WL-TR-91-8035

PRODUCT DEFINITION DATA INTERFACE (PDDI) TRANSLATOR USER'S MANUAL

J. Altemueller
R. Helldoerfer
C. Magnuson et al.

December 1991

McDonnell Aircraft Company
P.O. Box 516
St. Louis, MO 63166

MATERIALS LABORATORY
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) WL-TR-91-8035
6a. NAME OF PERFORMING ORGANIZATION McDonnell Aircraft Company 6b. OFFICE SYMBOL (If applicable) MCAIR	7a. NAME OF MONITORING ORGANIZATION Manufacturing Technology Directorate Wright Laboratory Air Force Systems Command
6c. ADDRESS (City, State, and ZIP Code) McDonnell Douglas Corporation P. O. Box 516, St. Louis, MO 63166	7b. ADDRESS (City, State, and ZIP Code) WL/MTIB Wright-Patterson AFB, OH 45433-6533
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-82-C-5036
8c. ADDRESS (City, State, and ZIP Code) Wright Patterson Air Force Base, Ohio 45433-6533	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 78011F; PROJECT NO. 3095; TASK NO. 06; WORK UNIT ACCESSION NO. 29
11. TITLE (Include Security Classification) PRODUCT DEFINITION DATA INTERFACE	
12. PERSONAL AUTHOR(S) (see reverse side)	
13a. TYPE OF REPORT User's Manual 13b. TIME COVERED FROM Dec 86 TO Sept 87	14. DATE OF REPORT (Year, Month, Day) December 1991 15. PAGE COUNT 98
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP		
1308	0905		Product Definition Data	ICAM Architecture
			Life Cycle Document	CAD/CAM
			Engrg./Mfg. Interface	(continued on back)

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

This document is the Translator User's Manual for the Product Definition Data Interface (PDDI) Extensions contract. This document provides procedures for using the PDDI Translator. Users Manual UM560130001 provides procedures for use of the PDDI Access Software.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☒ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Eric J. Gunther, Lt. USAF	22b. TELEPHONE (Include Area Code) (513) 255-6976 22c. OFFICE SYMBOL WL/MTIB

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

12. Personal Author(s):

Altemueller, Jeffrey
Helldoerfer, Rick
Magnuson, Charles
Purses, John
Whelan, Anna

18. Subject Terms:

Needs Analysis Document
System Requirement Document
State-of-the-Art Document
System Specification Document
SS - Draft Standard
System Design Specification
Product Specification
Operators Manual
Users Manual - Access Software

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and 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.

Alan R Winn — 30 May '91

ALAN R. WINN
Project Manager

DATE

FOR THE COMMANDER:

Bruce Rasmussen — 31 May 91

BRUCE A. RASMUSSEN, Chief
Integration Technology Division
Manufacturing Technology Directorate

DATE

"If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WL/MTI, W-PAFB, 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.

A-1

USER'S MANUAL

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

This document was produced under Air Force Contract F33615-82-C-5036, Product Definition Data Interface (PDDI). This contract is sponsored by the Air Force Wright Aeronautical Laboratories, Materials Laboratory, Air Force Systems Command, Wright-Patterson, Air Force Base, Ohio.

This program is being administered under the technical direction of Lt. Eric Gunther, ICAM Project Manager. The MCAIR Program Manager is Mr. Jerry Weiss and Mr. Herb Ryan is the Deputy Program Manager.

This document was prepared in accordance with the ICAM Configuration Management Life Cycle Documentation requirements for the Configuration Item.

USER'S MANUAL

SECTION 1

SCOPE

1.1 Identification

This User's Manual provides a guide for the use of software developed for the Product Definition Data Interface (PDDI) Project 5601. This project was developed under Air Force Contract F33516-82-C-5036.

1.2 Introduction

Capabilities documented in this manual include:

- o "PUT" a PDDI model into the Working Form - Post-processor
- 0 "GET" a PDDI model from the Working Form - Pre-processor

via the Access Software using the PDDI Translator, and use of the Printout Software Utility.

This PDDI software operates on IBM 43xx, 308xx or DEC VAX 11/780 computers. The environmental requirements are described in Section 3.

The PDDI system documentation does not address local (native) system or computing environment documentation.

This manual addresses IBM procedures and terminology only.

1.3 Other System Manuals

The associated Operator's Manual (OM) describes the system operation and installation procedures. It is intended for use by computer operators and programming personnel.

An associated Users Manual (UM 560130001) provides a guide for application programmers to use the Access Software. Capabilities described in this manual are Access Software Initialization; Entity Creation, Deletion and Manipulation; and List Operations.

The PDDI Product Specification (PS) provides routine descriptions, data dictionary listings and PDDI system messages for system maintenance purposes.

1.4 Approach

This User's Manual is divided into six (6) main sections: Scope, References, System Operations, Post-processing Procedures, Pre-processing Procedures and Miscellaneous Notes and Exceptions.

Section 1 - Scope of this document.

Section 2 - Reference documentation applicable to PDDI and this document.

Section 3 - The PDDI architecture at a high level, system environmental needs, system interfaces, printout procedures, and translator initialization and termination procedures.

Section 4 - The Post-processor procedures. The post-processor "GETS" a PDDI model from the Exchange Format and "PUTS" into the Working Form.

Section 5 - The Pre-processor procedures. The pre-processor "GETS" a PDDI model from the Working Form and "PUTS" it into the Exchange Format.

Section 6 - Miscellaneous information and user messages.

Appendix A - Glossary of terms used in this manual.

Appendix B - An example of a post-process translation.

Appendix C - An example of a pre-process translation.

Appendix D - An example of an Exchange Format Printout.

Appendix E- An example of a Working Form Printout.

SECTION 2

REFERENCES

2.1 Applicable Documents

2.1.1 Specification:

DOD-D-1000B	Drawings, Engineering and Associated Lists
MIL-D-5840	Requirements for Data, Engineering and Technical Reproduction

2.1.2 Standards:

ANSI Y14.5	Dimensioning and Tolerancing
ANSI Y14.26M	Digital Representation Communication of Production Definition Data
ANSI B46.1	Surface Texture (Surface Roughness, Waviness and Lay)
ANSI B92.1	Involute Splines and Inspection
DOD-STD-100C	Engineering Drawing Practices
MIL-STD-9	Screw Thread Conventions and Methods of Specifying
MIL-STD-12	Abbreviations for Use on Drawings, Specifications, Standards and in Technical Documents
IDS150120000C	ICAM Documentation Standards
IEEE STD 829	Standards for Software Test Documentation
ISO/TC184/SC4/WG1	4.2:2 The Step File Structure (Working Paper Version 1.0 28 April 1981

2.1.3 Other Publications:

CLD150120000	ICAM Document Catalog
FTR110210000U	ICAM Architecture
FTR110232000U	ICAM Architecture Part II, Automated IDEFO Development

Product Definition Data Interface

ITR560130001U	First Interim Technical Report (Period 1 Oct 82 - 31 Dec 82)
ITR560130002U	Second Interim Technical Report (Period 1 Jan 83 - 31 Mar 83)
ITR560130003U	Third Interim Technical Report (Period 1 Apr 83 - 30 June 83)
ITR560130004U	Fourth Interim Technical Report (Period 1 Jul 83 - 30 Sep 83
ITR560130005U	Fifth Interim Technical Report (Period 1 Oct 83 - 1 Dec 83)
ITR560230006U	Sixth Interim Technical Report (Period 1 Jan 84 - 31 Mar 84)
ITR560130007U	Seventh Interim Technical Report (Pericd 1 Apr 84 - 30 Jun 84)
ITR560130008U	Eighth Interim Technical Report (Period 1 Jul 84 - 30 Sep 84)
ITR560130009U	Ninth Interim Technical Report (Period 1 Oct 84 - 31 Dec 84)
ITR560130010U	Tenth Interim Technical Report (Period 1 Jan 85 - 31 Mar 85)
ITR560130011U	Eleventh Interim Technical Report (Period 1 Oct 86 - 31 Dec 86)
ITR560130012U	Twelfth Interim Technical Report (Period 1 Jan 87 - 31 Mar 87)
ITR560130013U	Thirteenth Interim Technical Report (Period 1 Apr 87 - 30 Jun 87)
ITR560130014U	Fourteenth Interim Technical Report (Period 1 Jul 87 - 31 Sep 87)
FTR560130001U	Task I, Final Technical Report - System Test Methodology, Volume III
	Technical Operating Report - Product Assurance/Quality Assurance - 15 Oct 85

SD 560130001U	Scoping Document
NAD560130000	Needs Analysis Document
SAD560130000	State-of-the-Art Document
SRD560130000	System Requirement Document
SDS560130000	System Design Specification Document
SS 560130100	System Specification Document
SS 560130200	System Specification Document - Draft Standard
STP560130000	System Test Plan
STR560130000	System Test Report
PS 560130000	Product Specification
OM 560130000	Operator's Manual
UM 560130001	User's Manual - Access Software

2.2 Terms and Abbreviations

The following list explains terminology, acronyms, and other abbreviations used in this document.

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

ANSI - American National Standards Institute.

APPLICATION - Refers generically to any software modules which are used in CAD/CAM functions.

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.

ASCII - American Standard Code for Information Interchange.

ATTRIBUTE - An item of information about an entity. A key attribute identifies the entity; a role iterate gives a fact about an entity.

CAD/CAM - Computer Aided Design/Computer Aided Manufacturing.

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

CLIST - IBM Command lists.

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

CONTEXT-FREE GRAMMAR - The syntax of the language gives a precise specification of the data without interpretation of it.

DOMAIN - The set of values permissible in a given context. A natural domain is the value set native to a given machine architecture; an imposed domain is a specific subset of the natural domain.

DYNAMIC ALLOCATION - The allocation (and decollation) 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.

EBCDIC - Extended Binary Coded Decimal Interchange Code (IBM character set).

ENTITY - A collection of facts (attributes) about something of interest.

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

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.

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 PDDI Access Software specifications.

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.

JCL - Job Control Language - IBM language used to identify a job and describe its requirements to an operating system.

KEY - An item of data that uniquely identifies some specific instance of an entity.

MAS - MCAIR's acronym for the PDDI Access Software (Model Access System).

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.

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

PDD - Product Definition Data.

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.

QUALITY - The composite of all the attributes or characteristics including performance of an item or product.

QUALITY ASSURANCE - The planned and systematic establishment of all actions (management/engineering) necessary to provide adequate confidence and nonconformance prevention provisions and reviews are established during the design phase and performed throughout the software development and life cycle phases.

QUALITY CONTROL - The planned and systematic application of all actions (management/technical) necessary to control raw materials or products through the use of test, inspect, evaluate, and control of processes.

REQUESTED DATA - See Application Request Data.

RUN SYSTEM - The Translator sub-package which provides the communication interface between the user and the pre/post-processors.

SCHEMA - Those definitions which describe the content of the data and the relationship between the various elements or components of the data.

SOFTWARE QUALITY ASSURANCE (SQA) - The planned and systematic establishment of all actions necessary to provide adequate confidence that nonconformance prevention provisions and reviews are established during the design phase and performed throughout the software development and life cycle phases.

SOFTWARE QUALITY ASSURANCE PLAN (SQAP) - An organized description of the methods, policies, and procedures necessary to conduct software quality assurance and control activities during the design, development, delivery, and maintenance phases.

SOFTWARE QUALITY CONTROL - The planned and systematic application of all actions (management/technical) necessary to ensure that the software under development or maintenance satisfies the technical requirements through the use of tests, demonstrations, inspections, evaluations, and control of processes.

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.

TRANSFER DATA - The data required to make an exchange of data between systems (e.g., 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.

TSO - Time Sharing Option - IBM function which provides conversational time sharing from remote terminals.

USER COMPUTER SYSTEM - The specific hardware, operating systems, and applications software systems that the user will employ to implement the PDDI Access Software.

WORKING FORM - A memory resident form of a model that supports rapid access to entities via the Access Software.

WORKING FORMAT - The physical representation of the Working Form within the computer.

SECTION 3

SYSTEM OPERATIONS

3.1 System Overview

The purpose of the 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 Access Software, Conceptual Schema, Exchange format and a Translator. (See Figure 3-1).

The Access Software is a set of callable utility programs that will allow applications to manipulate and query PDD. The Conceptual Schema is a data dictionary that contains the data needed to define a CAD/CAM model. The Exchange Format is a neutral physical sequential format for passing data between dissimilar systems. The PDDI 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 the PDDI conceptual schema and the specification for the neutral file format.

3.1.2 Software Packages

The software for the system consists of two (2) packages - Access Software and Translator.

3.1.2.1 Access Software

The PDDI Access Software package is an integrated set of routines that create and manage an incore Working Form of the PDDI data structure through key access. This Access Software keeps the application independent of the actual physical definition of the Working Form. It also serves as a bridge between existing CAD/CAM systems and the PDDI Exchange Format. The PDDI Access Software reduces the task of writing the Exchange Format by providing the utility functions for initializing the Working Form model, manipulating entities, and maintaining lists.

The PDDI Access Software operates on the data structure of the application and the Working Form, by using either entity or list operations. The entity operations allow the user to create, delete, modify and query entities. List operations manage the lists which are temporary data structures containing references to entities (keys). An application can build and maintain lists specifically for its needs.

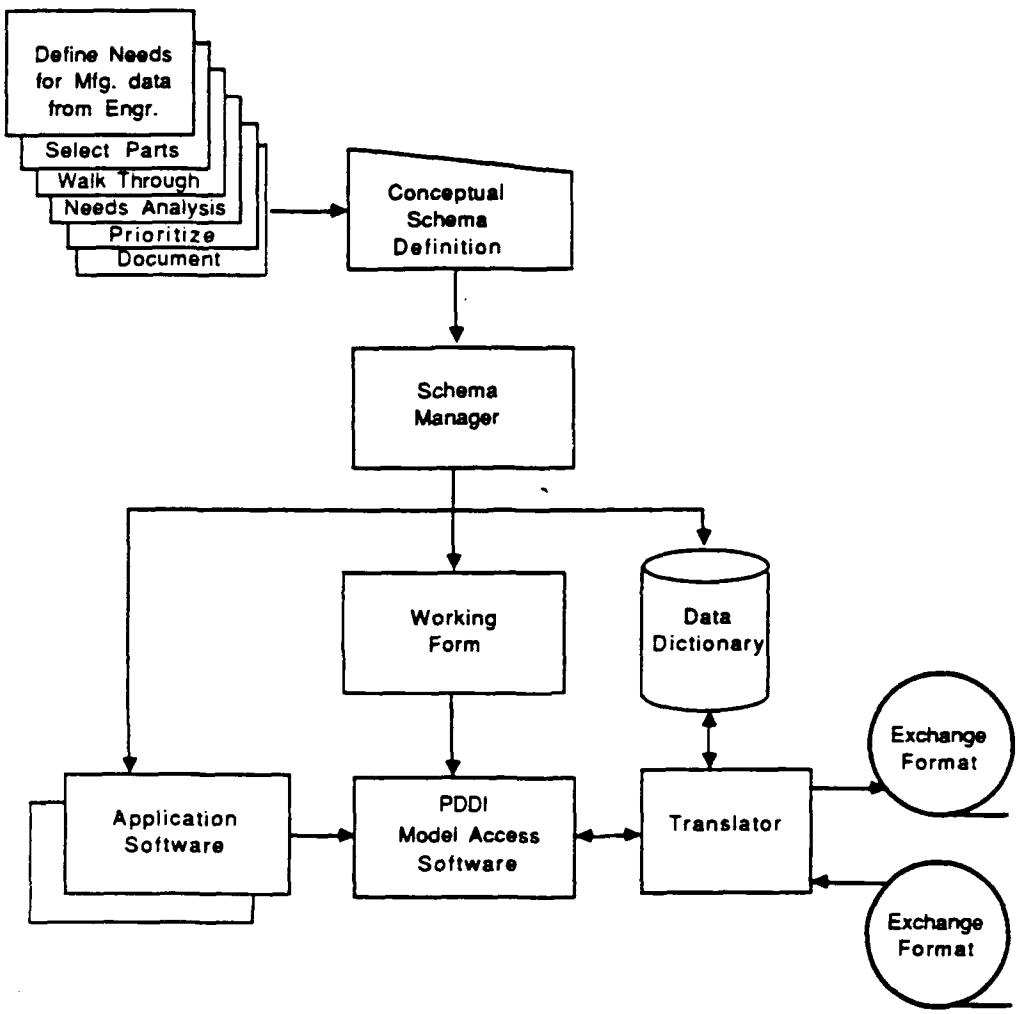


Figure 3-1 PDDI System Architecture

The PDDI Translator is the software package used to transmit the PDD between systems. The Translator consists of three main sub-packages. These sub-packages are: "Run System", "Pre-Processor" and "Post-Processor". (See Figure 3-2).

The Run System is the interface between the user and the "processors". This package provides menus, queries and system responses to the user.

Functions performed by this package include: Perform system configuration activities, determine files needed by the processors and make them available, and provide messages to aid user interfaces.

The ability to access the native database is provided for by this package via calls to user-supplied routines. These routines allow data from the native database to be placed into or obtained from the Working Form using calls to the Access Software. The pre-processor or post-processor is then called to perform the desired translation.

The Pre-Processor provides the interface from the Working Form to the Exchange Format.

Working Form entities, in the Working Form physical schema, are accessed via the Access Software. Tables, obtained from the Run System, are then used to map the Working Form entities to the Exchange Format physical schema. The Exchange Format entities are then encoded and placed into the Exchange Format file.

Transfer data is collected during entity processing. This data is encoded and placed into the Exchange Format file.

Error messages or condition codes are sent to the "Run System" to indicate the status of the transfer.

The Post-Processor provides the interface from the Exchange Format to the Working Form.

A set of tables, obtained from the Run System, are then used to map the Exchange Format entities to the Working Form physical schema. The Access Software is then used to place these entities into the Working Form.

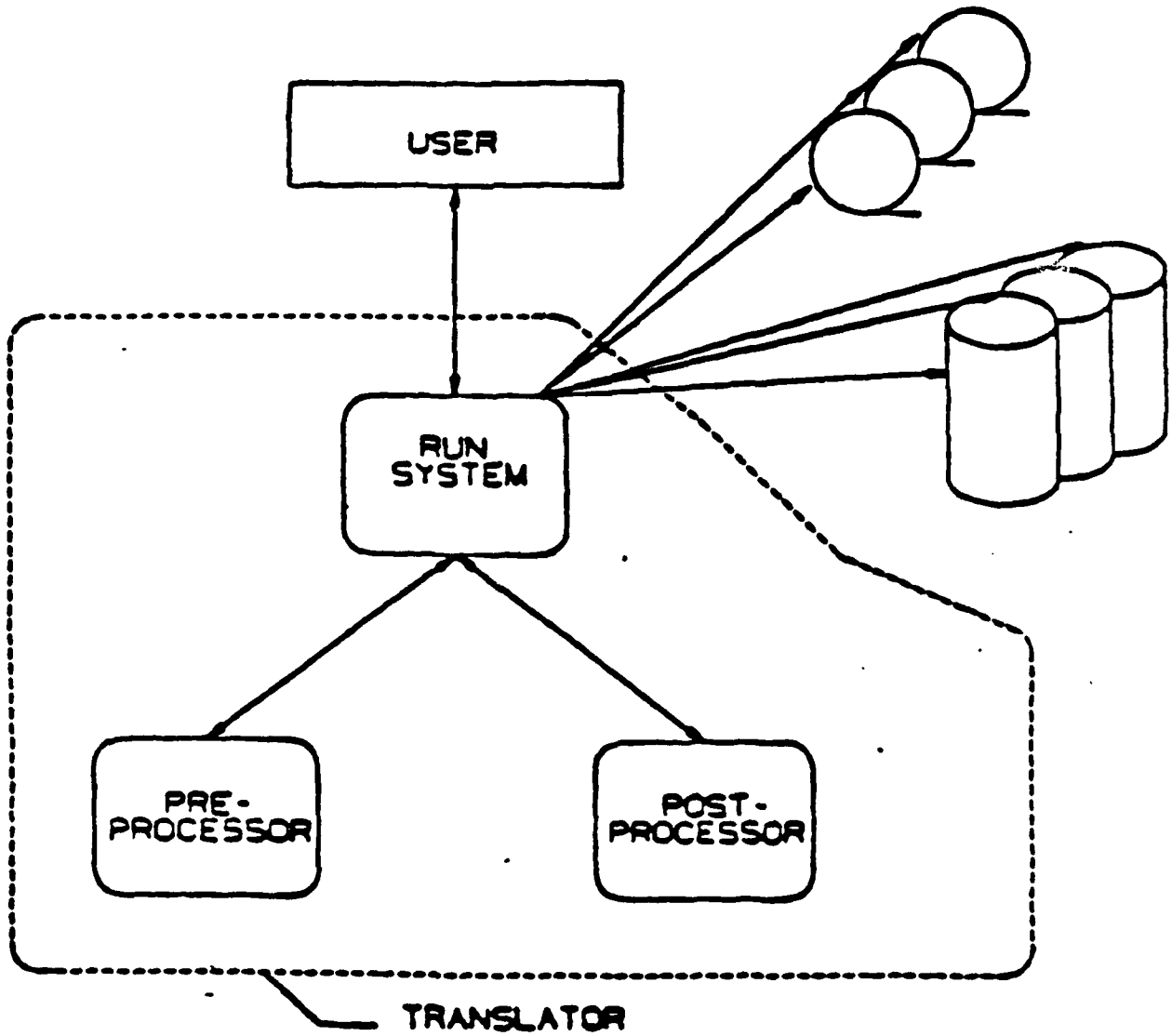


Figure 3-2 Translator Architecture

3.2 System Interfaces

The PDDI software must interface with the computer system on which it is installed, the local (native) CAD/CAM database, the Exchange Format, the Working Form, and the user (application). It does this via PDDI Access Software, the PDDI Translator and local (native) developed software packages. Note: Simple interim database software is included in the Translator software. This software is an interim program to be used until an interface to the native database system is available. See Appendix D for an explanation. Figure 3-3 shows the environment in which the PDDI system was developed. This figure also shows the versatility of the system and the multi-hardware environment in which it may be used. The left-hand side of Figure 3-3 shows the PDDI development environment.

3.3 System Environment

The PDDI system was developed in the following computing environment:

Computer/Operating System

IBM 43XX/MVS with TSO and associated tape drives, disk drives and terminals.

DEC VAX 11/780 VMS with associated tape drives, disk drives and terminals.

Storage (Core) Requirements

The minimum core requirements for the PDDI software and database is 1.0M plus the size of the model. (The PDDI Machined Rib model required .57M)

Compilers

IBM-PASCAL/VS Release 2.2
DEC-PASCAL V3.3, FORTRAN 77 V4.4

Terminals

E&S PS300 (or equivalent for graphics applications)
IBM 3270 (or equivalent)

The PDDI system is transportable to other computing systems. However, appropriate local (native) interfaces (translator) must be provided. The right-hand side of Figure 3-3 shows the PDDI commercial demonstration architecture for UNIGRAPHICS and Computervision Systems.

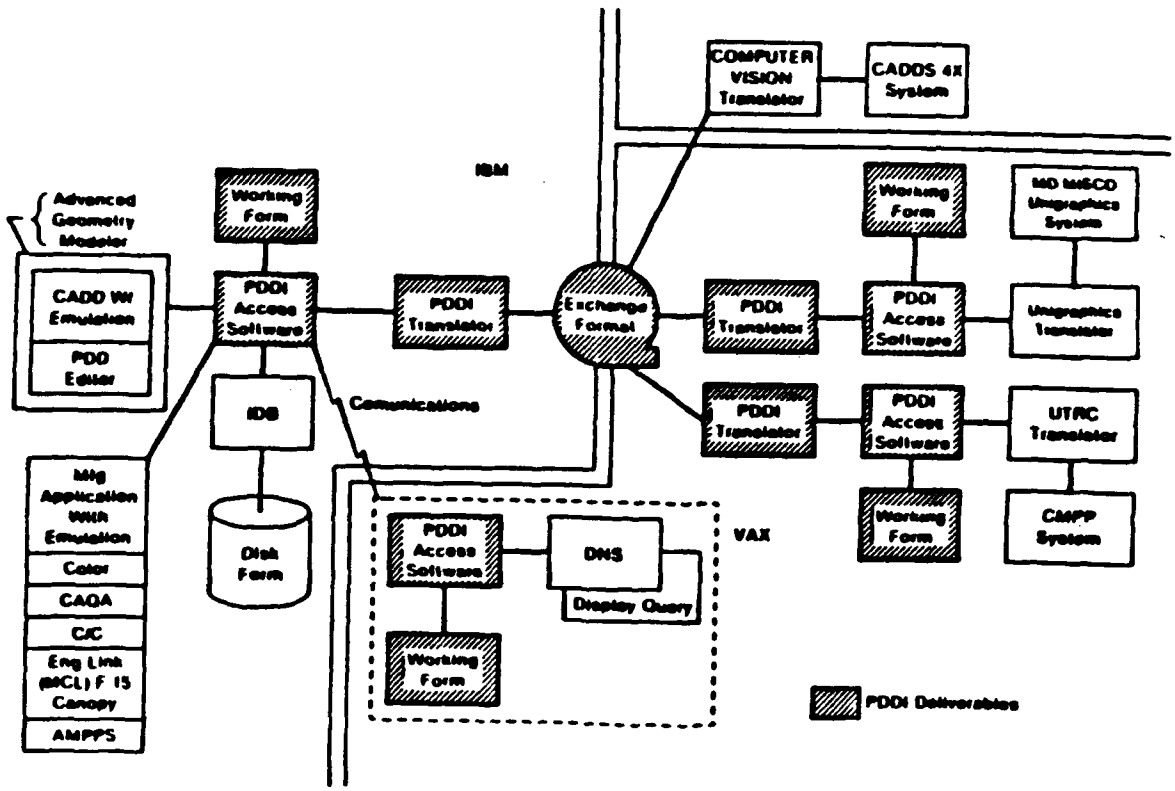


Figure 3-3 PDDI Environment

3.4 Translator Initialize/Terminate Procedures

This manual does not provide (native) system logon or terminal operating procedures since each computing environment is unique.

It is assumed that the PDDI software has been installed in the local computing environment and that the necessary local (native) translator requirements have been met. In addition, it is assumed that the PDDI models to be translated exist on tape or disk, or in the Working Form.

To initiate the PDDI Translator Session enter:

```
V30MENU
```

after the system READY message appears. The Translator User Menu then appears on the screen (Figure 3-4).

3.4.1 Initialize Procedures

Table 3-1 provides the step by step procedure to initialize the Pre-processor (Working For to Exchange Format) or Post-processor (Exchange Format to Working Form).

```
                EXCHANGE FORMAT/WORKING FORMAT TRANSLATOR

PDES                                                VERSION:  2.02
                                                    DATE:  09/10/86
                                                    TIME:  15:48:10


   USER'S LAST NAME:____________________
          LOGON ID:____________________

   TERMINATE (Y/N):___


   TRANSACTION REQUIRED:
     WORKING TO EXCHANGE FORMAT (Y/N):___
     EXCHANGE TO WORKING FORMAT (Y/N):___
```

Figure 3-4 Translator Main User Menu

Table 3-1 Initialize Transaction

MESSAGE	RESPONSE	COMMENTS
USER'S LAST NAME:	Enter Name	
LOGON ID:	Enter User ID	
TERMINATE (Y/N):	Enter N	If Y is entered Termination Function is initiated (see Table 3-2).
TRANSACTION REQUIRED: WORKING TO EXCHANGE FORMAT (Y/N):	Y	If Y is entered the Pre-processor function is initiated (see Section 5).
	N	If N is entered the following message appears.
EXCHANGE TO WORKING FORM (Y/N):	Y	If Y is entered the Post-processor Function is initiated (see Section 4).
	N	If N is entered error message 1 appears. (See Section 6.1)

3.4.2 Terminate Procedure

After the post-processing or the pre-processing translations are completed, the Main User Menu appears on the screen. To terminate the Session enter Y in the appropriate space and (CR) (see Table 3-2).

Session termination information will then be displayed (see Figure 3-5).

Table 3-2 Terminate Transaction

MESSAGE	RESPONSE	COMMENTS
Terminate (Y/N)	Y	This takes the User back to the local (native) system.

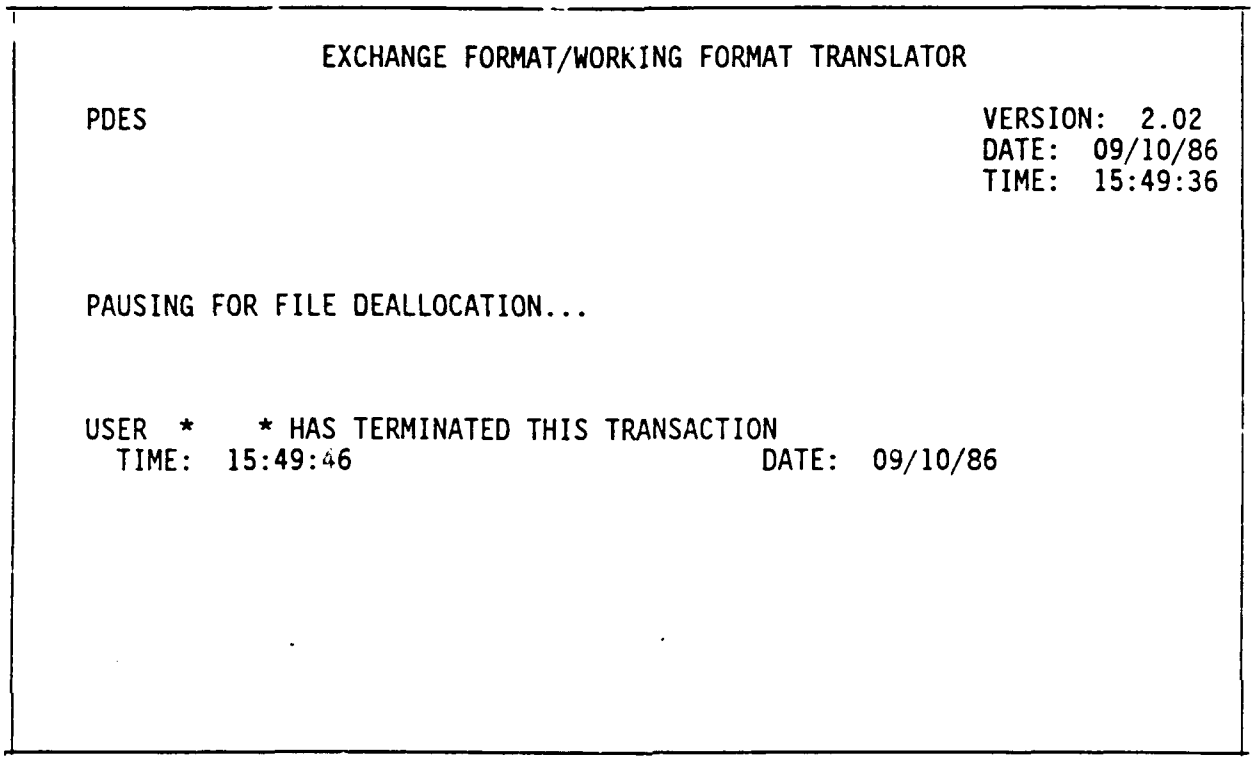


Figure 3-5 Terminate Menu

3.5 Printout Utility Procedures

Prior to post-processing sessions, the system queries the User for desired printouts.

The working form printout provides a hardcopy of each entity instance after post translation (see Figure 3-6). The Attribute Data Block (ADB), Constituents (pointers to other entities), and User's Data (pointers to parents) for each entity are provided in the printout.

Note: The printout utility prints out each instance of every entity type within a model. This will result in a VERY LARGE resource consuming printout for large models.

Figure 3-7 shows the menu used to select the desired printout.

Figure 3-8 shows the message for a Working Form printout.

Appendix B provides an example of a Working Form printout. This is a printout of a sample model which has been post-processed into the Working Form.

3.6 Mapping Function

A mapping file may be requested when it maintains a correspondence between the entity identifier on the Exchange Format and the Kind and Ident of the entity on the Working Form (see Table 3.1). This provides a means to manually identify each entity.

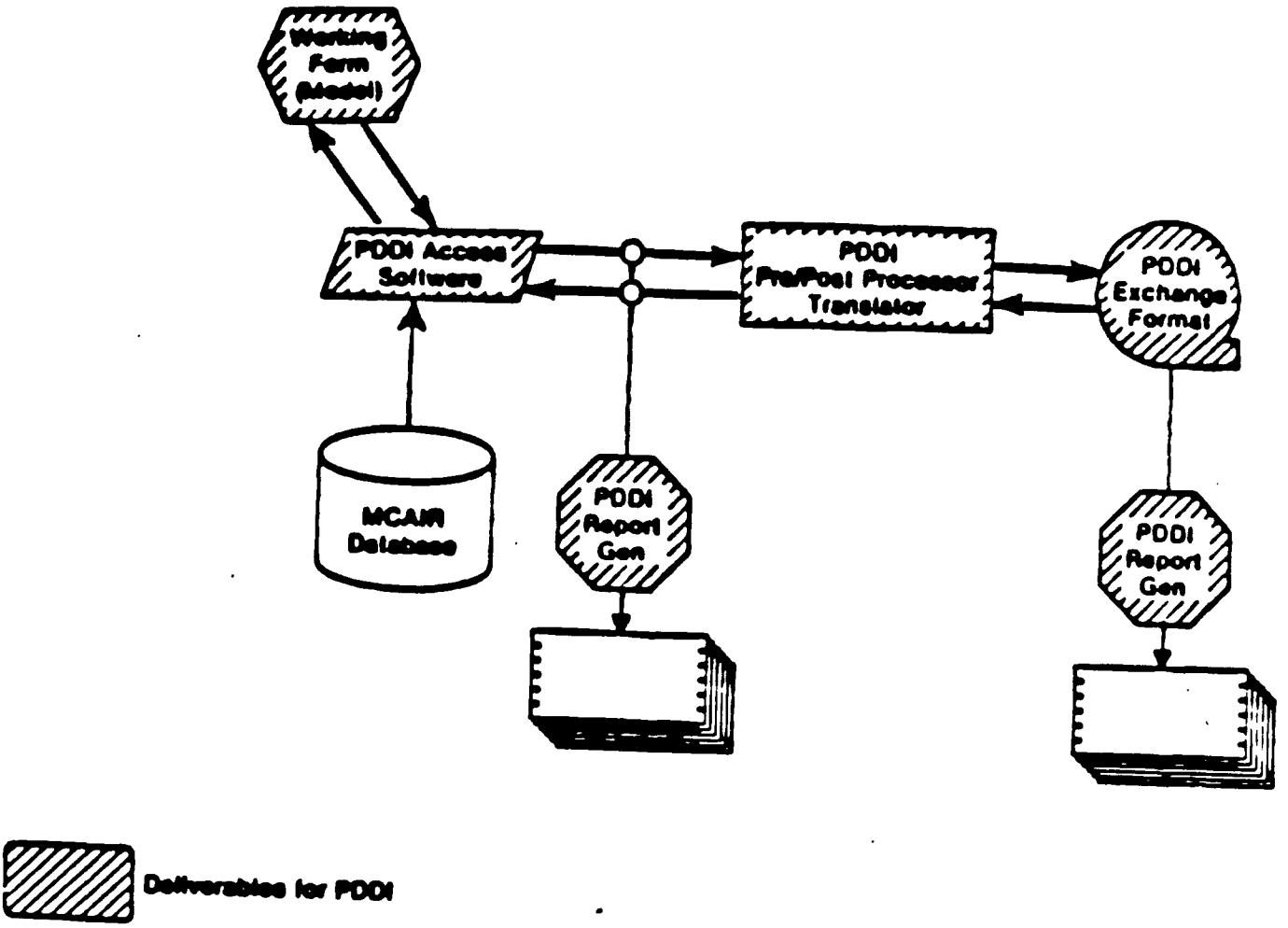


Figure 3-6 Printout Utility

```
                EXCHANGE FORMAT/WORKING FORMAT TRANSLATOR

PDES                                                    VERSION:  2.02
                                                        DATE:  09/10/86
                                                        TIME:  15:54:42


NOTE:  The working Form Printout will only be generated in the
       post-processor


PLEASE ENTER WF FOR WORKING FORM PRINTOUT
             MA For mapping file
             BO for both printouts
             NO FOR NO PRINTOUTS

    >>BO

   Fully qualified file name for mapping file:
```

Figure 3-7 Printout Menu

```
                EXCHANGE FORMAT/WORKING FORMAT TRANSLATOR

PDES                                                  VERSION:  2.02
                                                      DATE:  09/12/86
                                                      TIME:  11:54:30


   WORKING FORM PRINTOUT WILL BE IN FILE:
          'TSOX053.WF115329.WFFILE'
   WORKING FORM PRINTOUT WILL BE PROCESSED...

    CONTINUE (CR):
```

Figure 3-8 Working Form Printout Menu

SECTION 4

POST-PROCESSING (EXCHANGE FORMAT TO WORKING FORM) PROCEDURES

4.1 Introduction

The Post-processor translates a PDDI model from the Exchange Format to the Working Form on the computer. The options available for processing include batch or interactive. The medium which may be used is tape or disk. However, tapes cannot be accessed interactively. These procedures will be divided into Batch Procedures-Disk, Batch Procedures-Tape and Interactive Procedures.

4.2 Model Filing

Table 4-1 shows procedures for filing a model in the Interim Database software (see Appendix D). These procedures will vary according to the local system's database procedures (see Table 4-1 and Figure 4-1).

Table 4-1 Model Filing Procedures

MESSAGE	RESPONSE	COMMENTS
Transaction required: Working to Exchange Format (Y/N):	N	From Main User Menu
Exchange to Working Format (Y/N)	Y	
Transaction: Exchange to Working Format		
Internal (I) or PDDI Supplied (P) database	P	
Key in the database dataset to file To/From	Enter Dataset Identifier	Enter the dataset in which to file the working form model.

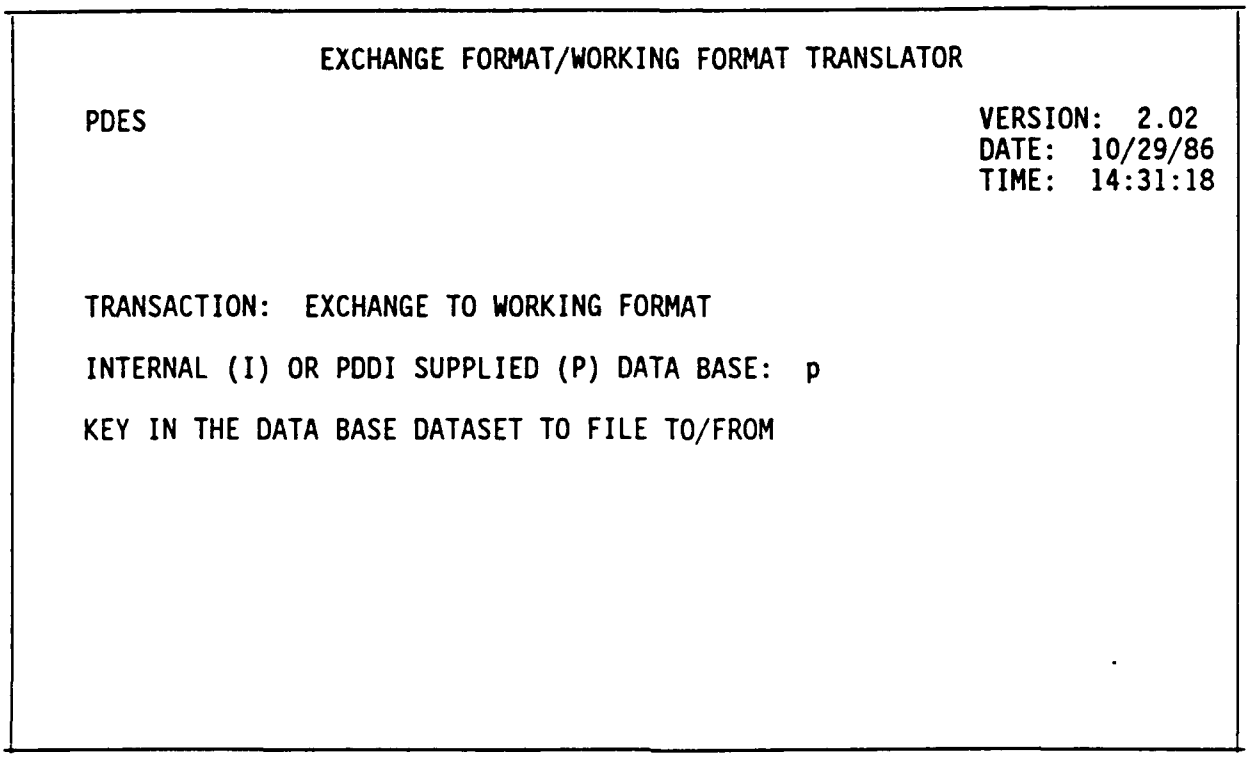


Figure 4-1 Cluster Menu - Post-Processor

4.3 Batch Post Processing (Disk) - Table 4-2 shows procedures that translate a model from the Exchange Format on disk to the Working Form in batch mode (see Table 4-2 and Figures 4-2).

Table 4-2 Batch Post-Processing Procedures - Disk

MESSAGE	RESPONSE	COMMENTS
Transaction: Exchange to Working Form		
Batch (B) or Interactive (I)	B	
Is the Exchange File Dataset on Disk? (Y/N):	Y	
Key in the Exchange File Dataset Name (without quotes)	Enter dataset identifier	This is the dataset containing the Exchange Format file.
For batch transfer, please enter the system account Number	Enter appropriate Account Number	When the job has been submitted, an appropriate system message appears. The main User Menu will then appear.

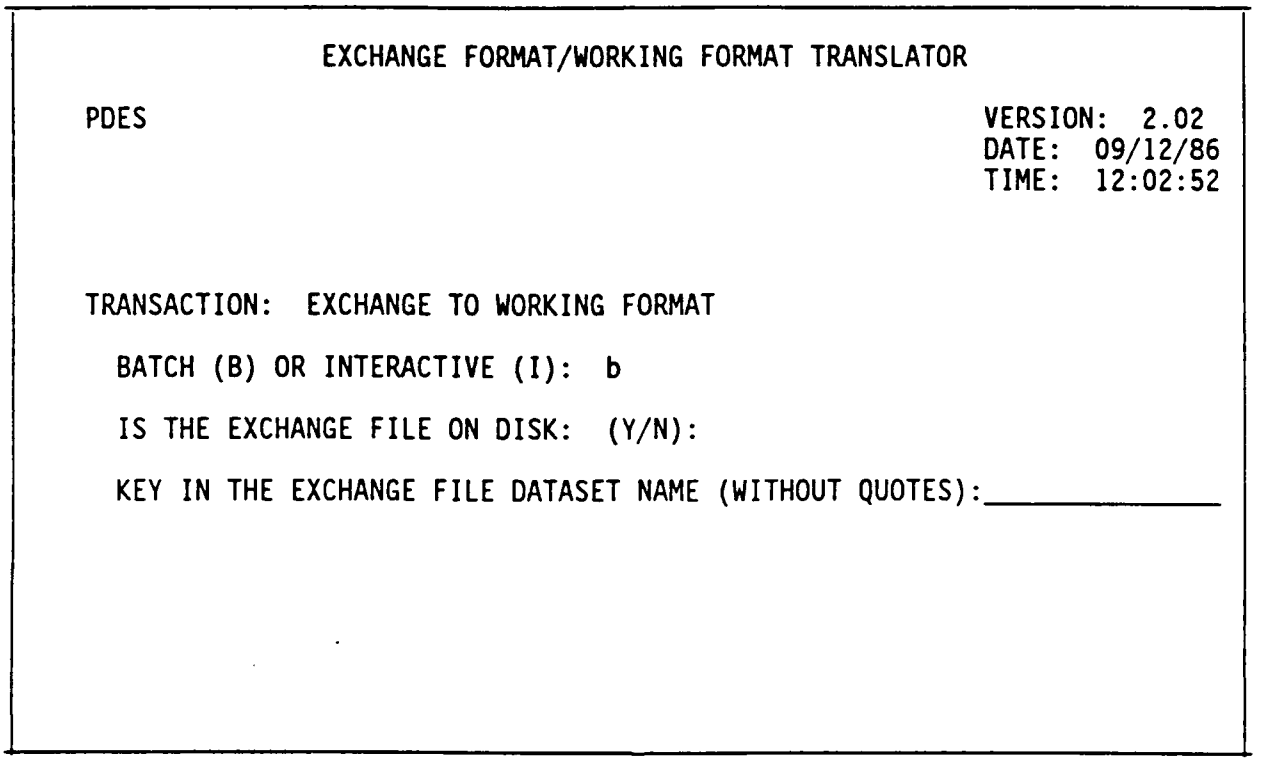


Figure 4-2 Batch Post-Processing Menu - Disk

4.4 Batch Post-Processing (Tape)

Table 4-3 shows procedures that translate a model from the Exchange Format on tape to the Working Form in batch mode (see Table 4-3 and Figure 4-3).

Table 4-3 Batch Post-Processing Procedures - Tape

MESSAGE	RESPONSE	COMMENTS
Transaction: Exchange to Working Form		
Batch (B) or Interactive (I)	B	
Is the Exchange File Dataset on Disk (Y/N):	N	
Dataset on tape (Y/N):	Y	
Tape Name:	Enter tape name	This is the volume serial for the magnetic tape reel.
Label Number:	Enter tape label number	This is the position of the Exchange Format file on the tape.
For Batch transfer, please enter the system Account Number	Enter appropriate system Account Number	When the job has been submitted, an appropriate system message appears. The main User Menu will then appear.

```
                EXCHANGE FORMAT/WORKING FORMAT TRANSLATOR

PDES                                                  VERSION:  2.02
                                                      DATE:  09/12/86
                                                      TIME:  12:04:30


TRANSACTION:  EXCHANGE TO WORKING FORMAT

  BATCH (B) OR INTERACTIVE (I):  b

  IS THE EXCHANGE FILE DATASET ON DISK:  (Y/N):  n

    REMINDER:  TAPES CAN ONLY BE ACCESSED BY BATCH

  DATASET ON TAPE (Y/N):  Y
    TAPE NAME:________
    LABEL NUMBER:________
```

Figure 4-3 Batch Post-Processing Menu - Tape

4.5 Interactive Post-Processing

Table 4-4 shows procedures that translate a model from the Exchange Format to the Working Form in an interactive mode (see Table 4-4 and Figure 4-4).

Note: o Tapes cannot be processed interactively.

Table 4-4 Interactive Post-Processing Procedures

MESSAGE	RESPONSE	COMMENTS
Transaction: Exchange to Working Form		
Batch (B) or Interactive (I)	I	
Please wait while files are allocated		
Is the Exchange File Dataset on disk? (Y/N):	Y	
Key in the Exchange File name (without quotes):	Enter dataset name	This is the dataset containing the Exchange Format file.
Please wait while files are allocated		
Processing begins . . .		
Processing ends		The Main User Menu will appear when processing ends.

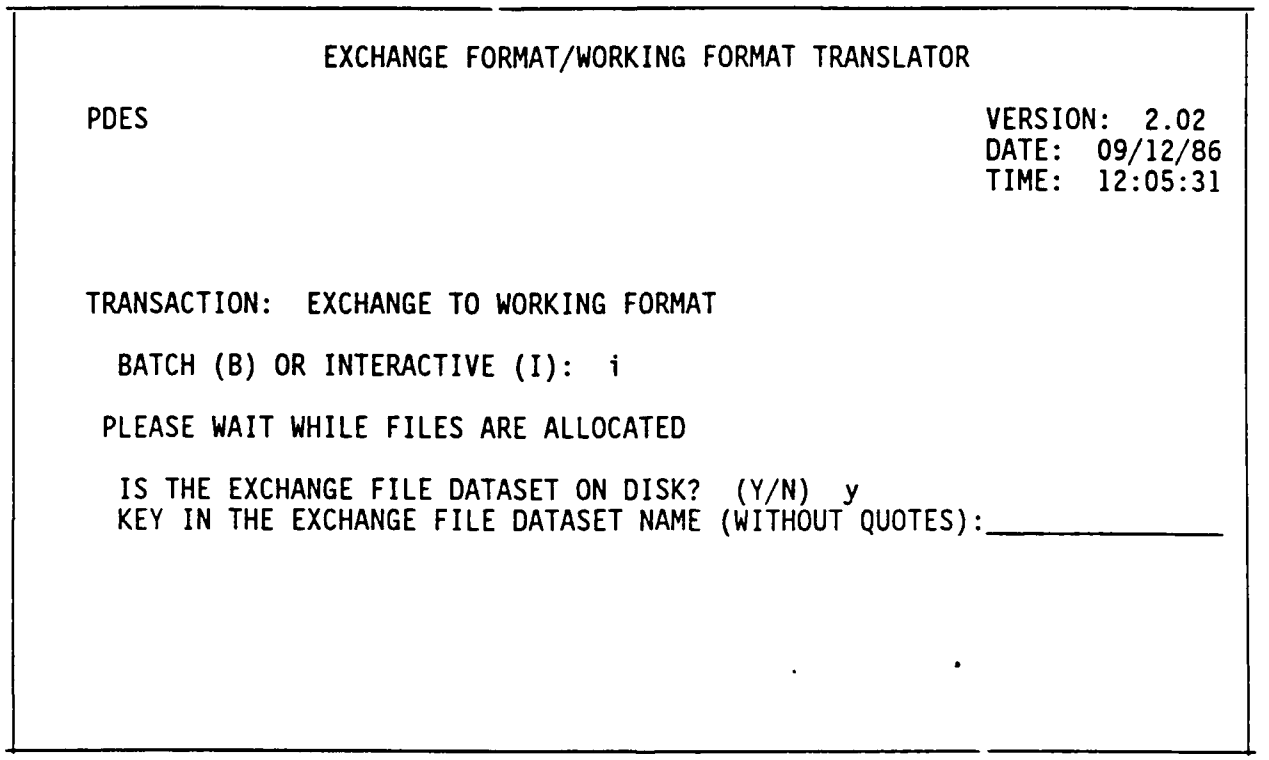


Figure 4-4 Interactive Post-Processing Menu

SECTION 5

PRE-PROCESSING (WORKING FORM TO EXCHANGE FORMAT) PROCEDURES

5.1 Introduction

The Pre-processor translates a PDDI model from the incore Working Form on the computer to the Exchange Format. The procedures are very similar to post-processing procedures The options available for processing include batch or interactive. The medium which may be used is tape or disk. However, tapes cannot be accessed interactively. These procedures will be divided into Batch Procedures-Disk, Batch Procedures-Tape, and Interactive Procedures.

5.2 Model Retrieval

Table 5-1 shows procedures for retrieving a model from the Interim Database software (see Appendix D). These procedures will vary according to the local system's database procedures (see Table 5-1 and Figure 5-1).

Table 5-1 Model Retrieval

MESSAGE	RESPONSE	COMMENTS
Transaction Required: Working to Exchange Format (Y/N)	Y	From Main User Menu
Transaction: Working to Exchange Format		
Internal (I) or PDDI Supplied (P) database:	P	
Key in the Database Dataset to File To/From	Enter Dataset Identifier	Enter the dataset from which to retrieve the working form model.

5.3 Batch Pre-Processing (Disk)

Table 5-2 shows the procedures that translate a model from the Working Form to the Exchange Format (on disk) in the batch mode (see Table 5-2 and figure 5-2).

Table 5-2 Batch Pre-Processing Procedures - Disk

MESSAGE	RESPONSE	COMMENTS
Transaction Required: Working to Exchange Format (Y/N):	Y	From Main User Menu
Transaction: Working to Exchange Format		
Batch (B) Interactive (I)	B	
Is the Exchange File Dataset on disk? (Y/N):	Y	
Key in the Exchange File Dataset name (without quotes):	Enter dataset name of desired model	This is the dataset containing the Exchange Format file.
Explanations (comments): (Enter a blank to end comments)	Enter desired comments	These will appear as comments in the Exchange Format file.
For batch transfer, please enter system Account Number:	Enter appro-priate System Account Number	When the job has been submitted an appropriate system message appears. The Main User Menu will then appear.

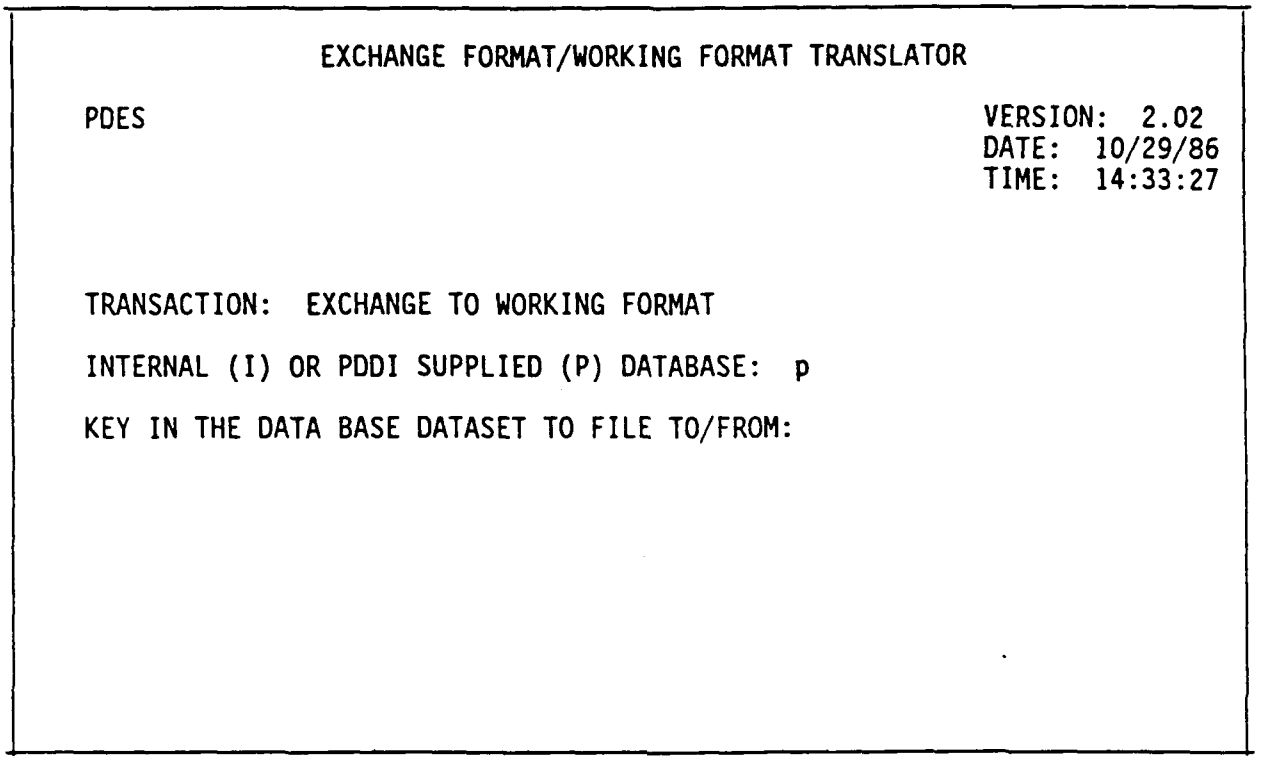


Figure 5-1 Cluster Menu - Post-Processor

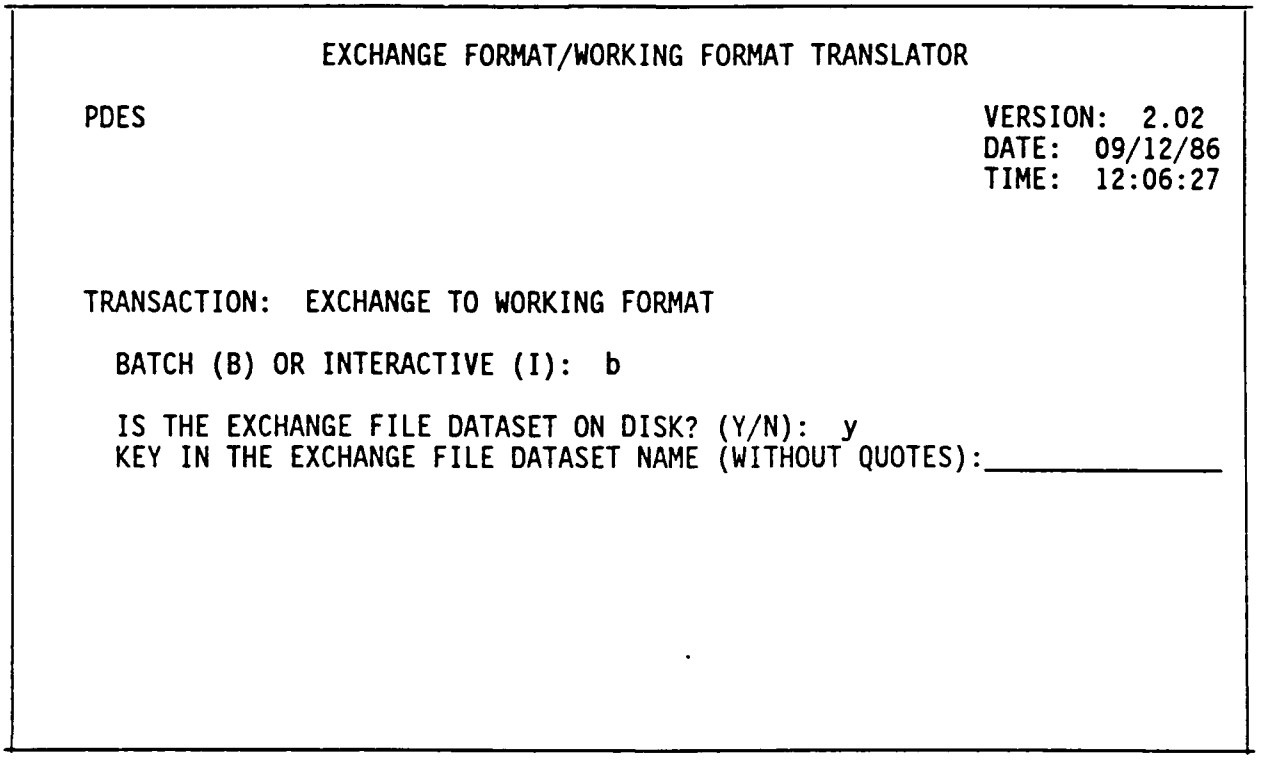


Figure 5-2 Batch Pre-Processing Menu - Disk

5.4 Batch Pre-Processing (Tape)

Table 5-3 shows the procedures that translate a model from the Working Form to the Exchange Format (on Tape) in the batch mode (see Table 5-3 and Figure 5-3).

Table 5-3 Batch Pre-Processing Procedures - Tape

MESSAGE	RESPONSE	COMMENTS
Transaction: Working to Exchange Format		
Batch (B) or Interactive (I)	B	
Is the Exchange File Dataset on Disk? (Y/N):	N	
Remainder: Tapes cannot be accessed interactively.		
Dataset on Tape (Y/N):	Y	
Tape Name:	Enter tape name	This is the volume serial for the magnetic tape reel.
Label Number:	Enter label number	
Explanations (comments): (Enter a blank to end comments)	Enter desired comments	These will appear as comments in the Exchange Format file.
(Enter a blank to end comments)		
For batch transfer, please enter system Account Number:	Enter appropriate System Account Number	When the job has been submitted an appropriate system message appears. The Translator Main User Menu will then appear.

```
                EXCHANGE FORMAT/WORKING FORMAT TRANSLATOR

PDES                                                VERSION:  2.02
                                                    DATE:  09/12/86
                                                    TIME:  12:07:13


TRANSACTION:  EXCHANGE TO WORKING FORMAT

  BATCH (B) OR INTERACTIVE (I):  b

  IS THE EXCHANGE FILE DATASET ON DISK? (Y/N):  n

    REMINDER:  TAPES CAN ONLY BE ACCESSED BY BATCH

  DATASET ON TAPE (Y/N):  y
    TAPE NAME:________
    LABEL NAME:________
```

Figure 5-3 Batch Pre-Processing Menu - Tape

5.5 Interactive Pre-Processing

Table 5-4 shows the procedures that translate a model from the Working Form to the Exchange Format in an interactive mode (see Table 5-4 and Figure 5-4).

Note: Model translation can take 2-3 hours, depending on the size of the model.

Table 5-4 Interactive Pre-Processing Procedures

MESSAGE	RESPONSE	COMMENTS
Transaction: Working to Exchange Format		
Batch (B) or Interactive (I)	I	
Please wait while files are allocated		
Is the Exchange File Dataset on Disk? (Y/N):	Y	
Key in the Exchange File dataset name (without quotes)	Enter dataset name of desired model	This is the dataset containing the Exchange Format file.
Explanations (comments): (enter a blank line to end comments)	Enter desired comments	These will appear as comments in the Exchange Format file.
Please wait while files are allocated		
Processing begins . . .		
Processing ends		The Main User Menu will appear when processing ends.

```
                EXCHANGE FORMAT/WORKING FORMAT TRANSLATOR

PDES                                                  VERSION:  2.02
                                                      DATE:  09/12/86
                                                      TIME:  12:08:01


TRANSACTION:  EXCHANGE TO WORKING FORMAT

  BATCH (B) OR INTERACTIVE (I):  i

 PLEASE WAIT WHILE FILES ARE ALLOCATED...

  IS THE EXCHANGE FILE DATASET ON DISK? (Y/N):  y
  KEY IN THE EXCHANGE FILE DATASET NAME (WITHOUT QUOTES):_____________
```

Figure 5-4 Interactive Pre-Processing Menu - Disk

SECTION 6

MISCELLANEOUS NOTES AND EXCEPTIONS

6.1 PDDI Translator User Error Messages:

ERROR 1: Please specify a transaction or ...

REASON : User needs to specify either
Working Form to Exchange Format (Pre-Processor)
-or-
Exchange Format to Working Form (Post-Processor)

ERROR 2: Please specify Batch (B) or Interactive (I)

REASON : User needs to specify whether the translator should be executed by Batch or Interactively.

ERROR 3: Dataset is invalid

REASON : Post-Processor - dataset specified does not exist or is not allocated.

ERROR 4: Tapes can only be accessed by Batch

REASON : If database is on tape or to be put on tape, the translator must be executed through Batch.

APPENDIX A

UNDERSTANDING AND MAPPING INTO THE WORKING FORM MODEL

APPENDIX A

UNDERSTANDING AND MAPPING INTO THE WORKING FORM MODEL

OVERVIEW

Pascal Include Files are the primary method for accessing the data in the PDDI entities. The Data Dictionary shows the way Pascal stores this data in the entity. It is intended to be used by FORTRAN and Assembler applications to access the data.

The data dictionary is a set of entity definitions where each entity is defined within a direct access file. An entity definition can be accessed by supplying the entity kind number to the routine "GETDD". GETDD uses an index file to locate the entity definition within the Direct Access File. A detailed description of the "GETDD" user specification is provided in Paragraph 4.3.

An example of an entity definition is illustrated below. IMPL_B_HOLE is a subentity. A subentity signifies that the entity was created as a result of the physical implementation of the conceptual schema. The original structure of this entity, in the Conceptual Schema, was the BLIND_HOLE entity.

```
1st rec    IMPL_B_HOLE      , 1204,13,1
2nd rec    KIND             , 1, 1,  1, 0,1,  4,     0
           LENGTH           , 2, 2,  1, 0,1,  4,     4
           SYSUSE           , 3, 3,  1, 0,1,  4,     8
           VERSION          , 4, 4,  1, 0,1,  4,    12
           SYS_IDENT        , 5, 5,  1, 0,1,  4,    16
    t      IDENT            , 6, 6,  1, 0,1,  4,    20
    h      DIAMETER         , 9, 7,  1, 0,2,  4,    24
    r      PARM             ,12, 8,  1, 0,2,  4,    28
    o      HOLE_TYPE        ,11, 9,  1, 0,5,  1,    32
    u    X  3,CONE             ,BULL                ,BALL
    g      LOCATE           , 7,10,  1, 0,7,  4,     1
    h    X  1, 4000
           AXIS             , 8,11,  1, 0,7,  4,     2
         X  1, 3002
           BOTTOM           ,10,12,  1, 0,7,  4,     3
         X  1, 4000
           ENTRY            ,13,13,  1, 1,7,  4,     4
         X  1, 8004
nth rec  A  1,254
```

ATTRIBUTE RECORDS

The definition of an entity is comprised of many attribute records. With the exception of the first record, each record defines an attribute or part of an attribute within the entity.

First Record

The first record of the entity definition contains four (4) pieces of information about the entity. Below is an example of this record.

```
   (1)                  (2)   (3)  (4)
IMPL_B_HOLE           , 1204,  13,   1
```

(1) ENTITY NAME field (16 character alpha)

(2) ENTITY KIND field

(3) NUMBER OF ATTRIBUTES FIELD (2 character numeric)

(4) DELETABILITY STATUS field (1 character) 0 or 1

- Zero (0) implies this entity is independent. This entity does not require a parent entity to exist and can be deleted at any time.
- One (1) implies this entity is dependent. This entity requires a parent entity to exist and cannot be deleted if its parent is present.

Second - n Records

The second through n records of an entity definition contains the definition of each attribute within the entity. The different definition fields contained in each of these records are as follows:

(1) CONTINUATION FLAG field (1 character alpha X, B, E, or A)

```
*    (1) (2)                 (3) (4) (5)(6)(7)(8)   (9)       *
*      ENTRY                   ,13,13,  1, 1,7,  4,     4     *
*     X  1, 8004                                               *
*     A  1,254                                                 *
```

- A letter signifying additional data about an attribute is contained on this line. There are four possible continuation flags.

  "X" - Marks a continuation line that defines additional data for SCALAR, SUBENTITY, and POINTER data types.

"B" and "E" - Respectively marks a continuation line that defines the beginning and end of a group of attributes that are defined as a structure.

"A" - Marks a continuation line that defines the array bounds for the previously defined attribute. An attribute can be defined in terms of a 1-n dimensional array. An example of the representation of a 3 dimensional array, of ADB information, is as follows.

```
Line 1 =>  MATRIX          , 7, 7,  1, 3,2,  4,    24
Line 2 => A  1,  2,1,  3,1,  4
```

This example defines a 3 dimensional array that is then defined as...

```
1)  REAL*4 MATRIX(4,3,2) for "FORTRAN" and
2)  MATRIX : ARRAY(.1..2.) OF
             ARRAY(.1..3.) OF
             ARRAY(.1..4.) OF SHORTREAL for "PASCAL"
```

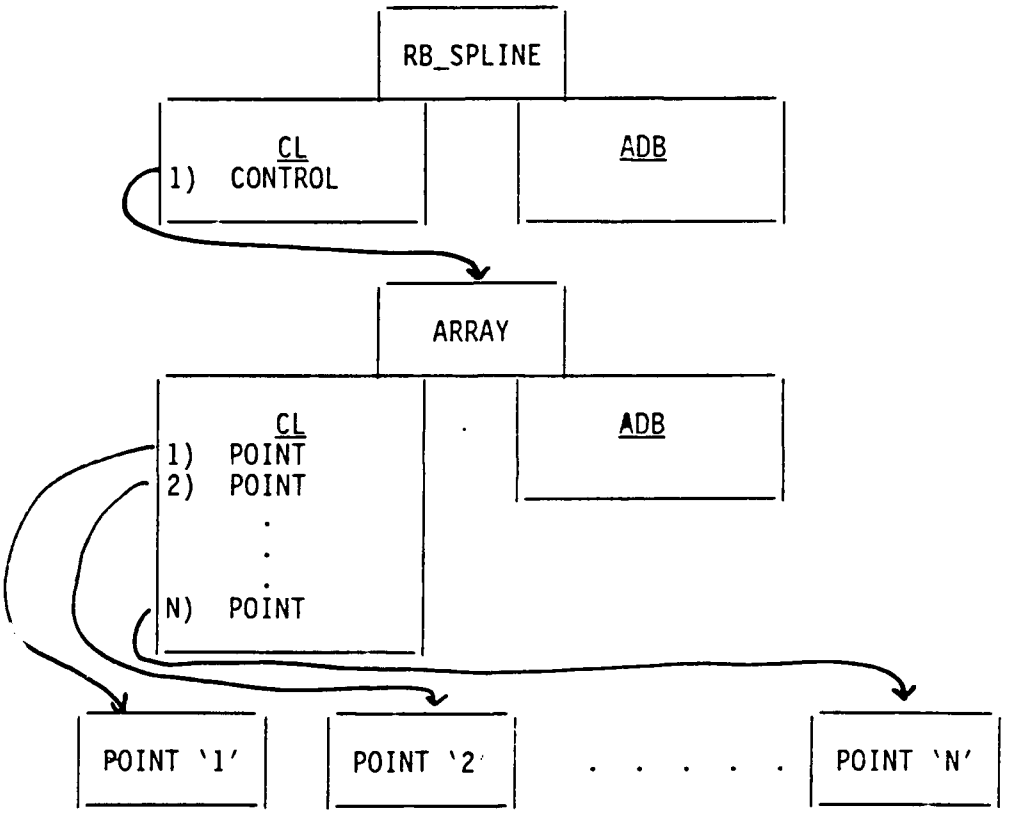
The location of data in storage for an ARRAY attribute is identical to Pascal stores arrays. For multidimensional arrays data is stored so that the Depth index is exhausted first, the Column index is exhausted second, and the Row index is exhausted last.

The REAL type data corresponding to this DIAMETER attribute would reside in the ADB as shown below.

	ADB displacement	Pascal Index	Fortran Index
	24	(1,1,1)	(1,1,1)
	28	(1,1,2)	(2,1,1)
	32	(1,1,3)	(3,1,1)
	36	(1,1,4)	(4,1,1)
	40	(1,2,1)	(1,2,1)
	44	(1,2,2)	(2,2,1)
	48	(1,2,3)	(3,2,1)
	52	(1,2,4)	(4,2,1)
	56	(1,3,1)	(1,3,1)
	.	.	.
	.	.	.
	68	(1,3,4)	(4,3,1)
	72	(2,1,1)	(1,1,2)
	76	(2,1,2)	(2,1,2)
	80	(2,1,3)	(3,1,2)
	84	(2,1,4)	(4,1,2)
	88	(2,2,1)	(1,2,2)
	.	.	.
	.	.	.
	.	.	.
(row,col,dep)	120	(2,3,4)	(4,3,2)

Arrays for attributes that reside in the CL are stored in ARRAY entities. The example below was taken from the definition of the RB_SPLINE entity.

```
Line 1 =>  CONTROL          ,14,14,  2, 1,7,  4,     1
Line 2 => X 2, 4000, 3001
Line 3 => A  2, 30
```



(2) ATTRIBUTE NAME field (16 character alpha)

```
-------------------------------------------------
*    (1)   (2)          (3)(4)(5)(6)(7)(8)     (9)   *
*        DIAMETER         , 9, 7,  1, 0,2,  4,    24  *
-------------------------------------------------
```

The name of the attribute entity.

(3) CONCEPTUAL SCHEMA ORDER field (2 digit numeric)

```
*    (1)   (2)          (3)(4)(5)(6)(7)(8)      (9)   *
*          DIAMETER       , 9, 7,  1, 0,2,  4,    24  *
```

- This field indicates the order of this attribute in the formal specification of this entity. It is this published conceptual schema order (CS order) that the entity attributes will follow in the Exchange Format file.

(4) PHYSICAL SCHEMA ORDER field (2 digit numeric)

```
*    (1)   (2)          (3)(4)(5)(6)(7)(8)      (9)   *
*          DIAMETER       , 9, 7,  1, 0,2,  4,    24  *
```

- This field indicates the order of this attribute in the computer memory resident entity. The mapping of attributes from the Conceptual Schema (CS) to the working form entity is not a direct mapping. The CS attributes to be put in the ADB reside at a position in the ADB that provides optimal use of space. The mapping rules to determine a CS's attribute location in the ADB are as follows:

  - Attributes that reside on double word boundaries (8 byte REALS) are first.

  - Attributes that must reside on single word boundaries (4 byte REALS, and 4 byte INTEGERS) are second.

  - Attributes that must reside on half word boundaries (2 byte INTEGERS) are third.

  - Attributes that must reside one byte boundaries (1 byte INTEGERS, LOGICALS, ENUMERATIONS, and STRINGS) are fourth.

(5) MINIMUM OCCURRENCES field (3 digit numeric)

```
*    (1)   (2)          (3)(4)(5)(6)(7)(8)      (9)   *
*          DIAMETER       , 9, 7,  1, 0,2,  4,    24  *
```

- An integer number signifying the minimum amount of data that can be stored for this attribute in increments of size.

A zero (0) for this field implies that this attribute is optional. The type code (1-6) signifies that this attribute is ADB type data, zeros (0) or blank ( ) may be stored. The type code (7 or 8) signifies that this is CL type data, the entity reference in the CL may point to a NIL entity.

(6) ARRAY DIMENSION field (2 digit numeric)

```
-------------------------------------------------
*    (1)   (2)        (3)(4)(5)(6)(7)(8)     (9)   *
*       DIAMETER       , 9, 7,  1, 0,2,  4,    24  *
-------------------------------------------------
```

An integer number signifying the dimension of the array of ADB or CL data. Zero (0) implies only one instance of data.

(7) TYPE CODE field (1 digit numeric)

```
-------------------------------------------------
*    (1)   (2)        (3)(4)(5)(6)(7)(8)     (9)   *
*       DIAMETER       , 9, 7,  1, 0,2,  4,    24  *
-------------------------------------------------
```

- The TYPE CODE field signifies the type of data this attribute contains and what part of the entity it resides in, either the ADB or CL. The types are;

1=> INTEGER
2=> REAL
3=> CHARACTER
4=> LOGICAL
5=> SCALAR
6=> SET
7=> POINTER
8=> SUBENTITY
9=> STRUCTURE

- Data types 1 through 6 reside in the ADB of the entity.
  Data types 7 and 8 reside in the CL of the entity.
  Data type 9 is a special type that gives location information about the attributes it encompasses.

- Below is a detailed explanation of each of the data types;

1 - INTEGER
  A 1, 2, or 4 byte integer.
  Resides on a single byte boundary, double byte boundary or a full word boundary. The example below was taken from the definition of the entity IMPL_B_HOLE.

(7) TYPE CODE field (Cont.)

```
KIND            , 1, 1,  1, 0,1,  4,     0
```

2 - REAL
A 4 or 8 byte real.
Resides on a full word boundary or a double word boundary. The example below was taken from the definition of the entity IMPL_B HOLE.

```
DIAMETER        , 9, 7,  1, 0,2,  4,    24
```

3 - CHARACTER
The character resides in 1 byte of storage and the SIZE field signifies how many characters are present. No boundary alignment. The example below was taken from the definition of the entity DETAIL_MODEL.

```
FSCM_CODE       ,10,10,  1, 0,3,  5,    55
```

4 - LOGICAL
A 1 byte integer such that 0 => FALSE and 1 => TRUE.
No boundary alignment. The example below was taken from the definition of the entity POINT_VECTOR.

```
DISPLAYED       , 8, 8,  1, 0,4,  1,    24
```

5 - SCALAR
A 1 byte integer (0 - 255) that indexes to the scalar stored. The scalar names are enumerated latter in the attribute record.

See example below. No boundary alignment.

```
APPLICATION     ,18,18,  1, 0,5,  1,    36
x 3,BILATERAL         ,INSIDE                ,OUTSIDE
```

If "INSIDE" was the scalar to be represented, a 1 would be located at the 36th byte in the ADB.

6 - SET
Not incorporated.

7 - POINTER
Attributes of this type signify that a reference resides in the CL. The kinds of entities that this attribute can reference are enumerated latter in the attribute record. The example below was taken from the definition of the entity CIRCULAR_RUNOUT.

(7) TYPE CODE field (Cont.)

```
TOLERANCE_ENTITY,13,13,  1, 1,7,  4,     1
X 4,11000, 8004, 8002, 20000
A  1,254
```

TOLERANCE_ENTITY is the first reference in the CL. Four different kinds of entities can be referenced.

8 - SUBENTITY
Attributes of this type are the same as attributes of type POINTER. The SUBENTITY code signifies that the type of entity pointed to was created as a result of the physical implementation of the Conceptual Schema. Entities of this type originally were structures in the definition of this entity. IMPL_B_HOLE is an example of a subentity. Its original structure was the BLIND HOLE Conceptual Schema entity.

9 - STRUCTURE
If in the Conceptual Schema definition of an entity a set of attributes are related to one another, then a STRUCTURE is used to bracket those attributes into a group. In the data dictionary a STRUCTURE type precedes and follows this set of attributes. The example below was taken from the Conceptual Schema surface entity - Parametric Ruled Surface (PRS).

```
BTUTABLE          ,15,15,  0, 1,9,  0,    40
 TVALUE           ,16,16,  1, 0,2,  4,    40
 UVALUE           ,17,17,  1, 0,2,  4,    44
ETUTABLE          ,18,18,  0, 1,9,  8,   296
A  0, 32
```

The attribute type STRUCTURE supplies information on the location of the array of attributes that are bracketed. The type data corresponding to the TVALUE, and UVALUE attributes would reside in the ADB as shown below.

ADB Displacement	TVALUE Index	UVALUE Index
40	1	-
44	-	1
48	2	-
52	-	2
56	3	-
60	-	3
65	4	-
68	-	4
.	.	.
.	.	.
.	.	.
292	32	-
296	-	32

(8) SIZE field (3 digit numeric)

```
-------------------------------------------------
*   (1)   (2)          (3)(4)(5)(6)(7)(8)    (9)  *
*         DIAMETER      , 9, 7,  1, 0,2,  4,   24  *
-------------------------------------------------
```

- An integer number signifying the number of bytes that one instance of this attribute takes in storage.

```
#DIAMETER          , 9, 7,  1, 0,2,  4,   24
```

(9) ADB/CL DISPLACEMENT field (6 digit numeric)

```
-------------------------------------------------
*   (1)   (2)          (3)(4)(5)(6)(7)(8)    (9) *
*         DIAMETER      , 9, 7,  1, 0,2,  4,   24  *
-------------------------------------------------
```

- An integer number signifying the starting location in the ADB of the entity for this attribute or the location of this reference in the CL.

ADB - An integer number signifying the starting location in the ADB of the entity for this attribute. The example below was taken from the definition of the entity IMPL_B_HOLE.

```
DIAMETER          , 9, 7,  1, 0,2,  4,  24
```

CL - An integer number signifying the location of this reference in the CL. The domain of this attribute type is enumerated on a continuation line following the attribute. The example below was taken from the definition of the entity CIRCULAR_RUNOUT.

```
TOLERANCE_ENTITY,13,13,  1, 1,7,  4,    1
X 4,11000, 8004, 8002, 2000
A  1,254
```

GETDD

This routine locates entity definitions with the Direct Access File.

```
%PAGE                                                                    00010000
(* BEGIN %INCLUDE GETDD ************************************************)00020000
(*                                                                       *)00030000
PROCEDURE GETDD ( CONST  KIND             : INTEGER;                     00040000
                  CONST  MAX_AVAIL        : INTEGER;                     00050000
                  CONST  ATTRIBUTE_ORDER  : CHAR;                        00060000
                  VAR    USER_ARRAY       : T_USER_ARRAY;                00070000
                  VAR    MAX_ACTUAL       : INTEGER;                     00080000
                  VAR    RETURN_CODE      : INTEGER );                   00090000
        EXTERNAL;                                                        00100000
(*                                                                       *)00110000
(*   $FUNCTION:                                                          *)00120000
(*        READ THE DATA DICTIONARY INTO THE APPLICATION PROGRAM.         *)00130000
(*                                                                       *)00140000
(*   $DESCRIPTION OF ARGUMENTS:                                          *)00150000
(*        NAME                 I/O  DESCRIPTION                          *)00160000
(*        ====                 ===  ===========                          *)00170000
(*        ATTRIBUTE_ORDER       I   SPECIFICATION OF THE ORDER FOR       *)00180000
(*                                  ATTRIBUTES IN THE ENTITY             *)00190000
(*                                  DEFINITION                           *)00200000
(*        KIND                  I   A KIND NUMBER OF ENTITY              *)00210000
(*        MAX_ACTUAL            O   AN ACTUAL NUMBER OF RECORDS IN       *)00220000
(*                                  ENTITY DEFINITION                    *)00230000
(*        MAX_AVAIL             I   A NUMBER OF 80 CHARACTER RECORDS     *)00240000
(*                                  AVAILABLE IN CALLER TO HOLE          *)00250000
(*                                  ENTITY DEFINITION                    *)00260000
(*        USER_ARRAY            O   AN ENTITY DEFINITION                 *)00270000
(*        RETURN_CODE           O   RETURN CODE                          *)00280000
(*                                  -1 = ACTUAL SIZE GREATER THAN        *)00290000
(*                                       SPACE AVAILABLE                 *)00300000
(*                                   0 = SUCCESS                         *)00310000
(*                                   1 = KIND NOT IN DATA DICTIONARY     *)00320000
(*                                                                       *)00330000
(*   $COMMONS:                                                           *)00340000
(*                                                                       *)00350000
(*   $ENVIRONMENT:                                                       *)00360000
(*        LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)                      *)00370000
(*        HARDWARE SYSTEM: IBM 360/370/4341/4381                         *)00380000
(*                                                                       *)00390000
(*   $EXECUTION PROCEDURE:                                               *)00400000
(*        CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM       *)00410000
(*                                                                       *)00420000
```

```
(*   $PROCESSING DESCRIPTION:                                                *)00430000
(*        LOOP THROUGH DATA DICTIONARY INDEX FILE                            *)00440000
(*          IF KIND IN DATA DICTIONARY THEN                                  *)00450000
(*             GET ENTITY DEFINITION FROM DDFILE                             *)00460000
(*             FILL UP THE ARRAY OF ENTITY DEFINITIONS UP TO NUMBER          *)00470000
(*             OF RECORDS AVAILABLE IN CALLER                                *)00480000
(*          END IF                                                           *)00490000
(*        END LOOP                                                           *)00500000
(*                                                                           *)00510000
(*   $COMMENTS:                                                              *)00520000
(*                                                                           *)00530000
(*   $CHANGE CONTROL:                                                        *)00540000
(*        ORIGINATED: 23 MARCH 1987, M. H. CHOI, DBMA                        *)00550000
(*                                                                           *)00560000
(* END %INCLUDE GETDD  ***************************************************)00570000
```

APPENDIX B

PRINTOUT EXAMPLE

This appendix provides an example of a Working Form printout. This is a printout of a Sample Model which has been post-processed into the Working Form.

```
**********************************
**********************************
*  ENTITY NAME : TEXT_LINE       *
*     9 ENTITIES EXIST           *
**********************************
**********************************
-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :          1302
LENGTH            :           158
SYSUSE            :             0
VERSION           :             1
SYS_IDENT         :             0
IDENT             :             8
LINE_LEN          :            71
A_LINE            : SHOT PEEN PER FSB 7777 WHERE INDICATED USING S22 DAE MAX CAS
THIS ENTITY HAS NO CONSTITUENTS

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   154
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   153
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF   146
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------

-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :          1302
LENGTH            :           158
SYSUSE            :             0
VERSION           :             1
SYS_IDENT         :             0
IDENT             :             7
LINE_LEN          :            71
A_LINE            : SHOT WITH INTENSITY T3 U05. DIM ARE BEFORE U05. PEENING MUST
THIS ENTITY HAS NO CONSTITUENTS

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   154
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   153
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF   145
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------
```

```
************************************
************************************
*  ENTITY NAME : LINE              *
*    41 ENTITIES EXIST             *
************************************
************************************
-------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :         5008
LENGTH            :           30
SYSUSE            :            0
VERSION           :            0
SYS_IDENT         :           41
IDENT             :           18
DISPLAYED         :       TRUE
RBG_LEVEL       (  1) :              0
RBG_LEVEL       (  2) :              0
RBG_LEVEL       (  3) :              0
INTENSITY         :            0
SYMBOL            :            1

THE CONSTITUENTS OF THIS ENTITY ARE:
P0                POINTS AT AN ENTITY OF KIND  4001 WITH AN IDENT OF    86
P1                POINTS AT AN ENTITY OF KIND  4001 WITH AN IDENT OF    31

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1201 WITH AN IDENT OF    64
USER HAS A KIND VALUE OF  8002 WITH AN IDENT OF    25
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   139
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   138
USER HAS A KIND VALUE OF  9020 WITH AN IDENT OF     1
-------------------------------------------------

-------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :         5008
LENGTH            :           30
SYSUSE            :            0
VERSION           :            0
SYS_IDENT         :           40
IDENT             :           41
DISPLAYED         :       TRUE
RBG_LEVEL       (  1) :              0
RBG_LEVEL       (  2) :              0
RBG_LEVEL       (  3) :              0
INTENSITY         :            0
SYMBOL            :            1

THE CONSTITUENTS OF THIS ENTITY ARE:
```

```
P0              POINTS AT AN ENTITY OF KIND  4001 WITH AN IDENT OF    87
P1              POINTS AT AN ENTITY OF KIND  4001 WITH AN IDENT OF    86

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1201 WITH AN IDENT OF    63
USER HAS A KIND VALUE OF  8002 WITH AN IDENT OF    64
USER HAS A KIND VALUE OF  9002 WITH AN IDENT OF    15
-----------------------------------------------------
```

```
************************************
************************************
*  ENTITY NAME : SIZE              *
*    42 ENTITIES EXIST             *
************************************
************************************
-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :         9003
LENGTH            :           40
SYSUSE            :            0
VERSION           :            1
SYS_IDENT         :            0
IDENT             :            1
DISPLAYED         :      FALSE
RBG_LEVEL       (  1) :            0
RBG_LEVEL       (  2) :            0
RBG_LEVEL       (  3) :            0
INTENSITY         :            1
SYMBOL            :            0
PLUS_TOL          :  1.4999996870756E-02
MINUS_TOL         :  1.4999996870756E-02

THE CONSTITUENTS OF THIS ENTITY ARE:
TOLERANCE_ENTITY POINTS AT AN ENTITY OF KIND  1100 WITH AN IDENT OF   156

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF     2
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF    37
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------

-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :         9003
LENGTH            :           40
SYSUSE            :            0
VERSION           :            1
SYS_IDENT         :            0
IDENT             :           19
DISPLAYED         :      FALSE
RBG_LEVEL       (  1) :            0
RBG_LEVEL       (  2) :            0
RBG_LEVEL       (  3) :            0
INTENSITY         :            1
```

```
SYMBOL          :            0
PLUS_TOL        :  9.9999979138374E-03
MINUS_TOL       :  9.9999979138374E-03

THE CONSTITUENTS OF THIS ENTITY ARE:
TOLERANCE_ENTITY POINTS AT AN ENTITY OF KIND  1100 WITH AN IDENT OF   126

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF    37
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF   187
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------
```

```
**************************/ **********
************************************
*  ENTITY NAME : EDGE              *
*    64 ENTITIES EXIST             *
************************************
************************************
-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :          8002
LENGTH            :            30
SYSUSE            :             0
VERSION           :             1
SYS_IDENT         :             0
IDENT             :            57
DISPLAYED         :         FALSE
RBG_LEVEL       (  1) :             0
RBG_LEVEL       (  2) :             0
RBG_LEVEL       (  3) :             0
INTENSITY         :               1
SYMBOL            :               0

THE CONSTITUENTS OF THIS ENTITY ARE:
CRVREF            POINTS AT AN ENTITY OF KIND  5002 WITH AN IDENT OF    20
VERTO             POINTS AT AN ENTITY OF KIND  8001 WITH AN IDENT OF    61
VERT1             POINTS AT AN ENTITY OF KIND  8001 WITH AN IDENT OF    55

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   156
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF    62
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF     1
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF    33
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF    24
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------

-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :          8002
LENGTH            :            30
SYSUSE            :             0
VERSION           :             1
```

```
SYS_IDENT          :              0
IDENT              :              1
DISPLAYED          :      FALSE
RBG_LEVEL       (  1) :                  0
RBG_LEVEL       (  2) :                  0
RBG_LEVEL       (  3) :                  0
INTENSITY          :              1
SYMBOL             :              0

THE CONSTITUENTS OF THIS ENTITY ARE:
CRVREF             POINTS AT AN ENTITY OF KIND  5008 WITH AN IDENT OF     1
VERT0              POINTS AT AN ENTITY OF KIND  8001 WITH AN IDENT OF     8
VERT1              POINTS AT AN ENTITY OF KIND  8001 WITH AN IDENT OF     9

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF     3
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF    35
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------
```

```
************************************
************************************
*  ENTITY NAME : MATERIAL          *
*     1 ENTITIES EXIST             *
************************************
************************************
-----------------------------------------------------
THE ADB OF THIS ENTITY IS :
KIND              :        10006
LENGTH            :          124
SYSUSE            :            0
VERSION           :            1
SYS_IDENT         :            0
IDENT             :            1
ENVELOPE        (  1) :  7.1469993591309E+00
ENVELOPE        (  2) :  5.5524997711182E+00
ENVELOPE        (  3) :  2.1771987915039E+01
MATERIAL_NAME     : NICKEL ALLOY
MATERIAL_TITLE    : XXXX
MATERIAL_CODE     :.XXXX
PART_NUMBER       : FORGING
PART_VERSION      : 001
FSCM_CODE         : UTRC
CONDITION_AFTER   : SAME
STOCK_STRUCTURE   : XXXX

THE CONSTITUENTS OF THIS ENTITY ARE:
NOTES             POINTS AT AN ENTITY OF KIND  1307 WITH AN IDENT OF     1
SPECIFICATIONS    POINTS AT AN ENTITY OF KIND  1307 WITH AN IDENT OF     1

THE USERS OF THIS ENTITY ARE:
USER HAS A KIND VALUE OF  1100 WITH AN IDENT OF   143
USER HAS A KIND VALUE OF  1301 WITH AN IDENT OF   153
USER HAS A KIND VALUE OF  1309 WITH AN IDENT OF     1
-----------------------------------------------------
```

APPENDIX C

SUPPLEMENTARY USER INTERFACE

TABLE OF CONTENTS

APPENDIX C

SUPPLEMENTARY USER INTERFACE

INTRODUCTION

This appendix describes an additional user-interface called the panel user-interface for running the PDES translator. Both user-interfaces perform the same function, however the panel user-interface uses ISPF dialog services for data entry.

OPERATION

To initiate the PDES translator using the Panel User Interface enter:

PDESPANL

after the system READY prompt. The translator will respond by displaying the data entry panel shown in Figure C-1. The first time PDESPANL is run for a given user, all the fields will be blank. On each subsequent invocation the fields will contain the values entered in the previous session. The old values may then be edited for this session.

The cursor movement keys may be used to move from one field to the next. The fields need not be entered in any particular order. Not all the fields are required for each application. Values in the unused fields are ignored, and saved for editing during the next invocation. If a required field is omitted or is invalid the cursor will automatically be moved to that field(s) and allows correction of the value. To determine which fields are required press ENTER on a partially filled data entry screen. This will cause a successive prompted for each required field until all the fields are valid. Once all the applicable fields are filled in press ENTER to begin the translator.

If at any time the user wants to terminate the translator he may enter END on the command line or press the END key (originally PF3).

TSO commands may be entered on the command line by prefixing them with the keyword: TSO.

The specifics of each field on the data entry panel are discussed in the next section.

A quick on-line reference to each field is available by entering HELP on the command line or hitting the help key (originally PF1).

The ISPF command KEYS may be entered on the command line to display the current settings of the PF keys, and allow the user to change the settings.

```
      WORKING FORM/EXCHANGE FORMAT (PDES) TRANSLATOR V1.0    NOVEMBER 4
COMMAND ===>                                                      12:05

Type the information where requested, or change the information shown by
typing over it.  Press ENTER to submit or END to quit.

     Processor (POST or PRE     ===>
     EF file on DISK or TAPE    ===>
       DISK:  EF file name      ===>
       TAPE:  Tape name         ===>
              Label number      ===>
     Keeping mapping file (Y/N ===>
       Mapping file name        ===>
     Internal (I) or PDDI Supplied (P) Data base ===>
     Batch or Interactive       ===>
       Batch account number     ===>
       User ID number           ===>
       User last name           ===>
     Post processor working
       Form printout (Y/N)      ===>
       WF printout file name    ===>
```

Figure C-1 PDESPANL Data Entry Panel

FIELD DEFINITIONS

Processor (POST or PRE)

This field specifies the pre-processor or the post-processor. Enter either the keyword PRE or POST. The PRE-processor, uses a model in the working form and creates a PDES Exchange Format. The POST-processor creates a working form model from a PDES exchange format. This is a required field.

EF on DISK or TAPE

This field determines the location of the Exchange Format file that the PRE-processor creates, or the POST-processor uses to create a working form model. Enter either the keyword DISK or TAPE. This is a required field. Note that tapes cannot be accessed interactively, and must be translated as a batch process.

EF file name

This field specifies the name of the disk file that is used when the Exchange Format file is written (PRE-processor) or is read (POST-processor). If the user is running the POST-processor, the file name will be verified that it exists. The PRE-processor does not check to see if the file exists and will write it over existing files of the same name. This is a required field when DISK is specified.

Tape name

This field specifies the name of the tape for the Exchange Format file. This is a required field when the keyword TAPE is specified.

Label number

This field specifies the position number on the tape where the Exchange Format is located on the TAPE. This is a required field if TAPE was specified.

Keep mapping file (Y/N)

This field specifies whether a mapping file is to be produced. A mapping file correlates entity kind and sys ident numbers in the Working Form model with the entity identifier numbers in the Exchange Format file. This is a required field.

Mapping file name

This field specifies the mapping file name. The name of the disk file to which the mapping file will be written must be specified. The mapping file can only be written to a disk file.

Internal (I) or PDDI (P) Supplied Database

Specify "P" to file the interim database supplied with the PDDI software.

Batch or Interactive

This field specifies batch or interactive translation. Enter either B for Batch mode, or I for Interactive mode. This is a required field. Note that tape files can not be accessed in interactive mode.

Batch account number

This field specifies the account number for batch processing to be inserted on the on JOB card in the JCL for the batch process.

User ID number

This field specifies the system ID number of the person you want to be able to retrieve the output from a batch translation. *This field is not required.* If is is omitted, it will default to the ID of the user who is submitting the batch translation.

User last name

This field is used to determine the distribution name in the JCL for a batch job. The user's last name (possibly with additional letters appended to make it unique), or a group ID should be used. This is a required field for all batch jobs.

Post-processor working form printout (Y/N)

This field specifies whether a post-processor printout is desired. The option exists for printing out some or all of the entities that were translated to the working form model.

If the user specified Y and is running the processor as a batch process, the Working Form data for all entities will be printed. If the user is running the POST-processor interactively and specifies Y, the system will prompt the user for either ALL or KIND after it has translated the entities . If the user specifies ALL, the Working Form data for all entities will be printed. If the user selects KIND, the user will be prompted for the kind number of the desired entity until a null (blank) response is given.

The Working Form data for the entities with those kinds will then be printed. This is a required field for the POST-processor and is ignored for the PRE-processor.

WF printout file name

This field specifies the name of the disk file to which the Working Form printout is to be written. This field is required only when running the POST processor, and a working form printout has been selected above.

PDDI SUPPLIED DATABASE PANEL

The panel shown in Figure C-2 will be displayed when filing to the PDDI supplied Interim database. This panel prompts the user to specify the name of the hex file to be used for filing/retrieving a drawing.

COMMENTS/EXPLANATIONS PANEL

When running the PRE processor (Working Form to PDES Exchange Format) the panel shown in Figure D-2 will be displayed. This panel allows the user to enter additional comments or explanations beyond those that the CLIST itself generates.

Comments are typed below the dashed line.

The character sequence '*)' should not be used any where in the text, since it is used to delimit the end of an exchange format file comment.

The character '&' should not be used because the CLIST processor will interpret it as part of a variable, and try to resolve it.

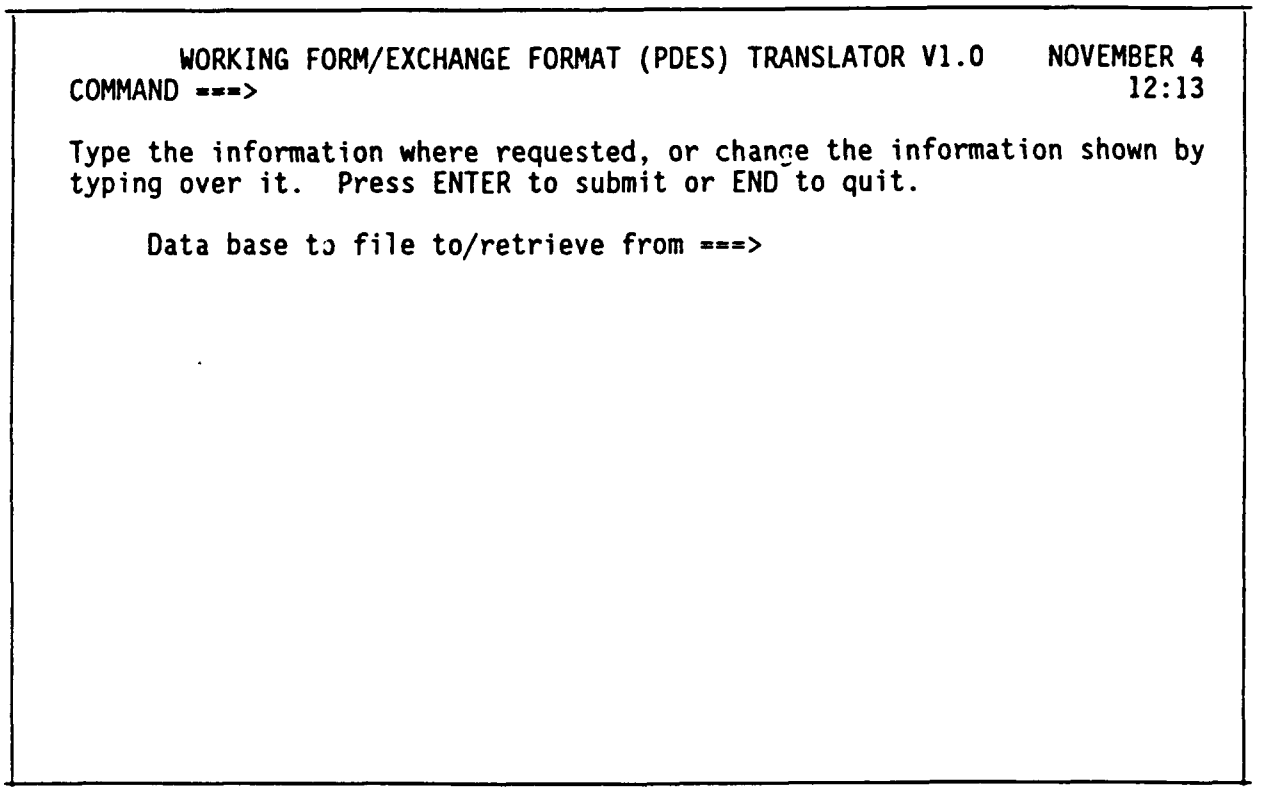


Figure C-2 PDDI Supplied Data Base Panel

Press ENTER to add the comments to the Exchange Format file produced. Another blank comments/explanations panel will be displayed for more text. The comments from successive panels will be added to the file with one blank line between them regardless of the number of lines of comments actually entered. To terminate press ENTER on a blank panel.

To terminate the translator enter END on the command line or press the END key (originally PF3).

A brief explanation on how to operate the comments/explanations panel can be obtained on-line by entering HELP on the command line or pressing the HELP key (originally PF1).

MESSAGES

As a result of a data entry error or an omission of a required field, messages will be displayed highlighted on the 3rd line of the screen below the command line. These messages are of two types: Error (E), and Directive (D). The list of possible messages is:

Msg ID	Type	Description
PHTM001	E	Invalid processor type - choose POST or PRE
PHTM002	D	The exchange format file name is required for disk files
PHTM003	E	The only valid responses are: Y or N
PHTM004	D	The mapping file name is required
PHTM005	E	Invalid File/Retrieval Key
PHTM006	D	File/Retrieval Key required for Preprocessor
PHTM007	E	The only valid responses are: B or I
PHTM008	D	The system accounting parameter is required for batch jobs
PHTM009	E	The system accounting parameter is invalid
PHTM010	E	File _____ does not exist
PHTM011	E	Invalid Cluster Name
PHTM012	D	Cluster name is a required field
PHTM013	D	Location of exchange format file is a required field
PHTM014	E	Enter either TAPE or DISK for location of exchange format
PHTM015	D	The tape name is a required field
PHTM016	D	The label number is a required field
PHTM017	E	Label number is non-numeric
PHTM018	E	Tapes cannot be accessed interactively - run the translator batch
PHTM019	D	Enter a distribution name for batch job
PHTM020	D	Working form printout file name is required

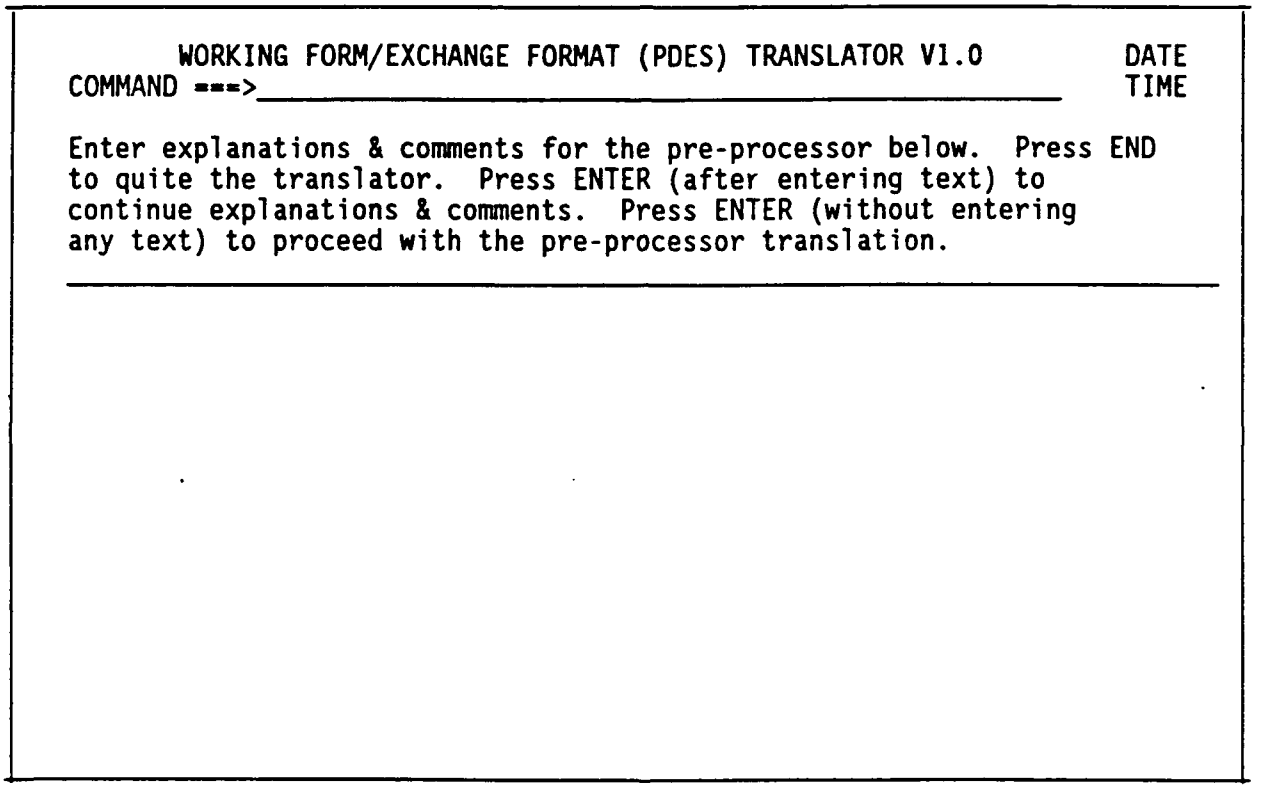


Figure C-3 Pre-processor Comments Entry Panel

APPENDIX D

INTERIM DATABASE SOFTWARE

OVERVIEW

The Interim Database Software will file a Working Form Access Software model to a sequential file or retrieve that file and recreate the model. It was written to be used by organizations interested in testing the PDDI Translator Software. This Interim Database Software will support both a hierarchical and a recursive PDDI model. This software is intended to be an interim database for the PDDI Translator until an interface to the organization's native database system is available.

Filing of the PDDI model is done in two passes. First, the model is read and all entity ADB's are written to the file. Second, the model is searched for those entities with constituents. If an entity is found to have a constituent, this information is written out to the file in order to preserve the relationship. (See figure D-1 and D-2)

Retrieval is also done in two passes. First, the ADB portion of the file is read and all entities, less constituents, are recreated. Second, the constituent portion of the file is read to determine constituent relationships which are then recreated through attachments. (See figure D-3 and D-4)

FILE STRUCTURE

The file is written out and read in as Hexadecimal using an "A" format in a Fortran format statement. (See Figure D-5)

Four different forms of records are written to the file. The first byte in each record determines the type of record. An "A" signifies that the record is the first record containing the information from an entity ADB. The "A" byte is followed by a four (4) byte "Identifier Number" that differentiates this entity instance from all other entity instances of the same entity kind. The "Identifier Number" is followed by up to seventy-five (75) bytes of ADB data.

An "M" signifies that the entity ADB was longer than seventy-five (75) types and is being continued on this record. The "M" byte is followed by up to seventy-nine (79) bytes of ADB data. The number of "M" records that follow an "A" record varies depending on the length of the ADB of a particular entity kind.

After all the ADB's of the model are written out to "A" and "M" records you may then find a "C" record depending on whether there are any entities in the model that have constituents. A "C" signifies that the record defines the constituents of an entity. In this record you will find a four (4) byte "Kind Number", a four (4) byte "Identifier Number", a four (4) byte "Constituent Count Number", and a set of up to eight (8), eight (8) byte "Kind Numoer "/" Identifier Number" data, in that order. This record defines the subject entity, the amount of constituents it contains and up to eight constituent entities.

If there are more than eight (8) constituent entities defined for the subject entity then one or more records may follow the "C" record. These records have no letter in the first byte identifying them and contain up to ten (10), eight (8) byte "Kind Number"/"Identifier Number" data that defines the constituents of the subject entity.

File Allocation

The sequential file will need to be allocated to the name "FT08F001" and have the following attributes:

Record Format : FB (Fixed Block)
Record Length : 80 bytes
Block Size : 6160 bytes

The number of blocks to allocate for this file is dependent upon the size of the model to be filed. It is a function of the number of entities in the model multiplied by the size of each of their ADB lengths and constituent lengths.

Software Languages

The Interim Database Software is written in IBM Pascal version 2.2 and IBM FORTRAN version IV. We attempted to write it in a language that is compatible with most computers.

Software Limitations

Several limitations exist in this software. We believe that these limitations will probably never be reached for most computer graphic models. These limitations were created to maintain the high performance levels of the software. These limitations are:

Maximum Allowable Instances for One Entity Kind : 5000
Maximum Allowable Entity Kinds : 500
Maximum Allowable ADB Length per Entity Instance : 45000 bytes
Maximum Allowable Model Size : Computer Region Alloc
Maximum Allowable File Size : Sequential File Alloc

FILE

STEP 1

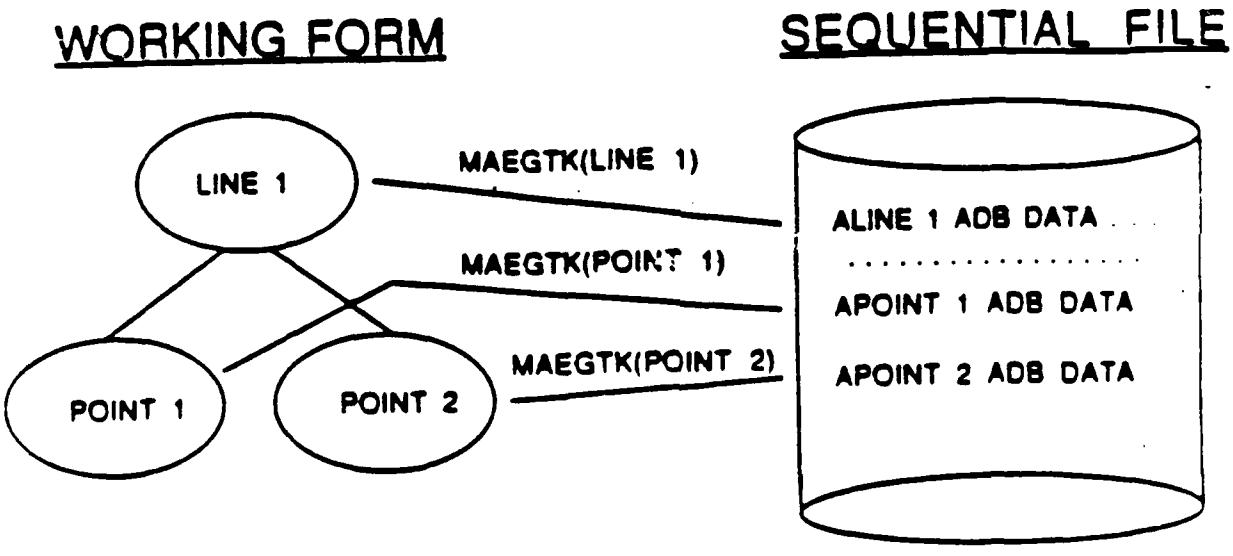


Figure D-1

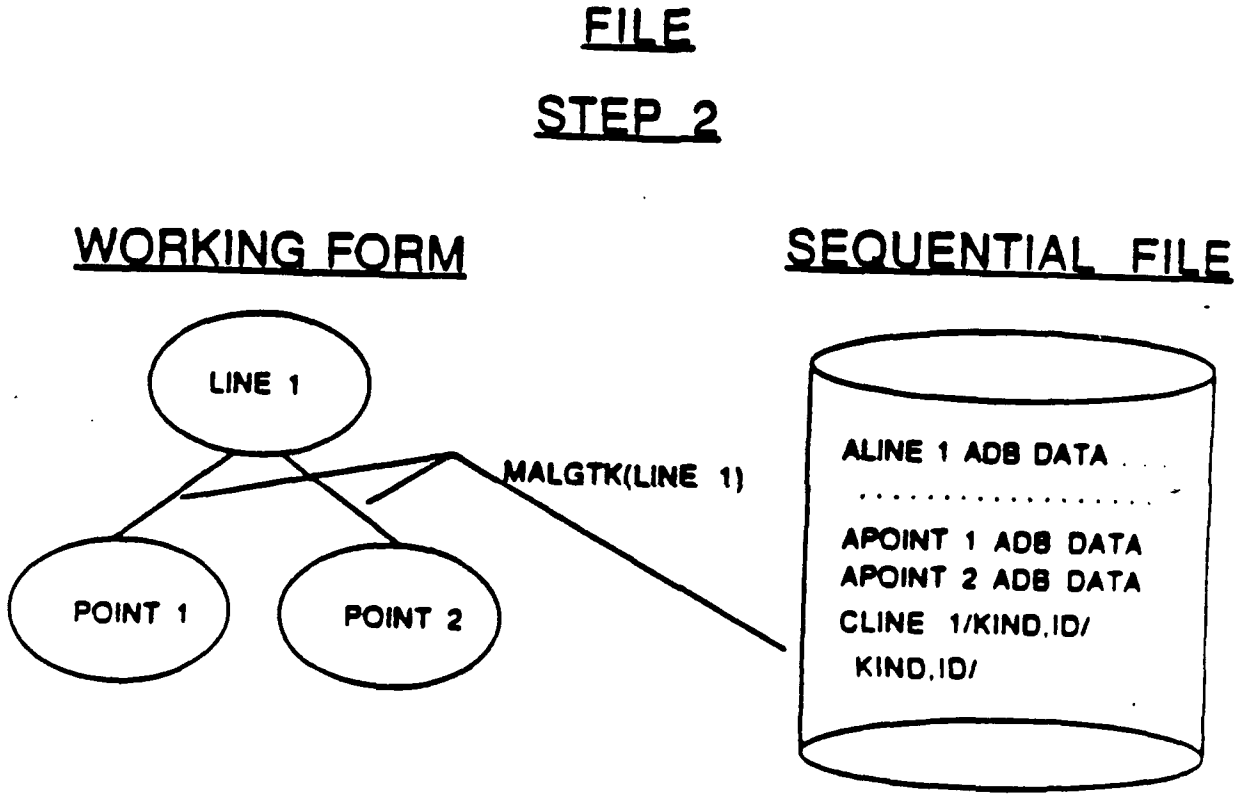

FILE
STEP 2
WORKING FORM
SEQUENTIAL FILE
LINE 1
POINT 1
POINT 2
MALGTK(LINE 1)
ALINE 1 ADB DATA . . .
. . . . . .
APOINT 1 ADB DATA
APOINT 2 ADB DATA
CLINE 1/KIND,ID/
KIND,ID/

Figure D-2

RETRIEVE

STEP 1

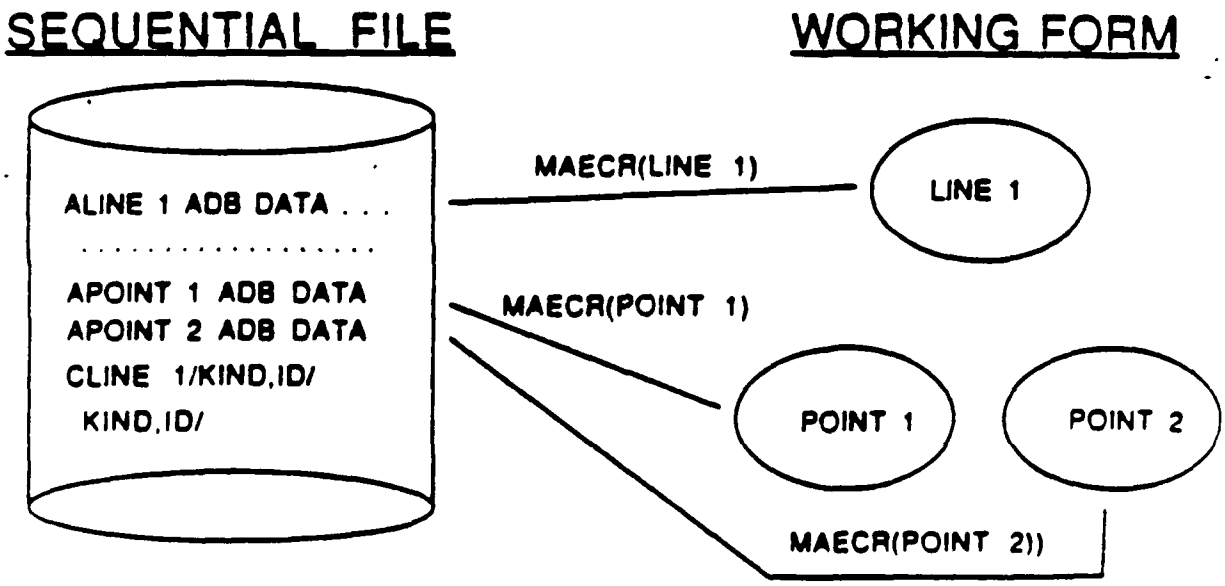


Figure D-3

RETRIEVE

STEP 2

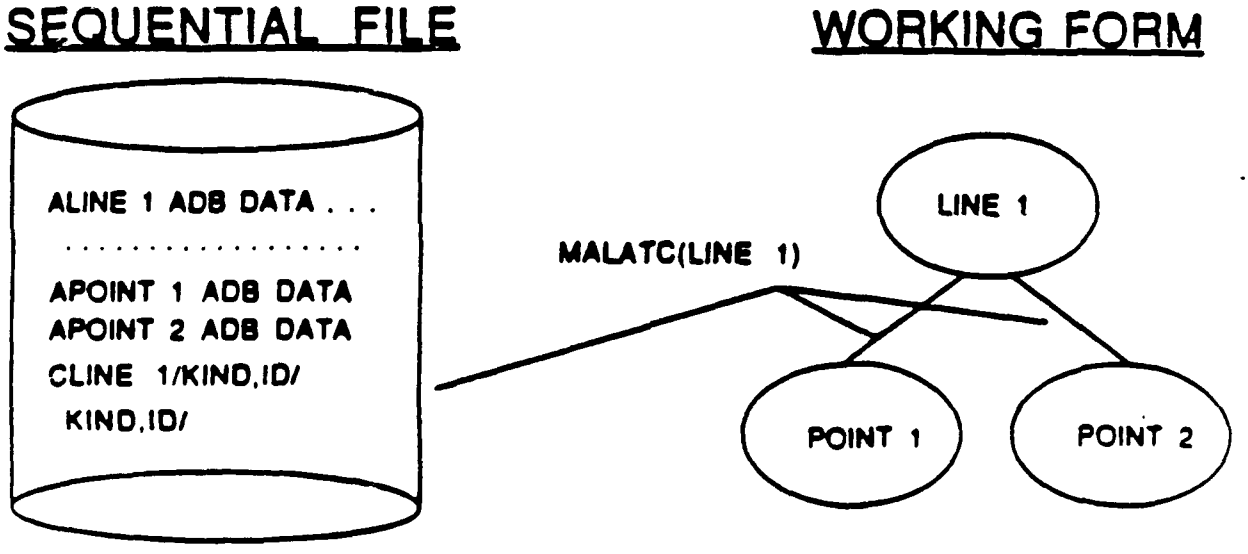


Figure D-4

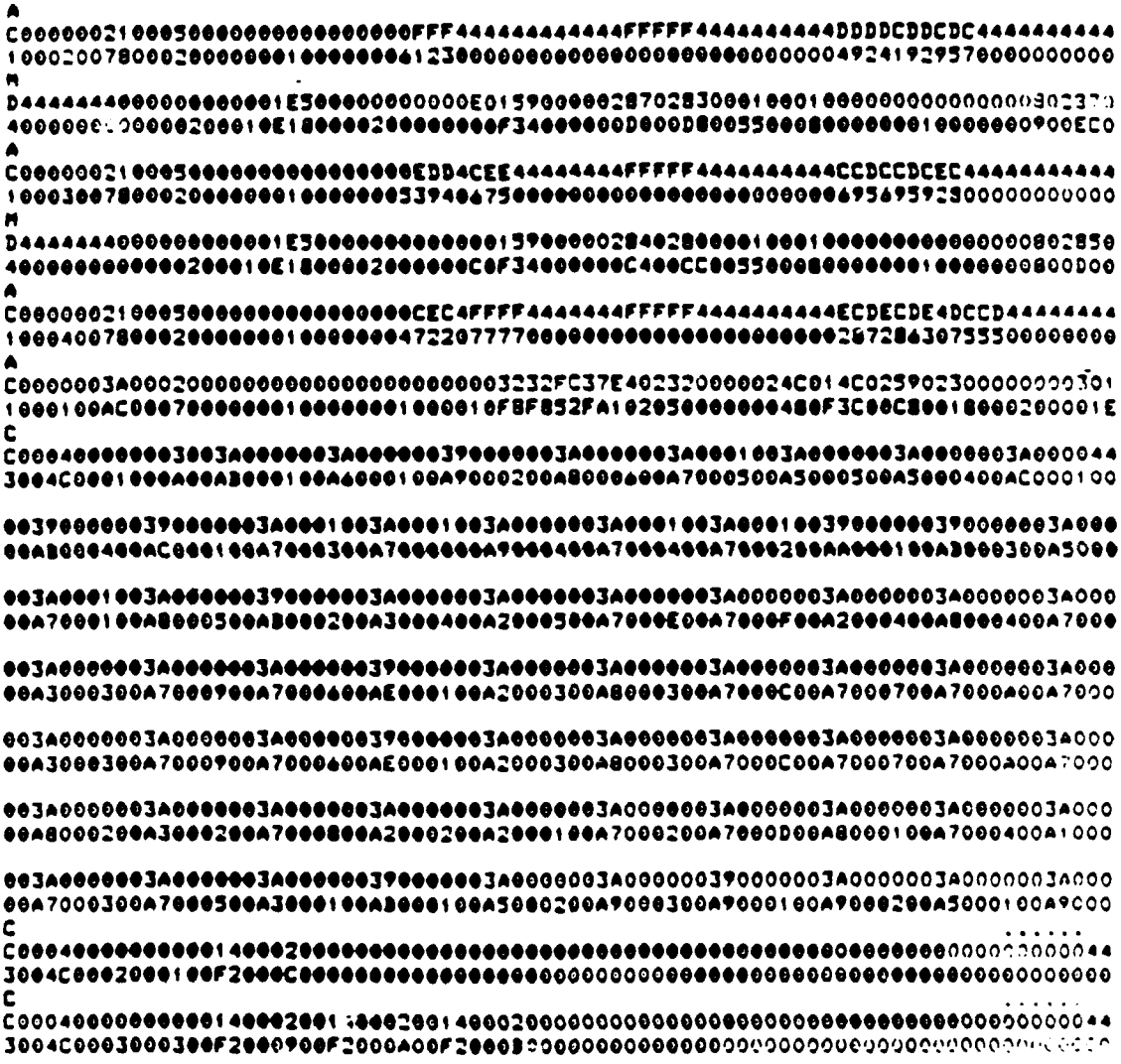


Figure D-5

Software Performance

The following are sample statistics generated from operations on the PDDI Sheet Metal Rib using a NAS (IBM 4381 look-a-like) computer.

Model Statistics

Number of entities in model	: 6433
Model Size	: 364,085 bytes

Interim Database Statistics

CPU time required to file the model	: 9.58 sec
CPU time required to retrieve the model	: 14.52 sec

PDDI Translator Statistics

CPU time required to Pre-Process the model	: 4 Min 34 Sec
Region Size required to Pre-Process the model	: 2208k bytes
CPU time required to Post-Process the model	: 4 Min 29 Sec
Region Size required to Post-Process the model	: 2148 bytes

Software Descriptions

```
**********
* FILRTV *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/16/86   *
C*             S. L. MADDERN                                          *
C*    VERSION:    1.0                            REVISED:             *
C*    ROUTINE NAME:   FILRTV                                          *
C*    FUNCTION:                                                       *
C*      THIS ROUTINE WILL EITHER TAKE THE INCORE WORKING FORM         *
C*      MODEL AND WRITE IT OUT TO A SEQUENTIAL FILE OR READ A         *
C*      SEQUENTIAL FILE OF A WORKING FORM MODEL AND PUT IT INTO       *
C*      AN INCORE WORKING FORM MODEL.                                 *
C*    ENVIRONMENT:                                                    *
C*      IBM PASCAL LANGUAGE                                           *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.      *
C*    EXECUTION PROCEDURE:                                            *
C*      CALLED BY    : MAIN PROGRAM                                   *
C*      CALLS        : ARTFID COMMON                                  *
C*                   : LASTAR COMMON                                  *
C*                   : ENTTMP COMMON                                  *
C*                   : LASTTP COMMON                                  *
C*                   : ENFILE                                         *
C*                   : ENRTV                                          *
C*    DESCRIPTION OF ARGUMENTS:                                       *
C*        FRFLAG    I A FLAG INDICATING WHETHER TO FILE OR RETRIEVE.  *
C*                    (0 = > FILE. 1 = > RETRIEVE, 2 = > MERGE)       *
C*        IRC       O THE MODEL WAS SUCCESSFULLY FILED OR RETRIEVED.  *
C*                    RETURN CODE (0 = > FILE/RETRIEVE SUCCESSFUL)    *
C*                    RETURN CODE (1 = > FILE/RETRIEVE UNSUCCESSFUL)  *
C*    PROCESSING DESCRIPTION:                                         *
C*        THIS ROUTINE NEEDS NO OPEN/CLOSE FILES                      *
C*    CHANGE CONTROL:                                                 *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                                *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF    *
C*                      THE ENTITIES BEING PROCESSED.                 *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE  *
C*                      "ARTFID" COMMON AREA.                         *
C*                    - "ENTTMP" HOLDS THE KEYS OF THE TEMPORARY      *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY    *
C*                      KINDS.                                        *
C*                    - "LASTTP" DEFINES THE LAST ARRAY INDEX OF THE  *
C*                      "ENTTMP" COMMON AREA.                         *
C*                    - "PIDSTA" RECORDS THE LAST LINE OF DATA THAT   *
C*                      HAS BEEN WRITTEN OR READ.                     *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* ENFILE *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/16/86   *
C*    VERSION:    1.0                            REVISED:             *
C*    ROUTINE NAME:   ENFILE                                          *
C*    FUNCTION:                                                       *
C*      FILES ANY ACCESS SOFTWARE CREATED WORKING FORM TO A           *
C*      SEQUENTIAL FILE.                                              *
C*    ENVIRONMENT:                                                    *
C*      IBM FORTRAN IV                                                *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.      *
C*    EXECUTION PROCEDURE:                                            *
C*      CALLED BY   : FILRTV                                          *
C*      CALLS       : WRTFIL                                          *
C*                  : ARTFID COMMON                                   *
C*                  : LASTAR COMMON                                   *
C*                  : ENTTMP COMMON                                   *
C*                  : LASTTP COMMON                                   *
C*                  : MAED                                            *
C*                  : MALD                                            *
C*                  : MALK                                            *
C*                  : CRETMP                                          *
C*                  : MAECTK                                          *
C*                  : MAEKND                                          *
C*                  : MAKXEQ                                          *
C*    DESCRIPTION OF ARGUMENTS:                                       *
C*        IRC  O RETURN CODE (0 = > FILE SUCCESSFUL)                  *
C*                           (1 = > FILE UNSUCCESSFUL)                *
C*    PROCESSING DESCRIPTION:                                         *
C*        THE FILE THAT IS WRITTEN TO IS FT08F001.                    *
C*    CHANGE CONTROL:                                                 *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                                *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF    *
C*                      THE ENTITIES BEING PROCESSED.                 *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE  *
C*                      "ARTFID" COMMON AREA.                         *
C*                    - "ENTTMP" HOLDS THE KEYS OF THE TEMPORARY      *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY    *
C*                      KINDS.                                        *
C*                    - "LASTTP" DEFINES THE LAST ARRAY INDEX OF THE  *
C*                      "ENTTMP" COMMON AREA.                         *
C*                    - "PIDSTA"                                      *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* ENRTV  *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/18/86   *
C*             S. L. MADDERN                                          *
C*    VERSION:    1.0                            REVISED:            *
C*    ROUTINE NAME:   ENRTV                                          *
C*    FUNCTION:                                                      *
C*      READS IN THE A WORKING FORM FILE FROM A SEQUENTIAL DATASET   *
C*      AND GENERATES THE CORRESPONDING INCORE WORKING FORM          *
C*    ENVIRONMENT:                                                   *
C*      IBM FORTRAN IV                                               *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.     *
C*    EXECUTION PROCEDURE:                                           *
C*      CALLED BY    : FILRTV                                        *
C*      CALLS        : SRCHAR                                        *
C*                   : ARTFID COMMON                                 *
C*                   : LASTAR COMMON                                 *
C*                   : ENTTMP COMMON                                 *
C*                   : LASTTP COMMON                                 *
C*                   : MAECR                                         *
C*                   : CRETMP                                        *
C*                   : GETTMP                                        *
C*                   : MAEXEQ                                        *
C*                   : MAINIT                                        *
C*                   : MAKILL                                        *
C*                   : MALATC                                        *
C*                   : REDFIL                                        *
C*    DESCRIPTION OF ARGUMENTS:                                      *
C*        IRC     O RETURN CODE                                      *
C*                  0 - SEQUENTIAL FILE RETRIEVAL WAS SUCCESSFUL     *
C*                  1 - SEQUENTIAL FILE RETRIEVAL WAS UNSUCCESSFUL   *
C*    PROCESSING DESCRIPTION:                                        *
C*        THIS ROUTINE NEEDS NO OPEN/CLOSE FILES                     *
C*    CHANGE CONTROL:                                                *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                               *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF   *
C*                      THE ENTITIES BEING PROCESSED.                *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ARTFID" COMMON AREA.                        *
C*                    - "ENTTMP" HOLDS THE KEYS OF THE TEMPORARY     *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY   *
C*                      KINDS.                                       *
C*                    - "LASTTP" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ENTTMP" COMMON AREA.                        *
C*                    - "PIDSTA"                                     *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* WRTFIL *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES              W315 2G CREATED:  6/24/85     *
C*    VERSION:    1.0                           REVISED:               *
C*    ROUTINE NAME:   WRTFIL                                           *
C*    FUNCTION:                                                        *
C*      WRITES THE ADB AND CONSTITUENT REFERENCES OF AN ENTITY TO      *
C*      A SEQUENTIAL FILE                                              *
C*    ENVIRONMENT:                                                     *
C*      IBM FORTRAN IV                                                 *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.       *
C*    EXECUTION PROCEDURE:                                             *
C*      CALLED BY    : ENFILE                                          *
C*      CALLS        : SRCHAR                                          *
C*                   : ARTFID COMMON                                   *
C*                   : LASTAR COMMON                                   *
C*                   : MALNO                                           *
C*                   : GETTMP                                          *
C*                   : MAEGKN                                          *
C*                   : MAEXEQ                                          *
C*                   : MALGTK                                          *
C*    DESCRIPTION OF ARGUMENTS:                                        *
C*        KEY      I KEY OF THE ENTITY BEING FILED                     *
C*        ADB      I ATTRIBUTE DATA BLOCK TO BE WRITTEN                *
C*        IFLAG    I FLAG INDICATING WHETHER THE ADB IS TO BE WRITTEN  *
C*                   (IFLAG = 0) OR THE CONSTITUENT RELATIONSHIP DATA  *
C*                   IS TO BE WRITTEN (IFLAG = 1).                     *
C*        IRX      O 0 TO 7 = > CONTINUE PROCESSING                    *
C*                   8 TO 15 = > ROUTINE ERROR HALT PROCESSING         *
C*                   -* TO -1 AND 15 TO * = > OUT OF RANGE ERROR HALT  *
C*    PROCESSING DESCRIPTION:                                          *
C*        THIS ROUTINE WRITES THE DATA TO THE FILE FT08F001.           *
C*    CHANGE CONTROL:                                                  *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                                 *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF     *
C*                      THE ENTITIES BEING PROCESSED.                  *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE   *
C*                      "ARTFID" COMMON AREA.                          *
C*                    - "ENTTMP" HOLDS THE KEYS OF THE TEMPORARY       *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY     *
C*                      KINDS.                                         *
C*                    - "LASTTP" DEFINES THE LAST ARRAY INDEX OF THE   *
C*                      "ENTTMP" COMMON AREA.                          *
C*                    - "PIDSTA"                                       *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* CRETMP *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/18/86   *
C*             S. L. MADDERN                                          *
C*    VERSION:    1.0                           REVISED:              *
C*    ROUTINE NAME:   CRETMP                                         *
C*    FUNCTION:                                                      *
C*      THIS ROUTINE CREATES A ENTITY OF KIND NUMBER "1" THAT CONTAINS*
C*      ALL THE ID'S/KEY'S STORED IN THE ARTFID COMMON AREA.  THIS    *
C*      STORED ARRAY OF ID'S/KEY'S ARE SORTED EITHER BY ID OR KEY     *
C*      DEPENDING WHETHER A FILE OR RETRIEVE IS IN PROCESS.           *
C*    ENVIRONMENT:                                                   *
C*      IBM FORTRAN IV                                               *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.     *
C*    EXECUTION PROCEDURE:                                           *
C*      CALLED BY   : ENFILE                                         *
C*      CALLS       : ENRTV                                          *
C*                  : ARTFID COMMON                                  *
C*                  : LASTAR COMMON                                  *
C*                  : ENTTMP COMMON                                  *
C*                  : LASTTP COMMON                                  *
C*                  : MAECR                                          *
C*                  : SORTAR                                         *
C*    DESCRIPTION OF ARGUMENTS:                                      *
C*        SRTFLG  I TRUE = > SORT BY KEY                             *
C*                  FALSE = > SORT BY ID                             *
C*        KIND    I KIND OF ENTITY ASSOCIATED WITH KEY'S IN COMMON   *
C*        IRC     O RETURN CODE                                      *
C*                  0 - TEMPORARY ENTITY CREATION SUCCESSFUL         *
C*                  1 - TEMPORARY ENTITY CREATION UNSUCCESSFUL       *
C*    PROCESSING DESCRIPTION:                                        *
C*        THIS ROUTINE NEEDS NO OPEN/CLOSE FILES                     *
C*    CHANGE CONTROL:                                                *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                               *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF   *
C*                      THE ENTITIES BEING PROCESSED.                *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ARTFID" COMMON AREA.                        *
C*                    - "ENTTMP" HOLDS THE KEYS OF THE TEMPORARY     *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY   *
C*                      KINDS.                                       *
C*                    - "LASTTP" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ENTTMP" COMMON AREA.                        *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* REDFIL *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/17/86   *
C*             S. L. MADDERN                                          *
C*    VERSION:    1.0                            REVISED:             *
C*    ROUTINE NAME:   REDFIL                                         *
C*    FUNCTION:                                                      *
C*      READS A RECORD OF A FILE THAT WAS GENERATED BY THE "INTERIM   *
C*      DATABASE" SOFTWARE.                                          *
C*    ENVIRONMENT:                                                   *
C*      IBM FORTRAN IV                                               *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.     *
C*    EXECUTION PROCEDURE:                                           *
C*      CALLED BY    : ENRTV                                         *
C*      CALLS        : FORTRAN READ                                  *
C*    DESCRIPTION OF ARGUMENTS:                                      *
C*        RECTYP  O A CHARACTER INDICATING WHETHER THE RECORD READ   *
C*                  IS OF TYPE "A" FOR INITIAL ADB RECORD,  "M" FOR  *
C*                  SUBSEQUENT ADB RECORD, "C" FOR INITIAL           *
C*                  CONSTITUENT LIST RECORD, OR "X" FOR SUBSEQUENT   *
C*                  CONSTITUENT LIST RECORD.                         *
C*        KIND    O KIND OF ENTITY DEFINED ON A "A" OR "C" RECORD    *
C*        ID      O ARTIFICIAL ID OF ENTITY DEFINED ON A "A" OR "C"  *
C*                  RECORD.                                          *
C*        NUMCON  O NUMBER OF CONSTITUENTS RECORDED ON THE "C" RECORD*
C*        RECDAT  O BYTES OF DATA ON RECORD PERTAINING TO THE ADB OF *
C*                  THE ENTITY (75 BYTES FROM "A" OR "M") OR THE     *
C*                  CONSTITUENTS OF THE ENTITY (80 BYTES FROM "C" OR *
C*                  "X")                                             *
C*                  "X")                                             *
C*        EOFFLG  O FLAG TO SIGNAL IF END OF FILE WAS REACHED        *
C*                  EOFFLG = 0 = > END OF FILE NOT REACHED           *
C*                  EOFFLG = 1 = > END OF FILE REACHED               *
C*    PROCESSING DESCRIPTION:                                        *
C*        THIS ROUTINE READS FROM FILE FT08F001                      *
C*    CHANGE CONTROL:                                                *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                               *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* GETTMP *
**********

C*--------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/18/86   *
C*             S. L. MADDERN                                          *
C*    VERSION:    1.0                            REVISED:             *
C*    ROUTINE NAME:   GETTMP                                         *
C*    FUNCTION:                                                      *
C*      THIS ROUTINE GETS THE KEY OF THE TEMPORARY ENTITY THAT IS    *
C*      ASSOCIATED WITH THE INPUT ENTITY KIND.  THE KEYS ARE GOTTEN  *
C*      FROM A COMMON AREA THAT HOLDS THE KEYS OF EACH OF THE        *
C*      TEMPORARY ENTITIES AND THEIR ASSOCIATED ENTITY KIND.         *
C*    ENVIRONMENT:                                                   *
C*      IBM FORTRAN IV                                               *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.     *
C*    EXECUTION PROCEDURE:                                           *
C*      CALLED BY    : WRTFIL                                        *
C*                   : ENRTV                                         *
C*      CALLS        : ENTTMP COMMON                                 *
C*                   : LASTTP COMMON                                 *
C*    DESCRIPTION OF ARGUMENTS:                                      *
C*        KIND     I KIND OF ENTITY ASSOCIATED WITH KEY'S IN COMMON  *
C*        KEY      O KEY OF THE ASSOCIATED TEMPORARY ENTITY          *
C*        IRC      O RETURN CODE                                     *
C*                   0 - TEMPORARY ENTITY CREATION SUCCESSFUL        *
C*                   1 - TEMPORARY ENTITY CREATION UNSUCCESSFUL      *
C*    PROCESSING DESCRIPTION:                                        *
C*        THIS ROUTINE NEEDS NO OPEN/CLOSE FILES                     *
C*    CHANGE CONTROL:                                                *
C*--------------------------------------------------------------------*
C*--------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                               *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF   *
C*                      THE ENTITIES BEING PROCESSED.                *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ARTFID" COMMON AREA.                        *
C*                    - "ENTTMP" HOLDS THE KEYS OF THE TEMPORARY     *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY   *
C*                      KINDS.                                       *
C*                    - "LASTTP" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ENTTMP" COMMON AREA.                        *
C*--------------------------------------------------------------------*
CEND
```

```
**********
* SRCHAR *
**********

C*-------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/16/86   *
C*             S. L. MADDERN                                          *
C*    VERSION:    1.0                            REVISED:            *
C*    ROUTINE NAME:   SRCHAR                                         *
C*    FUNCTION:                                                      *
C*      GIVEN A SPECIFIC KIND AND ARTIFICIAL ID OR KEY THIS ROUTINE  *
C*      WILL FIND THE CORRESPONDING ARTIFICIAL ID OR KEY.            *
C*    ENVIRONMENT:                                                   *
C*      IBM PASCAL LANGUAGE                                          *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W      *
C*    EXECUTION PROCEDURE                                            *
C*      CALLED BY    : WRTFIL                                        *
C*                   : ENRTV                                         *
C*      CALLS        : NONE                                          *
C*    DESCRIPTION OF ARGUMENTS                                       *
C*        KEY        I THE KEY OF THE TEMPORARY ENTITY               *
C*        ADB        I THE ADB OF THE TEMPORARY ENTITY               *
C*        DAT_REC I/O A RECORD CONTAINING  THE DATA TO BE INPUT AND  *
C*                      OUTPUT                                       *
C*        - SRFLAG   I A FLAG INDICATING WHETHER TO SEARCH FOR THE   *
C*                     KEY VALUE GIVEN THE ARTIFICIAL ID (1) OR THE  *
C*                     ARTIFICIAL ID GIVEN THE KEY VALUE (0).        *
C*        - KIND     I THE KIND OF ENTITY THAT THE ARTIFICIAL ID/KEY *
C*                     CORRESPOND TO.                                *
C*        - KEY    I/O THE KEY INDEXING THE SEARCH (SRFLAG = 0) OR   *
C*                     TO BE SEARCHED FOR (SRFLAG = 1)               *
C*        - ID     I/O THE ARTIFICIAL ID INDEXING THE SEARCH         *
C*                     (SRFLAG = 1) OR TO BE SEARCHED FOR (SRFLAG = 0) *
C*        RCC        0 RETURN CODE (0 = > SEARCH SUCCESSFUL)         *
C*                     RETURN CODE (1 = > SEARCH UNSUCCESSFUL        *
C*    PROCESSING DESCRIPTION:                                        *
C*        THIS ROUTINE NEEDS NO OPEN/CLOSE FILES                     *
C*    CHANGE CONTROL:                                                *
C*-------------------------------------------------------------------*
C*-------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                               *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF   *
C*                      THE ENTITIES BEING PROCESSED.                *
C*-------------------------------------------------------------------*
CEND
```

```
**********
* SORTAR *
**********

C*-------------------------------------------------------------------*
C*    AUTHOR:  J. M. PURSES                D360  CREATED:  12/16/86   *
C*             S. L. MADDERN                                         *
C*    VERSION:    1.0                            REVISED:            *
C*    ROUTINE NAME:   SORTAR                                         *
C*    FUNCTION.                                                      *
C*      SORTS THE ARTIFICIAL ID'S/KEYS COMMON AREA BY EITHER         *
C*      ARTIFICIAL ID OR KEY.                                        *
C*    ENVIRONMENT:                                                   *
C*      IBM PASCAL LANGUAGE                                          *
C*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.     *
C*    EXECUTION PROCEDURE:                                           *
C*      CALLED BY    : CRETMP                                        *
C*      CALLS        : ARTFID COMMON                                 *
C*                   : LASTAR COMMON                                 *
C*    DESCRIPTION OF ARGUMENTS:                                      *
C*        ISFLAG  I FLAG TO DETERMINE IF SORT IS BY ID OR KEY        *
C*                  0 = > SORT BY ID                                 *
C*                  1 = > SORT BY KEY                                *
C*        IRC     O RETURN CODE                                      *
C*    PROCESSING DESCRIPTION:                                        *
C*        THIS ROUTINE NEEDS NO OPEN/CLOSE FILES                     *
C*    CHANGE CONTROL:                                                *
C*-------------------------------------------------------------------*
C*-------------------------------------------------------------------*
C*    DATA STRUCTURES/MAJOR VARIABLES:                               *
C*        COMMON AREA - "ARTFID" HOLDS THE ARTIFICIAL ID'S/KEYS OF   *
C*                      THE ENTITIES BEING PROCESSED.                *
C*                    - "LASTAR" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ARTFID" COMMON AREA.                        *
C*                    - "ENTIMP" HOLDS THE KEYS OF THE TEMPORARY     *
C*                      ENTITIES AND THEIR ASSOCIATED W. F. ENTITY   *
C*                      KINDS.                                       *
C*                    - "LASTEIP" DEFINES THE LAST ARRAY INDEX OF THE *
C*                      "ENTIMP" COMMON AREA.                        *
C*-------------------------------------------------------------------*
CEND
```

Software References

Note the following chain of software references excludes calls to lower level Access Software and Pascal routines.

```
1..CSECT(FILRTV.FILRTV.E,272)     (* Main Interim DB Routine *)
 2..COMMON(FILRTV.ARTFID,40000)
 2..COMMON(FILRTV.LASTAR,4)
 2..COMMON(FILRTV.ENTTMP,4000)
 2..COMMON(FILRTV.LASTTP,4)

 2..CSECT(FILRTV.ENFILE,736)      (* Main File Routine *)
  3..CSECT(FILRTV.WRTFIL,1446)
   4..CSECT(FILRTV.SRCHAR,764)
   4..COMMON(FILRTV.ARTFID,40000)
   4..COMMON(FILRTV.LASTAR,4)
   4..CSECT(FILRTV.MALNO,512)
   4..CSECT(FILRTV.GETTMP,366)
    5..COMMON(FILRTV.ENTIMP,4000)
    5..COMMON(FILRTV.LASTAR,4)
   4..ENTRY(FILRTV.IHOECOMH.IBCOM )
   4..CSECT(FILRTV.MAEGKN,420)
   4..CSECT(FILRTV.MAEXEQ,1368)
   4..CSECT(FILRTV.MALGTK,576)
  3..COMMON(FILRTV.ARTFID,40000)
  3..COMMON(FILRTV.LASTAR,4)
  3..COMMON(FILRTV.ENTIMP,4000)
  3..COMMON(FILRTV.LASTIP,4)
  3..CSECT(FILRTV.MAED,1564)
  3..CSECT(FILRTV.MALD,688)
  3..CSECT(FILRTV.MALK,664)
  3..CSECT(FILRTV.CRETMP,40572)
   4..COMMON(FILRTV.ARTFID,40000)
   4..COMMON(FILRTV.LASTAR,4)
   4..COMMON(FILRTV.ENTIMP,4000)
   4..COMMON(FILRTV.LASTIP,4)
   4..CSECT(FILRTV.MAECR.576)
   4..CSECT(FILRTV.SORTAR,1468)
    5..COMMON(FILRTV.ARTFID,40000)
    5..COMMON(FILRTV.SORTAR,1468)
  3..ENTRY(FILRTV.IHOECOMH.IBCOM )
  3..CSECT(FILRTV.MAECTK,380)
  3..CSECT(FILRTV.MAEKND,448)
  3..CSECT(FILRTV.MALXEQ,816)
```

```
2..CSECT(FILRTV.ENRTV,81316)    (* Main Retrieve Routine *)
 3..CSECT(FILRTV.SRCHAR,764)
 3..COMMON(FILRTV.ARTFID,40000)
 3..COMMON(FILRTV.LASTAR,4)
 3..COMMON(FILRTV.ENTTMP,40000)
 3..COMMON(FILRTV.LASTTP,4)
 3..CSECT(FILRTV.MAECR,576)
 3..CSECT(FILRTV.CRETMP,40572)
  4..COMMON(FILRTV.ARTFID,40000)
  4..COMMON(FILRTV.LASTAR,4)
  4..COMMON(FILRTV.ENTTMP,40000)
  4..COMMON(FILRTV.LASTTP,4)
  4..CSECT(FILRTV.MAECR,576)
  4..CSECT(FILRTV.SORTAR,1468)
    5..COMMON(FILRTV.ARTFID,40000)
    5..COMMON(FILRTV.LASTAR,4)
 3..CSECT(FILRTV.GETTMP,366)
    5..COMMON(FILRTV.ENTTMP,4000)
    5..COMMON(FILRTV.LASTAR,4)
 3..ENTRY(FILRTV.IHOECOMH.IBCOM )
 3..CSECT(FILRTV.MAEXEQ,1368)
 3..CSECT(FILRTV.MAINIT,428)
 3..CSECT(FILRTV.MAKILL,366)
 3..CSECT(FILRTV.MALATC,836)
 3..CSECT(FILRTV.REDFIL,810)
  4..ENTRY(FILRTV.IHOECOMH.IBCOM )
```