMOD
  SCELAB
  ISPLNK

Routine: Refers to:

EC
  SCELAB
  CLRSTK

Routine: Refers to:

GETDD

Routine: Refers to:

LEXICAL
  BATERR
  BLEXICAL
  CLRSTK
  DEFATT
  ERRREC
  REFSUP

Routine: Refers to:

DEFTYP
  SCELAB

TC
  MALD
  MALFND
  MALK
  MALKL
  MALN
  MALRD
  MALRDE
  MALREP
  MALRFL
  MALSTF

CLRSTK
  DEFATT
  ERRREC
  REFSUP
Routine:  Refers to:

DEFGBL
SCELAB
SCPUSHTR
BSCTRSPr

ELAB
SCEXIT
SCPUSHTR
BSCTRSPr
SCUNQEST
SCUNQPND
BATERR
BDUNRES
BLEXICAL
CLRSTK
DEFADD

STF

Routine:  Refers to:

DEFARR
SCELAB
SCPUSHTR
BATERR
BLEXICAL
CLRSTK
DEFBAS

RT
MALSTF
RSGTSM
SCXERRCK
SCXERRCK
SCEXIT

Routine:  Refers to:

DDENUM

3-115
Routine: Refers to:
CSLOGWRT
  CSPTRWRT
  CSRELWRT
  CSSTGWRT
  CSSTRWRT
  MAEGTK
  MALNO
  MALRD
  MALSTF

Routine: Refers to:
ALSTF

Routine: Refers to:
CSSUBHDG
  CSNEWPG

Routine: Refers to:
CSSUBWRT
  SCELAB
  MAEGTK
  MALNO
  MALRD
  MALSTF

Routine: Refers to:
CSRELWRT
  SCELAB
  MAEGTK
Routine: Refers to:

CSINDWRT
SCELAB
CSCLSHDG
CSENTHDG
CSSUBHDG
CSSUPHDG
MAEC
MAEGTK
MALNO
MALRD
MALSTF

Routine: Refers to:

SARYWRT
CSENTHDG
CSINTWRT
CSLOGWRT
CSPTWRWRT
CSRELWRT
CSSTGURT
MAEGTK
MALD
MALKL
MALNO
MALRD
MALSTF
MALRD
MALSTF

Routine: Refers to:

CSCLSHDG
CSNEWPG

Routine: Refers to:

CSCLSWRT
SCELAB

K
Routine: Refers to:

CRSET
SCELAB
ISPLNK

CRSTRING

LNK
ISPLNKID

CRENTITY
SCELAB
ISPLNK
ISPLNKID
ISPLNK50

BLDSUPER
BLDUNRES
DEFADD
DEFQUERY
SCPOPKE
SCPOPTR
SCPUSHTR
SCPUSHKE
MAL
MALATC
MALD
MALFND

SCELAB
MAECR

BLDUNRES
SCELAB
MAECR

3-118
Routine: Refers to:

BLDREAL
SCELAB
MAECK
MAEGTK
MALD
MALK
MALNO
MALRD

ENT
SCELAB
MAECK

Routine: Refers to:

BLDENUMR
SCELAB
MAECK

Routine: Refers to:

AECR
MAEGTK
MALD
MALK
MALNO
MALRD
MALSTF

Routine: Refers to:

BLDBPDEF
DEFENT
DEFGBL
DEFSUB
DEFSUP
DEFTYP

Routine: Refers to:

BATERR
ERRMSG
LEXICAL
### 3.9.2 Model Access Software Hierarchy

This section provides a cross-reference listing for MAS routines. The Control Sections (CSECTs) that are referred to by a particular CSECT (routine) are provided.

<table>
<thead>
<tr>
<th>Routine:</th>
<th>Refers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCRBM</td>
<td>NEWCRB</td>
</tr>
<tr>
<td></td>
<td>EXPCRB</td>
</tr>
<tr>
<td></td>
<td>EXCRBE</td>
</tr>
<tr>
<td>ADRLSM</td>
<td>NEWLSM</td>
</tr>
<tr>
<td></td>
<td>LSTMXLNM</td>
</tr>
<tr>
<td></td>
<td>LSTLNM</td>
</tr>
<tr>
<td></td>
<td>DISPLSM</td>
</tr>
<tr>
<td></td>
<td>MOVRLSM</td>
</tr>
<tr>
<td>ADSCH</td>
<td>FDSCH</td>
</tr>
<tr>
<td></td>
<td>NEWNSI</td>
</tr>
<tr>
<td></td>
<td>CRURUL</td>
</tr>
<tr>
<td></td>
<td>ADSCHR</td>
</tr>
<tr>
<td></td>
<td>ADTLSM</td>
</tr>
<tr>
<td>ADSCHR</td>
<td>FDSCH</td>
</tr>
<tr>
<td></td>
<td>ADRLSM</td>
</tr>
<tr>
<td></td>
<td>EXPSUDB</td>
</tr>
<tr>
<td>ADTLSM</td>
<td>NEWLSM</td>
</tr>
<tr>
<td></td>
<td>LSTMXLNM</td>
</tr>
<tr>
<td></td>
<td>LSTLNM</td>
</tr>
<tr>
<td></td>
<td>MOVRLSM</td>
</tr>
<tr>
<td></td>
<td>DISPLSM</td>
</tr>
</tbody>
</table>
Routine: Refers to:
ADTNM ADTLSM
Routine: Refers to:
BIGCREMM ADTLSM
Routine: Refers to:
CHKDEL ADTLSM ADTNM
       DELCNST INDLSM
       LSTLNM MSTART
       MSTOP RDLSM
       RSTLSM
       SETRULS
Routine: Refers to:
CHKTDEL ADTNM
       DETCNST
       DELRLSM
       MSTART
       MSTOP
       LSTLNM
       RDLSM
       RSTLSM
       SETRULS
Routine: Refers to:
CMPCRB
       MASNEW
       DISPCRB
Routine: Refers to:

CNNODM
MRKNM
NEWNM
ADTLSM
VERCN
RLSNM
CREMM
TVERIFY

Routine: Refers to:

CNNODMN
MRKNM
NEWNM
ADTLSM
VERCN
RLSNM
BIGCREMM

Routine: Refers to:

CNVOSP

Routine: Refers to:

CNVRR

Routine: Refers to:

CPYAUDB
AMFXMOVE
NEWSADB

Routine: Refers to:

CPYCST
LSTLNM
MRTLSM
NEWNM
FDSCH
Routine: Refers to:
CPYLSM
LSTLNM
NEWLSTM
MOVRLSTM

Routine: Refers to:
CPYNM
NEWM
RSTLSM
RDLSM
ADTLM

Routine: Refers to:
CRCLST
RSTLSM
RDLSM
CREMM

Routine: Refers to:
CRCLSTN
RSTLSM
RDLSM
BIGCREMM

Routine: Refers to:
CRCNM
CRCLST

Routine: Refers to:
CRDLST
ADTLM
DISPLSTM
DISPNM
EXPCLSTM
MRGTLSTM
NEWLSTM
NEWM
RSTSFLG
SORTDLSTM
Routine: Refers to:

CREMM
ADTLSM

Routine: Refers to:

CRURUL

Routine: Refers to:

DELCNST

CHKDEL
ADTLSM
ADTNM
INDLSM
INNM
MSTART
MSTOP
RDLSM
RSTLSM
SETRULS

Routine: Refers to:

DELCRBE

FNDCRBE
CMPCRB
EXCRBE

Routine: Refers to:

DELEMM

RSTLSM
RDLSM
DELRLSM
DISPPEMM

Routine: Refers to:

DELPLST

LSTMXLNM
LSTLNM
MOVRLSM
NEWLSM
DISPLSM
Routine: Refers to:

DELPNLA
- NEWLSM
- RDLSM
- LSTLNM
- LSTMXLNLM
- DISPLSM
- ADTLSM
- DISPEMM
- MOVRLSM
- MRGTLSM

Routine: Refers to:

DELRLSM
- LSTMXLNLM
- LSTLNM
- MOVRLSM
- NEWLSM
- DISPLSM

Routine: Refers to:

DELRUL
- ADTLSM
- CHKDEL
- CPFYLSM
- DELCNST
- DISPLSM
- ELDNM
- MRGTLSM
- NEWLSM
- NEWNM
- RDLSM
- RSTLSM
- XIEMM

Routine: Refers to:

DELSCH
- FDSCH
- INDLSTM
- DELRLSM
- DELPLST
- DISPEMM
Routine:  Refers to:

DETLSM  LSTLN
        LSTMXLN
        NEWLN
        MVRSL
        DISPPL

Routine:  Refers to:

DETCNST  ADTN
        CHKTD
        DELRL
        MSTART
        MSTOP
        RDLS
        RSTLN
        SETRUL

Routine:  Refers to:

DETRUL  ADTN
        CHKTD
        DETCNST
        DELRL
        DISP
        IN
        MRGT
        NW
        RSTLN
        RDLS

Routine:  Refers to:

DIFLSM  LSTLN
        CPYLS
        NEWLN
        INDLS

Routine:  Refers to:

DISPCRB  MASDSP
Routine: Refers to:
DISPEMM DISPLSM

Routine: Refers to:
DISPLSM

Routine: Refers to:
DISPNM
  DELRLSM
  DISPEMM
  RDLSM
  RSTLSM

Routine: Refers to:
ELDNM
  LSTLNM
  RSTLSM
  RDLSM
  ADTLSM
  DISPLSM

Routine: Refers to:
ELMNODM
  REVAADB

Routine: Refers to:
EXCRBE
  EXPCRB

Routine: Refers to:
EXPCLSM
  ADTLSM
  RDLSM
  RSTLSM
Routine: Refers to:
EXPCRB
MASNEW
DISPCRB

Routine: Refers to:
EXPSUDB
AMPXMOVE
NEWSADB

Routine: Refers to:
EXPULSM
ADTLSM
RDLMSM
RSTLSM

Routine: Refers to:
EXPULSMI
MRGTLSM
RDLMSM
RDRLSM
DELPILST
DISPLSM
LSTLNM
ADTLSM
CPYLSM

Routine: Refers to:
FDSCH
RDRLSM

Routine: Refers to:
FNDCRBE

Routine: Refers to:
FNDSKIND
RSTLSM
RDLMSM
Routine: Refers to:
GTGRBE

Routine: Refers to:
INDLSM

Routine: Refers to:
INITMGR

Routine: Refers to:
INNM

Routine: Refers to:
INTLSM

Routine: Refers to:
LSTLN

Routine: Refers to:
LSTMXLNM

Routine: Refers to:
MABRST

Routine: Refers to:
RDLSM
MSTART
MSTOP
CNVRR
CNVOSP
<table>
<thead>
<tr>
<th>Routine: Refers to:</th>
</tr>
</thead>
</table>
| MACPDT            | CNVRR  
|                   | CNVOSP  
|                   | NODECNN  
|                   | MRGTLSM  
|                   | MSTART  
|                   | MSTOP  
|                   | RDLSM  
|                   | RSTLSM  
<p>|</p>
<table>
<thead>
<tr>
<th>Routine: Refers to:</th>
</tr>
</thead>
</table>
| MAEA              | RDLSM  
|                   | RSTLSM  
|                   | MSTART  
|                   | MSTOP  
|                   | CNVRR  
|                   | CNVOSP  

3-130
Routine: Refers to:

MAEAI
CNVRR
CNVOSP
DISPNM
EXPCLSM
MSTART
MSTOP
NEWMN
RDLSM
RSTLSM

Routine: Refers to:

MAEAV
CNVRR
CNVOSP
MSTART
MSTOP

Routine: Refers to:

MAEC
CNVRR
CNVOSP
DISPNM
MRGTLSM
MSTART
MSTOP
NEWMN
NODECMN
RDLSM
RSTLSM
Routine: Refers to:

MAECI
  CNVRR
  CNVOSP
  DELRLSM
  DISPNM
  EXPCLSM
  MSTART
  MSTOP
  NEWNM
  RDLSTM
  RSTLSM

Routine: Refers to:

MAECIK
  ADTLSM
  CNVRR
  CNVOSP
  DISPNM
  EXPCLSM
  MSTART
  MSTOP
  NEWNM
  RDLSTM
  RSTLSM

Routine: Refers to:

MAECMP
  ADTLSM
  CNVRR
  CNVOSP
  DISPNM
  MSTART
  MSTOP
  NEWNM
  RDLSTM
  RSTLSM
  SETRULS
  TVERIFY
Routine: Refers to:

MAECQY
CNVRR
CNVOSP
MSTART
MSTOP
SETRULS
TVERIFY

Routine: Refers to:

MAECR
CNVRR
CNVOSP
CRCLST
MSTART
MSTOP
NEWNODE
TVERIFY
VERCR

Routine: Refers to:

MAECRN
CNVRR
CNVOSP
CRCLST
MSTART
NEWLSM
MSTOP
NEWNODE
VERCR

Routine: Refers to:

MAECTK
MSTART
MSTOP
CNVRR
CNVOSP
Routine: Refers to:

MAECXQ
ADCRBM
ADTLSM
CNVRR
CNVOSP
DELCRBE
DISPCRB
DISPNM
GTCRBE
FNDGRBE
LSTLNMM
MSTART
MSTOP
NFWM
RDLSM
RDRLSM
RSTLSM
TVERIFY
UPDGRBE

Routine: Refers to:

MAED
CNVRR
CNVOSP
CPYNM
DELRUL
DISPLSM
DISPNM
ELDNM
LSTLNMM
MSTART
MSTOP
NEWLSM
NEWNM
RDLSM
RDRLSM
RSTLSM
SORTDLST
TVERIFY
VERDEL
Routine: Refers to:

MAEDI
CNVRR
CNVOSP
CRDLST
DELRUL
DISPLSM
DISPNM
LSTLNM
MSTART
MSTOP
NEWLSM
NEWNM
RDLSM
RDRLSM
RSTLSM
TVERIFY
VERDEL

Routine: Refers to:

MAEDT
CPYNM
DETRUL
DISPNM
ELDNM
LSTLNM
MRGTMN
MSTART
MSTOP
NEWNM
RDLSM
RDRLSM
RSTLSM
SORTDLST
TVERIFY
VERDEL
CNVRR
CNVOSP
Routine: Refers to:

MAEDTI
   CNVRR
   CNVOSP
   CRDLST
   LSTLNM
   DISPLSM
   DISPNM
   DETRUL
   MRGTNM
   MSTART
   MSTOP
   NEWNM
   RDLSM
   RDRLSM
   RSTLSM
   TVERIFY
   VERDEL

Routine: Refers to:

MAEDTS
    CPYNM
    DETRUL
    DISPNM
    ELDNM
    LSTLNM
    MRGTNM
    MSTART
    MSTOP
    NEWNM
    RDLSM
    RDRLSM
    RSTLSM
    SORTDLST
    TVERIFY
    VERDEL
    CNVRR
    CNVOSP
Routine: Refers to:

MAEGKN
  CNVRR
  CNVOSP
  MSTART
  MSTOP

Routine: Refers to:

MAEGTK
  CNVRR
  CNVOSP
  ELMNODM
  MSTART
  MSTOP
  TVERIFY
  VERGT

Routine: Refers to:

MAEKND
  MSTART
  MSTOP
  CNVRR
  CNVOSP

Routine: Refers to:

MAERST
  RSTLSM
  RDLSM
  MSTART
  MSTOP
  CNVKR
  CNVOSP
Routine: Refers to:

MAESCI
  CNVRR
  CNVOSP
  DISPLSM
  NEWNMM
  EXPCLSM
  MSTART
  MSTOP
  RDLSM
  RSTLSM

Routine: Refers to:

MAESVL
  MSTART
  MSTOP
  CNVRR
  CNVOSP

Routine: Refers to:

MAESWA
  RSTLSM
  RDLSM
  MSTART
  MSTOP
  CNVRR
  CNVOSP

Routine: Refers to:

MAESWT
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  RDLSM
  RSTLSM

3-138
Routine: Refers to:

MAEU
CNVRR
CNVOSP
DISPNM
MRGTLSM
MSTART
MSTOP
NEWNM
NODEUNM
RDLSM
RSTLSM

Routine: Refers to:

MAEUD
CNVRR
CNVOSP
MSTART
MSTOP
REVNODM
TVERIFY
VERUD

Routine: Refers to:

MAEUI
CNVRR
CNVOSP
DELRLSM
DISPNM
EXPULSM
MSTART
MSTOP
NEWNM
RDLSM
RSTLSM

3-139
Routine: Refers to:

MAEUIK
  ADTLSM
  CNVOSP
  CNVRR
  DISPNM
  EXPULSM
  MSTART
  MSTOP
  NEWNM
  RDLASM
  RSTLSM

Routine: Refers to:

MAEUSR
  CNVRR
  CNVOSP
  LSTLNM
  MSTART
  MSTOP

Routine: Refers to:

MAEUXQ
  ADTLSM
  CNVRR
  CNVOSP
  DISPNM
  INDLASM
  LSTLNM
  MSTART
  MSTOP
  NEWNM
  RDLASM
  RDRLSM
  RSTLSM
  TVERIFY
Routine: Refers to:

MAEXEQ
    CNVRR
    CNVOSP
    LSTLNRM
    MSTART
    MSTOP
    RDLSM
    RDRLSM
    RSTLSM
    TVERIFY

Routine: Refers to:

MAINIT
    CNVRR
    CNVOSP
    MSTART
    MSTOP
    NEWNDM
    OSTART
    TVERIFY

Routine: Refers to:

MAKCNT
    FDSCH
    CNVRR
    CNVOSP
    MSTART
    MSTOP

Routine: Refers to:

MAKILL
    MSTART
    NDSRML
    CNVRR
    CNVOSP
    MSTOP

3-141
Routine: Refers to:

MAKXEQ
CNVRR
CNVOSP
MSTART
MSTOP
FDSCH
LSTLNK
RDRLSM
TVERIFY

Routine: Refers to:

MAL
CNVRR
CNVOSP
MSTART
MSTOP
NEWNM

Routine: Refers to:

MALAND
CNVRR
CNVOSP
DISPNM
INTLSM
MSTART
MSTOP
NEWNM

Routine: Refers to:

MALATC
ADTNM
CNVRR
CNVOSP
CRCNM
CREMM
MRGTMN
MSTART
MSTOP
TVERIFY
VERAPN
Routine: Refers to:
MALCPY
CNVRR
CNVOSP
CPYNM
MSTART
MSTOP

Routine: Refers to:
MALD
CNVRR
CNVOSP
DELRLSLM
DISPEMM
MSTART
MSTOP
RDLSM
RSTLSM

Routine: Refers to:
MALDA
RDTLSLM
DELPNLAA
DISPLSLM
MSTART
MSTOP
CNVRR
CNVOSP

Routine: Refers to:
MALDI
CNVRR
CNVOSP
INDLSM
MSTART
MSTOP
RDTLSLM
DELPNLAA
DISPLSLM

3-143
Routine: Refers to:
MALFND
CNVRR
CNVOSP
MSTART
MSTOP
RDLSM

Routine: Refers to:
MALGTK
CNVRR
CNVOSP
MSTART
MSTOP
RDRLSM

Routine: Refers to:
MALINS
'DRLSM
ADTLSM
CNVRR
CNVOSP
FNDCRBE
GTCRBE
LSTLNM
MSTART
MSTOP
RDLSM
RSTLSM
UPDCRBE

Routine: Refers to:
MALK
CNVRR
CNVOSP
CPYCST
DISPNM
FDSCH
MSTART
MSTOP
NEWNM
Routine: Refers to:

MALKC
ADTLSM
CNVRR
CNVOSP
DISPLSM
INTLSM
FDSCH
NEWLMSM
MOVRLSM
MSTART
MSTOP
NEWWNM

Routine: Refers to:

MALKL
ADTNM
CNVRR
CNVOSP
FDSCH
FNDSKIND
MSTART
MSTOP
NEWWNM
RSTLSM
RDLSM

Routine: Refers to:

MALKU
CNVRR
CNVOSP
DISPLSM
FDSCH
INTLSM
LSTLNLM
MOVRLSM
NEWLMSM
MSTART
MSTOP
NEWWNM
RDLSM
Routine: Refers to:
MALN
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  NEWNM
  NEWLSM

Routine: Refers to:
MALNO
  CNVRR
  CNVOSP
  LSTLNLM
  MSTART
  MSTOP

Routine: Refers to:
MALNOT
  CNVRR
  CNVOSP
  DIFLSM
  DSPNLM
  MSTART
  MSTOP
  NEWNM

Routine: Refers to:
MALOCK
  CNVRR
  CNVOSP
  MSTART
  MSTOP
Routine: Refers to:
MALOR
    CNVVR
    CNVOSP
    DISPNM
    ELDNM
    MRGLSM
    MSTART
    MSTOP
    NEWNM

Routine: Refers to:
MALRD
    ADCRBM
    CNVRR
    CNVOSP
    DELCRBE
    DISPCRB
    FNDCRBE
    GTCRBE
    MSTART
    MSTOP
    RDRLSM
    UPDGRBE

Routine: Refers to:
MALRDE
    CNVRR
    CNVOSP
    ELDNM
    MSTART
    MSTOP
Routine: Refers to:

MALREP
  ADTLSM
  CNVRR
  CNVOSP
  CPYLSM
  DELCRBE
  DELRLSM
  DISPCRB
  FNDURUL
  LSTLNM
  MSTART
  MSTOP
  RDLSM
  RSTLSM

Routine: Refers to:

MALRMV
  CNVRR
  CNVOSP
  DELCRBE
  DELPLST
  DELRLSM
  DELRUL
  DISPCRB
  DISPNN
  DISPLSM
  FNDCRBE
  GTCRBE
  LSTLNM
  MSTART
  MSTOP
  NEWLSM
  NEWNM
  SETRULS
  UPDCRBE

Routine: Refers to:

MALROR
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  ORDRLST
  TVERIFY

3-148
Routine: Refers to:
MALRORI
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  ORDRLSTI
Routine: Refers to:
MALRPL
  ADTLSM
  CNVRR
  CNVOSP
  DELPLST
  DELRLSM
  DELRL
  DISPLSM
  DISPNM
  INDLSM
  LSTLN
  MSTART
  MSTOP
  NEWNM
  NEWLSM
  REVRLSM
Routine: Refers to:
MALRRI
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  ORDRLSTI
Routine: Refers to:
MALRST
  CNVRR
  CNVOSP
  MSTART
  MSTOP
Routine: Refers to:

MALRVS
  CNVRR
  CNVOSP
  RVRLSM
  DISPLSM
  MSTART
  MSTOP

Routine: Refers to:

MALSRT
  CNVRR
  CNVOSP
  SORTLSM
  MSTART
  MSTOP
  TVERIFY

Routine: Refers to:

MALSTF
  ADCRBM
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  UPDCRBE

Routine: Refers to:

MALSTR
  ADCRBM
  CNVRR
  CNVOSP
  LSTLNRM
  MSTART
  MSTOP
  UPDCRBE
Routine: Refers to:

MALXEQ
  ADCRBM
  ADTLSM
  CNVRR
  CNVOSP
  DELCRBE
  DISPCRBB
  DISPNUM
  FNDRCRBE
  GTCRBE
  LSTLNMM
  MSTART
  MSTOP
  NEWNM
  RDLSM
  RDLRSLM
  RSTLSM
  TVERIFY
  UPDCRBE

Routine: Refers to:

MAQURY
  CNVRR
  CNVOSP
  MSTART
  MSTOP

Routine: Refers to:

MASALOC  *

Routine: Refers to:

MARDLT
  CNVRR
  CNVOSP
  FDSCH
  MSTART
  MSTOP
  REVAAADB
  REVSAADB
  RDLRSLM
  RSTLSM
  MASDSP
Routine: Refers to:

MARSCT
  CNVRR
  CNVOSP
  FDSCH
  MSTART
  MSTOP
  REVAADB

Routine: Refers to:

MASDSP

Routine: Refers to:

MASMSZ
  MSTART
  NDSFCT
  MSTOP
  CNVRR
  CNVOSP

Routine: Refers to:

MASNEW

Routine: Refers to:

MASOVR

Routine: Refers to:

MAUPDT
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  RDLSM
  RSTLSM
Routine: Refers to:

MIDBD
  CNVOSP
  CNVRR
  XIEMM
  ELDNM
  MSTART
  MSTOP
  RDLSM
  RSTLSM

Routine: Refers to:

MIDERV
  ADTLSM
  CNVRR
  CNVOSP
  DELCRBE
  DELPLST
  DELRLSM
  DELENTY
  DELRUL
  DISPCRB
  DISPLSM
  FNDCRBE
  GTCRBE
  LSTLNM
  MSTART
  MSTOP
  NEWLSM
  UPDCRBE

Routine: Refers to:

MOVRLSM
  AMPXMOVE
  LSTLNM
  LSTMXLNM
Routine: Refers to:
MRGTLSM
  CPYLSM
  LSTMXLN
  LSTLN
  DISPLSM
  NEWLSM
  MOVRLSM

Routine: Refers to:
MRGTNM
  MRGTLSM

Routine: Refers to:
MRKNM
  ADTLSM
  NEWEMM
  NEWLSM

Routine: Refers to:
MRSCR
  CNVRR
  CNVOSP
  FDSCH
  MSTART
  MASNEW
  MASDSP
  MSTOP
  RSTLSM
  RDLSTM
  REVSAADB

Routine: Refers to:
MSINIT
  CNVRR
  CNVOSP
  MSTART
  MSTOP
  NEWNDM
  OSTART
Routine: Refers to:
MSTART
Routine: Refers to:
MSTOP
Routine: Refers to:
NDSCMM
Routine: Refers to:
NDSFCT
Routine: Refers to:
NDSCMB
Routine: Refers to:
NDSRML
Routine: Refers to:
NEWCRB
MASNEW
Routine: Refers to:
NEWEMM
Routine: Refers to:
NEWIIM
NEWEMM
NEWLSM
CPYAUDP
Routine: Refers to:
NEWLSM
Routine: Refers to:

NEWNDM
  NEWEMM
  ADTLSM
  NEWNSR

Routine: Refers to:

NEWNM
  NEWEMM
  NEWLSM
  RDTLSM
  ADTLSM

Routine: Refers to:

NEWNMM
  NEWEMM
  NEWLSM
  RDTLSM
  ADTLSM

Routine: Refers to:

NEWNODE
  NEWIIM
  ADSCH

Routine: Refers to:

NEWNSI
  NEWSCHI

Routine: Refers to:

NEWNSR
  NEWSCHR

Routine: Refers to:

NEWSADB
  AMPXNEW
Routine: Refers to:
NEWSCHI
    NEWIIM

Routine: Refers to:
NEWSCHR
    NEWIIM

Routine: Refers to:
NODECNM
    NEWNM
    CPYLSM

Routine: Refers to:
NODECNN
    NEWEMM
    CPYLSM
    LSTLNM

Routine: Refers to:
NODEUNM
    NEWNM
    CPYLSM

Routine: Refers to:
OCOUNT

Routine: Refers to:
ORDRLST
    CPYLSM
    DISPNM
    INNM
    LSTLNM
    NODEUNM
    RDSLM
    REVRLSM
    RSTLSM
Routine: Refers to:
ORDRLSTI
    ADTLSM
    AMPXMOVE
    DISPLSM
    MASNEW
    MASDSP
    MOVRLSM
    NEWLSM

Routine: Refers to:
OSTART

Routine: Refers to:
PASASM *

Routine: Refers to:
RDLSM
    LSTLNM

Routine: Refers to:
RDRLSM
    LSTMXLNM

Routine: Refers to:
RDTLSM
    LSTLNM

Routine: Refers to:
REVAADB
    AMPXMOVE

Routine: Refers to:
REVNODM
    REVSADB
Routine: Refers to:

REVRLSM
  LSTLNM

Routine: Refers to:

REVSA

  AMPXMOVE
  NEWSADB

Routine: Refers to:

RLSNM

  DELTLSM
  DISPEMM
  RDSL
  RDTLSM
  RSTLSM

Routine: Refers to:

RSTLSM

Routine: Refers to:

RSTSFLG

  RDSL
  RSTLSM

Routine: Refers to:

RVRLSM

  NEWLSM
  RDRSL
  ADTLSM
  LSTLNM

Routine: Refers to:

SETRULS

  INDLNM

3–159
Routine: Refers to:

SORTDLST
   RSTLSM
   RDLSM
   NEWLSM
   SRTBYCNT

Routine: Refers to:

SORTLSM
   LSTLNLM
   ELMNODM
   ELMNODM

Routine: Refers to:

SRTBYCNT
   RSTLSM
   RDLSM
   ADTLSM

Routine: Refers to:

UPDCRBE
   FNDCRBE

Routine: Refers to:

VERAPN

Routine: Refers to:

VERCN

Routine: Refers to:

VERCR

Routine: Refers to:

VERDEL
Routine: Refers to:
VERGT

Routine: Refers to:
VERUD

Routine: Refers to:
XIEMM

  DELCRBE
  DELSCH
  DELEMM
  DispCRB
  RDTLSM
  DELCRBE
  DELTLSM
  DELRLSM
  DispCRB
  FNDCRBE
  GTCRBE
  INDLSTM
  UPDCRBE
  RDTLSM
  DELTLSM
  DELRLSM
### 3.9.3 Name/Value Interface Hierarchy

This section provides a cross-reference listing for the N/VI.

<table>
<thead>
<tr>
<th>Routine</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADBLOCA</td>
<td></td>
</tr>
<tr>
<td>ENUMLOCA</td>
<td>ADBLOCA</td>
</tr>
<tr>
<td>GETDD</td>
<td></td>
</tr>
<tr>
<td>GETDDBN</td>
<td></td>
</tr>
<tr>
<td>NVCPATAV</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>NVCPAV</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>NVCRTE</td>
<td>NVGTAN</td>
</tr>
<tr>
<td>NVDLTM</td>
<td></td>
</tr>
<tr>
<td>NVDQAN</td>
<td>NVDQGTAV</td>
</tr>
</tbody>
</table>

3-162
Routine: Refers to:
NVDQARLO
  ADBLOCA
  NVDQARPT
  NVCPAV

Routine: Refers to:
NVDQARPT
  NVCPATAV

Routine: Refers to:
NVDQGTAV
  ADBLOCA
  ENUMLOCA
  NVDQARLO
  NVGTAN
  NVGTDD

Routine: Refers to:
NVDSARLO
  ADBLOCA
  NVDSARPT
  NVCPAV

Routine: Refers to:
NVDSARPT
  NVCPATAV

Routine: Refers to:
NVDSAV
  NVCPAV
  NVDSGTAV

Routine: Refers to:
NVDSENLO
  ADBLOCA
Routine: Refers to:
NVDSGTAV
ADBLOCA
NVDSARLO
NVDSENLO
NVGTAN
NVGTDD

Routine: Refers to:
NVEQAV
MALATC
NVLCAV

Routine: Refers to:
NVGEAV
MALATC
NVLCAV

Routine: Refers to:
NVGRAV
MALATC
NVLCAV

Routine: Refers to:
NVGTAN

Routine: Refers to:
NVGTAT
NVGTAN
NVGTDD

Routine: Refers to:
NVGTDD
MRSCH
MARSCHGT
RSGTDD

3-164
Routine: Refers to:
NVGTED GETDDBN

Routine: Refers to:
NVGTRS MRSCT MARSCT NVRTVRS

Routine: Refers to:
NVLCAV ADBLOCA ENUMLOCA NVPQARLO NVGTDD

Routine: Refers to:
NVLEAV MALATC NVLCAV

Routine: Refers to:
NVLTAV MALATC NVLCAV

Routine: Refers to:
NVNEAV MALATC NVLCAV

Routine: Refers to:
NVPQARLO ADBLOCA NVPQARPT NVCPATAV
<table>
<thead>
<tr>
<th>Routine</th>
<th>Refers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVPQARPT</td>
<td>NVCPATAV</td>
</tr>
<tr>
<td>NVPQAV</td>
<td>MAL</td>
</tr>
<tr>
<td></td>
<td>NVCRTM</td>
</tr>
<tr>
<td></td>
<td>NVDLTM</td>
</tr>
<tr>
<td></td>
<td>MAEXEQ</td>
</tr>
<tr>
<td>NVRTVRS</td>
<td></td>
</tr>
<tr>
<td>RSCPAI</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>RSCPAT</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>RSCPCI</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>RSCPCT</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>RSCPEI</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>RSCPET</td>
<td>AMPXMOVE</td>
</tr>
<tr>
<td>Routine</td>
<td>Refers to:</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td>RSGTDD</td>
<td>RSCPAI</td>
</tr>
<tr>
<td></td>
<td>RSCPAT</td>
</tr>
<tr>
<td></td>
<td>RSCPCI</td>
</tr>
<tr>
<td></td>
<td>RSCPCT</td>
</tr>
<tr>
<td></td>
<td>RSCPEI</td>
</tr>
<tr>
<td></td>
<td>RSCPET</td>
</tr>
<tr>
<td></td>
<td>RSTRDD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routine</th>
<th>Refers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSTRDD</td>
<td>GETDD</td>
</tr>
</tbody>
</table>
3.9.4 System Translator Hierarchy

3.9.4.1 Postprocessor

The routine hierarchy for the postprocessor Exchange Format/Working Form Translator (IBM Version) is listed below.

Legend:
* MAS routine
+ Stub routine

POST
MAECRN *
MAEXEQ *
RDINX
RESOLVE
MAEGKN *
MAEU *
MALD *
MALGTK *
MALNO *
LTRIM
PRHEAD
GETTOK
LTRIM
RTRIM
RDENT
MAECR *
MAECRN *
MAEXEQ *
MALATC *
MALD *
MALN *
MALRPL *
NEWDD
PUTKEY
MAED *
MAEU *
MALD *
MALFND *
MALGTK *
MALNO *
MALRPL *
GTKEY
    MAECRN *
GTVAL
GETTOK
    LTRIM
    RTRIM
3.9.4.2 Preprocessor

The routine hierarchy for the postprocessor Exchange Format/Working Form Translator (IBM Version) is listed below.

Legend:
* MAS routine
+ Stub routine

PRE
MAECTK *
MAEKND *
MAKXEQ *
CRHEAD
RDINX
CHECK
MAESVL *
WRTENT
MAEXQ *
MAEGKN *
MAESWT *
MAEXEQ *
MALGTK *
MALNO *
CHECK **
GETID
PUTID
NEWDD