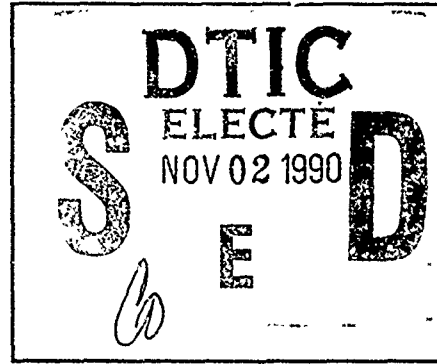


LOAN DOCUMENT

AD-A228 115	DTIC ACCESSION NUMBER	LEVEL	PHOTOGRAPH THIS SHEET	1	H A N D L E W I T H C A R E
			DTIC FILE COPY	INVENTORY	
		WRDC-TR-90-8027 Vol. II DOCUMENT IDENTIFICATION NOV 1990			
		DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited			
		DISTRIBUTION STATEMENT			
ACCESSION FOR NTIS ☐ GRA&I ☒ DTIC ☐ TRAC ☒ UNANNOUNCED ☐ JUSTIFICATION ☐ BY _____ DISTRIBUTION/ _____ AVAILABILITY CODES _____ DISTRIBUTION _____ AVAILABILITY AND/OR SPECIAL _____					
A-1 DISTRIBUTION STAMP				DATE ACCESSIONED	
				DATE RETURNED	
				REGISTERED OR CERTIFIED NUMBER	
DATE RECEIVED IN DTIC					
PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC					

AD-A228 115

WRDC-TR-90-8027
Volume II



GEOMETRIC MODELING APPLICATIONS INTERFACE PROGRAM

GMAP/PDDI SYSTEM COMPONENT PRODUCT SPECIFICATION (AS BUILT)

VOL. II - Schema Manager Listings

United Technologies Corporation
Pratt and Whitney
Government Products Division
P.O. Box 9600
West Palm Beach, Florida 33410-6900

November 1990

Final Report For Period August 1985 - March 1989

Approved for public release; distribution is unlimited

Manufacturing Technology Directorate
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

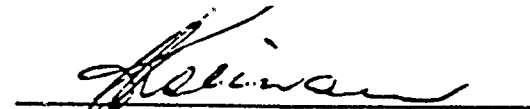
NOTICE

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

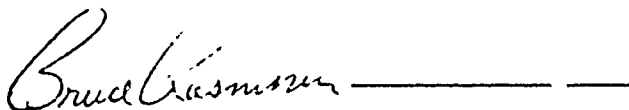
This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

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


Charles Gilman
Project Manager


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

FOR THE COMMANDER


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

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

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

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) FR 20889		5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8027, Vol. II	
6a. NAME OF PERFORMING ORGANIZATION United Technologies Corporation Pratt & Whitney Government Products Division	6b. OFFICE SYMBOL (If applicable) (P&W)	7a. NAME OF MONITORING ORGANIZATION Wright Research and Development Center Manufacturing Technology Directorate (WRDC/MTI)	
6c. ADDRESS (City, State and ZIP Code) P.O. Box 9600 West Palm Beach, Florida 33410-9600		7b. ADDRESS (City, State and ZIP Code) Wright-Patterson Air Force Base, OH 45433-6533	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-5122	
8c. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FUNDING NOS.	
11. TITLE (Include Security Classification) GEOMETRIC MODELING APPLICATIONS INTERFACE PROGRAM (GMAP)		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT NO.
		78011F	MTPI 06 84
12. PERSONAL AUTHOR(S) R. Disa, C. Van Wie, K. Arnold, J. Altemueller, A. Whelan, G. White, J. Purses			
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM 1 Aug 85 TO 31 MAR 89	14. DATE OF REPORT (Yr., Mo., Day) November 1990	15. PAGE COUNT 455
16. SUPPLEMENTARY NOTATION			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB. GR.	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)			
This "As-Built" Product Specification establishes the "as-built" Computer Program Configuration Item (CPCI) identified as the GMAP/PDDI System Components under U.S. Air Force Contract F33615-85-C-5122. It includes descriptions of the structure, functions, language, database requirements, interfaces, and quality assurance provisions of the primary GMAP system components: the System Translator, Model Access Software with Name/Value Interface, and the Schema Manager.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL David Judson		22b. TELEPHONE NUMBER (Include Area Code) (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

18. Subject Terms (Continued)

Product Life Cycle
Engineering
Manufacturing
Interface
Exchange Format
CAD
CAM
CIM
IBIS
RFC
System Translator
Schema Manager
Model Access Software
Name/Value Interface

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

FOREWORD

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

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

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

Volume II of this document provides the Schema Manager routine listings.

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

TABLE OF CONTENTS

Volume I

		<u>Page</u>
SECTION 1.	SCOPE.....	1-1
1.1	Identification.....	1-1
1.2	Functional Summary.....	1-1
1.3	Approach.....	1-2
SECTION 2.	REFERENCES.....	2-1
2.1	Reference Documents.....	2-1
2.1.1	Military.....	2-1
2.1.2	Commercial.....	2-4
2.1.3	Standards Organizations.....	2-5
2.2	Terms and Acronyms.....	2-6
2.2.1	Glossary A -- Terms Used In GMAP.....	2-6
2.2.2	Glossary B -- Terms Used In IDEF0 Diagrams....	2-19
2.2.3	Acronyms Used In GMAP.....	2-24
SECTION 3.	DETAIL DESIGN.....	3-1
3.1	System Overview.....	3-1
3.1.1	Physical Schemas.....	3-1
3.1.2	Software Packages.....	3-1
3.2	IDEF0 Function Models.....	3-1
3.2.1	Schema Manager.....	3-3
3.2.1.1	A-0 Manage Schema Data.....	3-3
3.2.1.2	A0 - Manage Schema Data.....	3-3
3.2.2	Model Access Software.....	3-7
3.2.2.1	A-0 Perform Model Access.....	3-7
3.2.2.2	A0 Perform Model Access.....	3-7
3.2.3	System Translator.....	3-10
3.2.3.1	A0 Exchange PDD.....	3-10
3.2.3.2	A1 Preprocess PDD.....	3-10
3.2.3.3	A12 Create Data Section.....	3-13
3.2.3.4	A2 Postprocess PDD.....	3-13
3.2.3.5	A23 Process Data Section.....	3-16
3.2.3.6	A24 Resolve Forward References.....	3-16
3.3	Application Interfaces.....	3-19
3.3.1	Interfaces Between GMAP/PDDI	
	System Components.....	3-19
3.3.1.1	Application Program/Working Form.....	3-20
3.3.1.2	User Interface/System Translator.....	3-20
3.3.1.3	Working Form/Exchange Format.....	3-22
3.3.1.4	Working Form/Database	
	Management System.....	3-22

TABLE OF CONTENTS (Continued)

	<u>Page</u>
3.3.1.5 Exchange Format.....	3-22
3.3.1.6 Schema Manager.....	3-23
3.3.2 Interfaces Between Applications.....	3-23
3.3.2.1 Working Form as Exchange Mechanism.....	3-24
3.3.2.2 Working Form as the Native Form.....	3-24
3.3.2.3 Exchange Format and Working Form.....	3-24
3.3.2.4 Transfer Using the Working Form, Exchange Format, and Direct Translator to the Exchange Format.....	3-28
3.4 Program Interrupts.....	3-29
3.4.1 Schema Manager.....	3-29
3.4.2 Model Access Software with Name Value Interface.....	3-31
3.5 Timing and Sequencing Description.....	3-33
3.6 Data Dictionary.....	3-34
3.6.1 Schema Manager.....	3-34
3.6.2 Model Access Software Data Dictionary.....	3-64
3.6.3 Name Value Interface Data Dictionary.....	3-92
3.6.4 System Translator Data Dictionary.....	3-104
3.7 Object Code Creation.....	3-107
3.7.1 Schema Manager.....	3-107
3.7.2 Model Access Software.....	3-107
3.7.3 System Translator.....	3-107
3.7.3.1 Use Portable Higher Order Language.....	3-107
3.7.3.2 Use Model Access Software.....	3-107
3.7.3.3 Interface to Exchange Format.....	3-108
3.7.3.4 Interface to Native System.....	3-108
3.8 Adaptation Data.....	3-108
3.9 Detail Design Description.....	3-108
3.9.1 Schema Manager Hierarchy.....	3-109
3.9.2 Model Access Software Hierarchy.....	3-120
3.9.3 Name/Value Interface Hierarchy.....	3-162
3.9.4 System Translator.....	3-168
3.9.4.1 Postprocessor.....	3-168
3.9.4.2 Preprocessor.....	3-170

TABLE OF CONTENTS (Continued)

		<u>Page</u>
Volume II		
3.10	Routine Listings.....	3-171
3.10.1	Schema Manager.....	3-171
3.10.1.1	Index.....	3-171
3.10.1.2	Listings.....	3-178
Volume III		
3.10.2	Model Access Software.....	3-625
3.10.2.1	Index.....	3-625
3.10.2.2	Listings.....	3-631
3.10.3	N/VI.....	3-913
3.10.3.1	Index.....	3-913
3.10.3.2	Listings.....	3-914
Volume IV		
3.10.4	System Translator.....	3-979
3.10.4.1	Index.....	3-979
3.10.4.2	Listings.....	3-980
SECTION 4	QUALITY ASSURANCE.....	4-1
4.1	Quality Assurance (QA) Requirements.....	4-1

3.10 Routine Listings

3.10.1 Schema Manager

This Section presents the software routines that comprise the GMAP/PDDI system. The routine for each primary system component, listed below, are presented in alphabetical order:

- o 3.10.1 -- Schema Manager (Volume II)
- o 3.10.2 -- MAS (Volume III)
- o 3.10.3 -- N/VI (Volume III)
- o 3.10.4 -- System Translator (Volume IV).

3.10.1.1 Index

ADDENUM	- Displays the ADD ENUMERATION menu.
ADDFIELD	- Displays the ADDFIELD menu.
APPROVE	- Displays the APPROVE UPDATE menu.
BATDVR	- Mainline driver for the Batch Interface.
BATERR	- Error reporting and recovery for the Batch Interface.
BATRPT	- Batch Interface routine to invoke report generation routines
BCSMAN	- Main driver for the Conceptual Schema Report.
BLDARRAY	- Builds the ARRAY definition.
BLDBPDEF	- Builds the BACKPATCH list for unresolved references.
BLDCLASS	- Builds the CLASS definition.
BLDCLS	- Batch Interface routine that creates a CLASS definition.
BLDDFTYP	- Builds the DEFINED TYPE definition.
BLDEITEM	- Builds the ENUMERITEM definition.
BLDENT	- Builds the ENTITY definition.
BLDENUMR	- Builds the ENUMERATION definition.
BLDFIELD	- Builds the ATTRIBUTE definition.
BLDGBLFD	- Builds the GLOBAL ATTRIBUTE definition.
BLDINT	- Builds the INTEGER definition.
BLDLOG	- Builds the LOGICAL definition.
BLDPNTR	- Builds the POINTER definition.
BLDREAL	- Builds the REAL definition.
BLDSSCMA	- Builds the SUBSCHEMA definition.
BLDSTRNG	- Builds the STRING definition.
BLDSTRUC	- Builds the STRUCTURE definition.
BLDSUB	- Creates a SUBSCHEMA definition.
BLDSUPER	- Create a SUPERTYPE definition.
BLDUNRES	- Stores unresolved references.
BLEXICAL	- Obtains the longest lexeme, excluding comments.
BSCINCLD	- Generates the PASCAL Include File.

BSCTRSPR - Process the entries on the transaction stack.
CLRSTK - Clears the transaction stack.
CRARRAY - Displays the CREATE ARRAY menu.
CRCLASS1 - Displays the first CREATE CLASS menu.
CRCLASS2 - Displays the second CREATE CLASS menu.
CRDEFTYP - Displays the CREATE DEFINED TYPE menu.
CRENTITY - Displays the CREATE ENTITY menu.
CRENUM - Displays the CREATE ENUMERATION menu.
CRFIELD - Displays the CREATE ATTRIBUTE menu.
CRINTGR - Displays the CREATE INTEGER menu.
CRLIST - Display the CREATE LIST menu.
CRPNTR - Displays the CREATE POINTER menu.
CRREAL - Displays the CREATE REAL menu.
CRSET - Displays the CREATE SET menu.
CRSTRING - Displays the CREATE STRING menu.
CRSUBSCM - Displays the CREATE SUBSCHEMA menu.
CRSUPTYP - Displays the CREATE/REFERENCE SUPERTYPE menu.
CRURUL - Creates user delete rules for MAS.
CSARYWRT - Writes out an ARRAY definition.
CSCLSHDG - Writes out a CLASS heading on a new page.
CSCLSWRT - Writes out CLASS definitions to a file.
CSDEFHDG - Writes out a DEFINED TYPE heading on a new page.
CSDEFWRT - Writes out a DEFINED TYPE name.
CSENHWRT - Writes out an ENUMERATION definition.
CSENTHDG - Writes out an ENTITY heading on a new page.
CSENTWRT - Writes out ENTITY definitions to a file.
CSGBLHDG - Writes out a GLOBAL ATTRIBUTE heading on a new page.
CSGBLWRT - Writes out GLOBAL ATTRIBUTE definitions to a file.
CSHDGWRT - Calls the appropriate heading routine.
CSINDWRT - Writes an index for ENTITIES, CLASSES, or SUBSCHEMAS.
CSINTWRT - Writes out an INTEGER definition.
CSLOGWRT - Writes out a LOGICAL definition.
CSMAIN - Main driver for the Conceptual Schema Report.
CSNEWPG - Creates a new page in the Conceptual Schema Report.
CSPTRWRT - Writes out a POINTER definition.
CSRELWRT - Writes out a REAL definition.
CSRMAIN - Generate, browse or print the Conceptual Schema Report.
CSRMENU - Display the Conceptual Schema Report menu.
CSRPTCVR - Prints out the cover page for the Conceptual Schema Report.
CSSTGWRT - Writes out a STRING definition.
CSSTRWRT - Writes out a STRUCTURE definition.
CSSUBHDG - Writes out a SUBSCHEMA heading on a new page.
CSSUBWRT - Writes out SUBSCHEMA definitions to a file.
CSSUPHDG - Write a SUPERTYPE heading on a new page.
CSSUPWRT - Write out SUPERTYPE definitions.
CSTYPWRT - Writes out DEFINED TYPE definitions to a file.
DDABNDS - Writes out the low bound and high bound for a ARRAY attribute.

DDADB - Writes out the ADB portion of an entity.
DDARRAY - Writes out the ARRAY attribute of an entity.
DDCL - Writes out the CONSTITUENT LIST portion of an entity.
DDCLASS - Writes the class KINDS to a file.
DDENTITY - Writes the entity KINDS to a file.
DDENUM - Writes out the ENUMERATION attribute of an entity.
DDREPORT - Writes the DATA DICTIONARY (character form) of the definitions.
DDSTRUC - Writes out STRUCTURE definitions.
DDWRITE - Writes out ENTITY definitions.
DEFADD - Add a forward reference to a list of unresolved references.
DEFARR - Process an ARRAY definition.
DEFATT - Process an ATTRIBUTE definition.
DEFBAS - Process a primitive data type definition.
DEFCLS - Process a CLASS definition.
DEFDEF - Process a DEFINED TYPE reference.
DEFENM - Process an ENUMERATION definition.
DEFENT - Process an ENTITY definition.
DEFGBL - Process a GLOBAL ATTRIBUTE definition.
DEFPRE - Process a numeric precision or string length.
DEFPTR - Process a POINTER definition.
DEFQUERY - Determines if a new entity satisfies an unresolved reference.
DEFSTC - Process a STRUCTURE definition.
DEFSUB - Process a SUBSCHEMA definition.
DEFSUP - Process a SUPERTYPE definition.
DEFTYP - Process a DEFINED TYPE definition.
DISPLIST - Displays a list of entities.
ENTCLS - Determines if a name has been used for an entity or class.
ERRMSG - Formats a Batch Interface error message.
ERRREC - Recover from a Batch Interface error.
GETDD - Reads the DATA DICTIONARY (character form) of the definitions.
LEXICAL - Obtains the longest lexeme, including comments.
LMEM23 - Displays the LIST MEMBERS menu.
MCREATE - Displays the CREATE menu.
MFILMOD - Displays the FILE/RETRIEVE menu.
MINCLUD - Displays the list of SUBSCHEMAS.
MMAIN - Displays the MAIN menu.
MNEWMOD - Displays the menu to confirm creation of a new model.
MQBHALL - Prints all entities in the model.
MQBHATT - Prints individual instances of a specified entity kind.
MQBHATTS - Prints all instances of a specified entity kind.
MQBHENT - Prompt for all or selected instances of an entity kind.
MQBHMAIN - Prompt for all or selected entity kinds.
MQCLMU - Displays a list of constituents.
MQGETVAL - Converts an attribute value to character form.
MQGTDEFN - Retrieves the entity definition from the Data Dictionary.
MQIAATT - Displays individual instances of a specified entity kind.
MQIAENT - Displays interactive entity menu.

MQIAMAIN - Displays main interactive menu.
MQNCLMU - Displays a menu when no constituents found.
MQNUSRMU - Displays a menu when no users found.
MQUDVR - Mainline driver for the Model Query Utility.
MQUSRMU - Displays the list of users for a specific entity.
MREPORT - Displays the menu for REPORT selection.
MREVIEW - Displays the REVIEW menu.
MUPDATE - Displays the UPDATE menu.
NVRTVRS - Retrieves entity definitions from the file.
PHALFLD - Creates physical definition for an ATTRIBUTE.
PHALST - Assign physical location for a STRUCTURE attribute.
PHBNST - Group attributes by boundary alignment requirements.
PHBYFPOS - Assign physical locations for attributes with a position override specified.
PHDECBYT - Determine byte precision from decimal precision and group by boundary alignment requirements.
PHENTITY - Assign physical locations to the attributes of an entity.
PHGLOBAL - Assign physical locations for GLOBAL ATTRIBUTES.
PHGTFLD - Get the ATTRIBUTE for which a physical definition is to be created.
PHGTST - Determines the physical definition of a STRUCTURE based on its component ATTRIBUTES.
PHPOSITN - Determines the attribute position order.
PHSRTFLD - Determines the location of attributes according to boundary alignment requirements.
PHSRTORD - Sort attributes by position override.
PHSUBTYP - Determines the physical location for supertype attributes.
PHWOFPOS - Determines the location of attributes without position override.
PHYSICAL - Main driver for creating physical definitions.
PSGTSM - Build the run-time subschema for the Physical Subschema Report.
PSMASKND - Insert the attributes required by MAS.
PSORDER - Determine physical schema order of attributes within entity.
PSRABNDS - Writes out the low bound and high bound for a ARRAY attribute.
PSRADB - Writes out the ADB portion of an entity.
PSRARRAY - Writes out the ARRAY attribute of an entity.
PSRCL - Writes out the CONSTITUENT LIST portion of an entity.
PSRENUM - Writes out the ENUMERATION attribute of an entity.
PSREPORT - Writes the PHYSICAL SUBSCHEMA REPORT.
PSRHEAD - Writes the heading for the PHYSICAL SUBSCHEMA REPORT.
PSRINDEX - Writes the table of contents for the PHYSICAL SUBSCHEMA REPORT.
PSRNAME - Identify an attribute as global, inherited, or local.
PSRSTC - Write a STRUCTURE attribute to the Physical Subschema Report.
PSTRGF - Insert the GLOBAL ATTRIBUTES into the run-time subschema for the Physical Subschema Report.
PSTRSM - Insert an entity into the run-time subschema for the Physical Subschema Report.

PSTRST - Insert a SUPERTYPE into the run-time subschema for the Physical Subschema Report.

PSTRSTC - Insert a STRUCTURE attribute into the run-time subschema for the Physical Subschema Report.

REARRAY - Displays the REVIEW ARRAY menu.

RECLASS - Displays the REVIEW CLASS menu.

REDEFTYP - Displays the REVIEW DEFINED TYPE menu.

REENTITY - Displays the REVIEW ENTITY menu.

REENUM - Displays the REVIEW ENUMERATION menu.

REFIELD - Displays the REVIEW ATTRIBUTE menu.

REFIELD1 - Displays the REVIEW ATTRIBUTE menu.

REFIELD2 - Displays the REVIEW ATTRIBUTE menu.

REFSUP - Batch interface routine to resolve a reference to a SUPERTYPE.

REINTGR - Displays the REVIEW INTEGER menu.

RELIST - Displays the REVIEW LIST menu.

REPNTN - Displays the REVIEW POINTER menu.

REREAL - Displays the REVIEW REAL menu.

RESET - Displays the REVIEW SET menu.

RESTRING - Displays the REVIEW STRING menu.

RESTRUC - Displays the REVIEW STRUCTURE menu.

RESUBSCM - Displays the REVIEW SUBSCHEMA menu.

RESUPTYP - Displays the REVIEW SUPERTYPE MENU.

RSCPAI - Copies the array index table into the run-time subschema data structure.

RSCPAT - Copies the array size and bounds into the run-time subschema data structure.

RSCPCI - Copies the pointer index table into the run-time subschema data structure.

RSCPCT - Copies the pointer kinds into the run-time subschema data structure.

RSCPEI - Copies the enumeration index table into the run-time subschema data structure.

RSCPET - Copies the enumeration values into the run-time subschema data structure.

RSFILE - Writes the run-time subschema data structure to a file.

RSGTSM - Builds the run-time data structure from the schema model.

RSMASKND - Inserts the MAS required attributes into the run-time subschema data structure.

RSORDER - Determines the physical schema order of the attributes.

RSTRGF - Enters GLOBAL ATTRIBUTES into the run-time subschema data structure.

RSTRSM - Enters the ATTRIBUTES for an entity into the run-time subschema data structure.

RSTRST - Inserts a SUPERTYPE into the run-time subschema data structure.

RSTRSTC - Inserts a STRUCTURE attribute into the run-time subschema data structure.

RS1100 - Inserts an ARRAY_ENTITY pseudo entity into the run-time subschema data structure.

SCALFSRT - The ordering function used with MALSRT.

SCARYCR - Creates an ARRAY definition.

SCARYUP - Updates an ARRAY definition.

SCBASIN - Writes the ENTITY KIND constants for the PASCAL Include File.

SCCHRCK - Validates an entity name.

SCCLSCR1 - Gathers the name and kind for a CLASS definition.

SCCLSCR2 - Gathers the members of a CLASS definition.

SCCLSUP - Updates a CLASS definition.

SCCOMPAR - Compares two names for uniqueness.

SCCONIN - Writes the ENTITY KIND constants for the PASCAL Include File.

SCCREATE - Determines the next menu for the selected CREATE option.

SCDEFER - Creates a DEFINED TYPE definition.

SCDEFUP - Updates a DEFINED TYPE definition.

SCENUMUP - Updates an ENUMERATION definition.

SCENTCR - Creates an ENTITY definition.

SCENTIN - Writes the ENTITY definitions for the PASCAL Include File.

SCENTUP - Updates an ENTITY definition.

SCENUCR - Creates an ENUMERITEM definition.

SCEXIST - Verifies a reference to an ENTITY.

SCFLDAD - Adds an ATTRIBUTE to an entity.

SCFLDCR - Creates an ATTRIBUTE definition.

SCFLDIN - Writes an ATTRIBUTE definition to the PASCAL Include File.

SCFLDLST - Creates a list of attributes for an entity, supertype, or structure.

SCFLDSRT - Locates the MAS key for a Schema Model element.

SCFLDST - Sorts the ATTRIBUTES of an entity into physical schema order.

SCFLDUP - Updates an ATTRIBUTE definition.

SCFNDKEY - Locates the MAS key for a Schema Model element.

SCGENRPT - Determines the SUBSCHEMA for a report and call the generation routine.

SCHDVR - Mainline driver for the Schema Manager Interactive Interface.

SCINCLD - Driver to generate the PASCAL Include File.

SCINTCR - Creates an INTEGER definition.

SCINTUP - Updates an INTEGER definition.

SCKEFIND - Locates the MAS key for a definition to be updated or reviewed.

SCKEYIN - Writes the KEYBLOCK declaration for the PASCAL Include File.

SCLISTCR - Creates a LIST definition.

SCLISTUP - Updates a LIST definition.

SCMASIN - Writes the ENTBLOCK declaration for the PASCAL Include File.

SCMEMAD - Displays a list of members, return the key of the one selected.

SCMEMLST - Lists the members/constituents of an entity, class, subschema, pointer, structure, global attribute or enumeration.

SCPOPKE - Pop an entry from the stack of keys.

SCPOPTR - Pop an entry from the stack of transactions.

SCPRMFL - Builds an array of entity names.

SCPRMRE - Builds a list of definitions for REVIEW.
SCPTRCR - Creates a POINTER definition.
SCPTRUP - Updates a POINTER definition.
SCPUSHKE - Pushes an entry on the stack of keys.
SCPUSHTR - Pushes an entry on the stack of transactions.
SCRELCR - Creates a REAL definition.
SCRELUP - Updates a REAL definition.
SCREVIEW - Determines the next menu for the selected REVIEW option.
SCRPTM - Determines the option for a report and call the generation routine.

SCSETCR - Creates a SET definition.
SCSETUP - Updates a SET definition.
SCSRTPS - Routine called by MALSRT to order attributes by position override position.

SCSTCUP - Updates a STRUCTURE definition.
SCSTGCR - Creates a STRING definition.
SCSTGUP - Updates a STRING definition.
SCSUBCR - Creates a SUBSCHEMA definition.
SCSUBUP - Updates a SUBSCHEMA definition.
SCSUPCR - Creates a SUPERTYPE definition.
SCSUPUP - Updates a SUPERTYPE definition.
SCTRSPR - Process a transaction stack entry.
SCTYPIN - Writes DEFINED TYPE definitions for the PASCAL Include File.
SCTYUP - Updates the DEFINED TYPE, ARRAY, or ATTRIBUTE data type.
SCUNIQUE - Verifies uniqueness of names and kind numbers within the Schema model.

SCUNQEST - Calls MAKXEQ to execute SCUNQEST.
SCUNQPN - Checks the transaction stack for names and kind numbers.
SCUPDATE - Determines the next menu for the selected UPDATE option.
SORTKIND - The ordering function called by MALSRT for kind numbers.
SORTNAME - The ordering function called by MALSRT for names.
UPARRAY - Displays the UPDATE ARRAY menu.
UPCLASS1 - Displays the first UPDATE CLASS menu.
UPCLASS2 - Displays the second UPDATE CLASS menu.
UPDEFTYP - Displays the UPDATE DEFINED TYPE menu.
UPENTY1 - Displays the first UPDATE ENTITY menu.
UPENTY2 - Displays the second UPDATE ENTITY menu.
UPENUM - Displays the UPDATE ENUMERATION menu.
UPFIELD - Displays the UPDATE ATTRIBUTE menu.
UPINT - Displays the UPDATE INTEGER menu.
UPLIST - Displays the UPDATE LIST menu.
UPPNTR - Displays the UPDATE POINTER menu.
UPREAL - Displays the UPDATE REAL menu.
UPSET - Displays the UPDATE SET menu.
UPSTRING - Displays the UPDATE STRING menu.
UPSTRUC - Displays the UPDATE STRUCTURE menu.
UPSUB1 - Displays the first UPDATE SUBSCHEMA menu.

UPSUB2 - Displays the second UPDATE SUBSCHEMA menu.
UPSUPER - Displays the UPDATE SUPERTYPE menu.
XATTDATA - Cross-reference by data type.
XATTNAME - Cross-reference by name.
XATTRES - Displays cross-reference results.
XEXPREC - Generates a list of numeric precisions and string lengths.
XFNDARY - Generates a list of ARRAY data types (including LIST, SET).
XFNDKEY - Determines the MAS key for an name or kind number.
XFNDNAME - Generates a list of ATTRIBUTES with a given name.
XFNDPREC - Generates a list of numeric data type of a given precision or strings of a given length.
XLISTENT - Generates a list of aggregations in which the entity is a member.
XMAIN - Cross-reference driver routine.
XMATTRES - Displays a cross-reference menu.
XMMAIN - Displays a cross-reference menu.
XMNAMSPE - Displays a cross-reference menu.
XMPRESPE - Displays a cross-reference menu.
XMRESULT - Displays a cross-reference menu.
XMTYPSPE - Displays a cross-reference menu.
XNAMENUM - Determines whether a string contains a name or a number.
XRESULT - Displays the cross-reference report results.

3.10.1.2 Listings

The listings begin on the following page.

```
(* %INCLUDE ADDENUM *)
(**)
PROCEDURE ADDENUM(VAR MESS      : MESSAGE;
                  VAR NAME      : IDCHAR;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE ADD ENUMERATION MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      O   THE NAME OF THE ENUMERATION FROM PANEL
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE ENUMERATION PANEL (ADDENUM) BY MAKING
(*   ISPLNK CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE.  THIS DATA AS WELL AS OTHER INFORMATION
(*   GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING
(*   PROCEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE ADDFIELD *)
(**)
PROCEDURE ADDFIELD(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR POS       : CHAR8;
                   (* VAR PURP    : CHAR8; *)
                   VAR REQD      : CHAR8;
                   (* VAR DEPD    : CHAR8; *)
                   VAR COM       : CHAR50;
                   VAR FTYPE     : ENTITY_TYPE;
                   VAR FLDTYPE   : T_FIELDTYPE;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE ADDFIELD PANEL
(*
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME       O   THE NAME OF THE FIELD
(*   POS        O   THE PHYSICAL POSITION NUMBER IN THE ADB
(*   PURP       O   THE PURPOSE OF THE FIELD
(*   REQD       O   REQUIREDNESS OF THE FIELD
(*   DEPD       O   DEPENDENCE OF THE FIELD
(*   FTYPE      O   THE CONSTITUENT ENTITY TYPE
(*   FLDTYPE    O   THE USER ENTITY TYPE
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
```

```
(* $EXECUTION PROCEDURE: *)
(* SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* DISPLAY THE ADD FIELD PANEL (ADDFIELD) BY MAKING ISPLNK *)
(* CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(* TYPE. *)
(* *)
(* $COMMENTS: *)
(* NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* REVISED: JANUARY 1988 C. H. MOHME DBMA *)
(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE. *)
(* *)
(* REVISED: 09/28/87 C. H. MOHME DBMA *)
(* CHANGED TO INCORPORATE THE SUPERTYPE FIELDS. *)
(* *)
(* REVISED: 08/13/87 C. H. MOHME DBMA *)
(* CHANGED PANEL OPTION NUMBERS; CHANGED FIELD ADB DATA; ADDED *)
(* LIST AND SET. NOTE: THE LIST AND SET DATA TYPES ARE IMPL- *)
(* MENTED IN THE SOFTWARE AS AN ARRAY. A FIELD WAS ADDED TO THE *)
(* ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL ARRAY, *)
(* LIST, OR SET. *)
(* *)
(* REVISED: 07/02/87 C. H. MOHME DBMA *)
(* CHANGED CURSOR POSITIONING. *)
(* *)
(* ORIGINATED: 07/30/86 C. H. MOHME DBMA *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE ADDFIELD *)
```


(* %INCLUDE APPROVE *)
(**)

PROCEDURE APPROVE(VAR MESS : MESSAGE;
VAR NAME : T_NAME;
VAR NEXT_OP : OPERATIONS;
VAR RR : RET_REC);

SUBPROGRAM;

(**)
(*-----*)
(* *)
(* \$FUNCTION: *)
(* THIS PROCEDURE : *)
(* DISPLAYS THE APPROVE UPDATE PANEL *)
(* *)
(* \$DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* MESS I THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(* NAME I THE NAME OF THE ENTITY *)
(* NEXT_OP 0 ENUMERATED TYPE INDICATING THE NEXT *)
(* OPERATION *)
(* RR 0 INDICATES IF AN ERROR HAS OCCURRED AND, *)
(* IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(* *)
(* \$COMMONS: *)
(* NONE *)
(* *)
(* \$ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* DDNAMES USED WITH STANDARD FILES: *)
(* NONE *)
(* *)
(* \$EXECUTION PROCEDURE: *)
(* SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* \$PROCESSING DESCRIPTION: *)
(* DISPLAY THE APPROVE UPDATE PANEL (APPROVE) BY MAKING ISPLNK *)
(* CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(* TYPE. *)
(* *)
(* \$COMMENTS: *)
(* NONE *)
(* *)
(* \$CHANGE CONTROL: *)

```
(* %INCLUDE BATDVR *)
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)     THIS IS THE MAINLINE PROGRAM WHICH DRIVES THE BATCH
(*)     INTERFACE OF THE SCHEMA MANAGER.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NONE
(*)
(*) $COMMONS:
(*)     NONE
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)     DDNAMES USED WITH STANDARD FILES:
(*)         SOURCE = INPUT FILE OF SCHEMA DEFINITIONS IN THE
(*)             EXPRESS INFORMATION MODELING LANGUAGE
(*)
(*)         REPORT1 = OUTPUT FILE OF ACTIONS TAKEN DURING THE
(*)             BATCH PROCESSING.
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
```

```
(* BEGIN %INCLUDE BATERR *****)
(*)
PROCEDURE BATERR ( Const Expected      : T_Expected;
                   Var  Ent_Kind       : Integer;
                   Var  Token          : T-Token;
                   Var  Token_Value    : T-Token_Value;
                   Var  Token_Location : Integer;
                   Var  Token_Length   : Integer;
                   Var  Report1        : Text );

    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   Error reporting and recovery for Batch Interface.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   Ent_Kind       I    Kind of entity being constructed
(*)   Expected       I    Record containing:
(*)                       Entries:   Number of entries
(*)                               in array
(*)                       Token_Value: Array of expected
(*)                               token values
(*)   Report1        O    Error messages
(*)   Token          I/O   Input:  unrecognized token
(*)                       Output: first recognized token
(*)   Token_Length   I/O   Input:  length of unrecognized token
(*)                       Output: length of recognized token
(*)   Token_Location I/O   Input:  location of unrecognized
(*)                       token
(*)                       Output: location of recognized token
(*)   Token_Value    I/O   Input:  value of unrecognized token
(*)                       Output: value of recognized token
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL/VS SEGMENT
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   Internal routine for Batch Interface of the Schema Manager
(*)
(*) $PROCESSING DESCRIPTION:
(*)   Highlight the unrecognized token.
(*)   List the expected tokens.
```

```
(*      Highlight the tokens ignored while searching for a      *)
(*      recognized token.                                         *)
(*                                                                *)
(* $COMMENTS:                                                     *)
(*                                                                *)
(* $CHANGE CONTROL:                                              *)
(*      ORIGINATED: 7 December 1987, G. A. White                *)
(*      REVISED:   <date, responsible person, reason/description> *)
(*                                                                *)
(* END %INCLUDE BATERR *****)
```

```
(* %INCLUDE BATRPT *)
(**)
PROCEDURE BATRPT(VAR IRC          : RET_REC;
                 VAR TOKEN        : T_TOKEN;
                 VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                 VAR REPORT1       : TEXT);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   BATCH INTERFACE ROUTINE THAT INVOKES THE NECESSARY ROUTINES
(*   TO GENERATE THE SPECIFIED REPORT.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           O    INTERNAL RETURN CODE
(*   TOKEN         I/O  TOKEN FROM BATCH INPUT
(*   TOKEN_VALUE   I/O  TOKEN VALUE FROM BATCH INPUT
(*   TEXT          I/O  OUTPUT FILE
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*   PERFORM INITIALIZATIONS
(*   GET THE REPORT TYPE
(*   IF THE REPORT TYPE IS NOT CONCEPTUAL SCHEMA, THEN GET THE
(*   SUBSCHEMA NAME.
(*   GET THE SUBSCHEMA KEY AS APPROPRIATE AND GENERATE A REPORT
(*   GET THE SUBSCHEMA KEY
(*   PHYSICALIZE THE SUBSCHEMA, IF NECESSARY
(*   IF REPORT TYPE IS PASCAL INCLUDES, THEN GENERATE THE PASCAL
(*   INCLUDE FILES.
(*   IF REPORT TYPE IS DATA DICTIONAY, THEN GENERATE THE DATA
(*   DICTIONARY.
(*   IF REPORT TYPE IS PHYSICAL SUBSCHEMA, THEN GENERATE
(*   THE PHYSICAL SUBSCHEMA REPORT.
(*   PRINT ERROR MESSAGES AS APPROPRIATE
(*)
```

```
(* IF REPORT TYPE IS CONCEPTUAL SCHEMA, THEN GENERATE THE *)
(* CONCEPTUAL SCHEMA REPORT *)

(* PRINT ERROR MESSAGES AS APPROPRIATE *)
(* GET THE NEXT TOKEN *)
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* ORIGINATED: 06/09/87 C. H. MOHME DBMA
(*
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE BATRPT *)
```

```

(*) %INCLUDE BCSMAIN *)
(**)
PROCEDURE BCSMAIN(VAR IRC      : RET_REC);

    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)     THIS ROUTINE SERVES AS THE MAIN DRIVER FOR THE CONCEPTUAL
(*)     SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME      I/O  DESCRIPTION
(*)     ====      ==  =====
(*)     IRC        0   INTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)     NONE
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*)     INITIALIZE THE VARIABLES USED WITHIN THIS ROUTINE.
(*)     WRITE OUT TO THE FILE THE REPORT COVER.
(*)     INITIALIZE THE PAGE NUMBER AND PAGE CHAIN.
(*)     MAKE A LIST OF DEFINED TYPES WITHIN THE SCHEMA.
(*)     ALPHABETIZE THE LIST OF DEFINED TYPES.
(*)     WRITE OUT TO THE FILE THE DEFINED TYPES WITHIN THE SCHEMA.
(*)     DELETE THE LIST OF DEFINED TYPES WITHIN THE SCHEMA.
(*)     MAKE A LIST OF GLOBAL FIELDS WITHIN THE SCHEMA.
(*)     WRITE OUT TO THE FILE THE GLOBAL FIELDS WITHIN THE SCHEMA.
(*)     DELETE THE LIST OF GLOBAL FIELDS WITHIN THE SCHEMA.
(*)     MAKE A LIST OF ENTITIES WITHIN THE SCHEMA.
(*)     ALPHABETIZE THE LIST OF ENTITIES.
(*)     WRITE OUT TO THE FILE THE ENTITIES WITHIN THE SCHEMA.
(*)     MAKE A LIST OF CLASSES WITHIN THE SCHEMA.
(*)     ALPHABETIZE THE LIST OF CLASSES.
(*)     WRITE OUT TO THE FILE THE CLASSES WITHIN THE SCHEMA.
(*)     MAKE A LIST OF SUBSCHEMAS WITHIN THE SCHEMA.
(*)     ALPHABETIZE THE LIST OF SUBSCHEMAS.

```

```
(*  WRITE OUT TO THE FILE THE SUBSCHEMAS WITHIN THE SCHEMA.      *)
(*  WRITE OUT TO THE FILE THE ENTITY INDEX.                      *)
(*  DELETE THE LIST OF ENTITIES.                                  *)
(*  WRITE OUT TO THE FILE THE CLASS INDEX.                        *)
(*  DELETE THE LIST OF CLASSES.                                   *)
(*  WRITE OUT TO THE FILE THE SUBSCHEMA INDEX.                   *)
(*  DELETE THE LIST OF SUBSCHEMAS.                               *)
(*  IF NO MODEL EXISTS, WRITE APPROPRIATE MESSAGE.               *)
(*  DISPOSE OF POINTERS USED IN THIS ROUTINE.                    *)
(*                                                                  *)
(*  $COMMENTS:                                                    *)
(*                                                                  *)
(*      WITHIN THE INCLUDE FILE 'CSTYPCON', ONE CAN SET THE MAXIMUM *)
(*      NUMBER OF LINES PER PAGE IN THE REPORT.                  *)
(*                                                                  *)
(*  $CHANGE CONTROL:                                              *)
(*                                                                  *)
```



```
(* %INCLUDE BLDARRAY *)
(**)
PROCEDURE BLDARRAY(VAR IRC : RET_REC;
                   CONST ARRAY_DATA : TRANSACTION;
                   CONST CNST_KEY : ENTIKEY;
                   VAR ARRAY_KEY : ENTIKEY);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE ARRAY ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   ARRAY_DATA    I    TRANSACTION DATA OF THE ARRAY ENTITY
(*   CNST_KEY      I    KEY TO THE CONSTITUENT
(*   ARRAY_KEY     0    KEY TO THE ARRAY ENTITY
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   A LIST IS MADE OF ALL THE ARRAY ENTITIES. IF THE LIST IS
(*   NOT EMPTY THEN IT IS SEARCHED FOR A MATCH. IF THE BOUNDS
(*   ARE IDENTICAL THE CONSTITUENT KEYS ARE COMPARED. IF THESE
(*   MATCH THE KEY TO THE ARRAY ENTITY THAT IS IDENTICAL TO THE
(*   ONE TO BE CREATED IS PASSED BACK TO THE CALLING PROCEDURE.
(*   THE LIST IS SEARCHED UNTIL A MATCH IS MADE OR THE END OF
(*   THE LIST IS ENCOUNTERED. IF NO MATCH IS FOUND THE ARRAY
(*   ENTITY IS CREATED USING MAS.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE BLDBPDEF *)
(**)
  PROCEDURE BLDBPDEF(VAR IRC          : RET_REC;
                     VAR IDENTIFIER    : T_NAME;
                     VAR KIND          : INTEGER;
                     VAR REFERENCE_KEY : ENTKEY;
                     VAR DEFINITION_KEY : ENTKEY);

    SUBPROGRAM;

(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE BUILDS THE BACKPATCH ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   IRC           0    INTERNAL RETURN CODE
(*)   IDENTIFIER    I    ENTITY NAME
(*)   KIND          I    ENTITY KIND NUMBER
(*)   REFERENCE_KEY I    THE KEY TO THE ENTITY THAT REFERENCES
(*)                   THE UNRESOLVED ENTITY
(*)   DEFINITION_KEY 0    THE KEY TO THE CREATED BACKPATCH
(*)                   ENTITY
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THE BACKPATCH ENTITY IS CREATED USING MAS ROUTINES
(*)
(*) $COMMENTS:
(*)   NONE
(*)
(*) $CHANGE CONTROL:
(*)

```

```

(** %INCLUDE BLDCLASS *)
(**)
  PROCEDURE BLDCLASS(VAR IRC : RET_REC;
                     CONST CLASS_DATA : TRANSACTION;
                     VAR CLASS_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE CLASS ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   CLASS_DATA    I    TRANSACTION DATA OF THE CLASS ENTITY
(*   CLASS_KEY     0    KEY TO THE CLASS ENTITY
(*
(* $COMMONS:
(*   REF
(*   CURRENT_LIST  I/O  LIST CURRENTLY IN USE CONTAINING THE
(*                       CONSTITUENTS OF THE CLASS ENTITY
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(*   ROUTINE MAECR IS CALLED TO CREATE THE CLASS ENTITY.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*

```

```
(* %INCLUDE BLDCLS *)
(**)
  PROCEDURE BLDCLS(VAR IRC          : RET_REC;
                   VAR TRANS_STACK  : TRANSPTR;
                   VAR CLASS_FLAG   : BOOLEAN;
                   VAR CLS_ENT_HEAD : ENTITY_LIST_PTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    BATCH INTERFACE ROUTINE THAT CREATES A CLASS ENTITY IN THE
(*    SCHEMA MODEL FROM A CLASS NAME, KIND NUMBER, AND LIST OF
(*    CONSTITUENTS.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME          I/O  DESCRIPTION
(*    ====          ==  =====
(*    IRC           0    INTERNAL RETURN CODE
(*    TRANS_STACK   I/O  TRANSACTION STACK
(*    CLASS_FLAG    I/O  INDICATES IF ENTITIES AND CLASSES ARE
(*                       DEFINED WITHIN THE CLASS
(*    CLS_ENT_HEAD  I/O  POINTER TO LIST OF CLASS ENTITIES
(*
(*  $COMMONS:
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(*  $PROCESSING DESCRIPTION:
(*
(*    INITIALIZE VARIABLES
(*    REMOVE CLASS NAME AND NUMBER FROM LIST AND PUSH THEM ONTO
(*    TRANSACTION STACK
(*    REMOVE EACH ENTITY KEY FROM THE LIST AND PUSH THEM ONTO TRANS-
(*    ACTION STACK
(*    PUSH FINAL CLASS TRANSACTION ONTO THE STACK AND PROCESS THE
(*    STACK.
(*    REINITIALIZE VARIABLES
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
```

CI PS560240032U
April 1990

```
(*
(*   ORIGINATED: 03/20/87       G. H. MOHME             DBMA      *)
(*                                     *)
(*-----*)
(*                                     *)
(*END-----*)
(* END %INCLUDE BLDCLS *)
```

```
(* %INCLUDE BLDDFTYP *)
(**)
PROCEDURE BLDDFTYP(VAR IRC : RET_REC;
CONST DEFINED_TYPE_DATA : TRANSACTION;
CONST CNST_KEY : ENTKEY;
VAR DEFINED_TYPE_KEY : ENTKEY);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE BUILDS THE DEFINED TYPE ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== == =====
(* IRC 0 INTERNAL RETURN CODE
(* DEFINED_TYPE_DATA I TRANSACTION DATA OF THE DEFINED
(* TYPE ENTITY
(* CNST_KEY I KEY TO THE CONSTITUENT
(* DEFINED_TYPE_KEY 0 KEY TO THE DEFINED TYPE ENTITY
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THE ADB DATA IS ASSIGNED AND THEN THE MAS ROUTINE MAECR
(* IS CALLED TO MODEL THE DEFINED TYPE ENTITY.
(*
(* $COMMENTS:
(* NONE
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE BLDEITEM *)
(**)
  PROCEDURE BLDEITEM(VAR IRC : RET_REC;
                     CONST ENUMERITEM_DATA : TRANSACTION;
                     VAR ENUMERITEM_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS ROUTINE BUILDS THE ENUMERITEM ENTITY. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   IRC           0    INTERNAL RETURN CODE *)
(*)   ENUMERITEM_DATA I    TRANSACTION DATA OF THE ENUMERITEM *)
(*)                   ENTITY *)
(*)   ENUMERITEM_KEY 0    KEY TO THE ENUMERITEM ENTITY *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   A LIST IS MADE OF ALL THE ENUMERITEM ENTITIES. *)
(*)   IF THE LIST IS NOT EMPTY THEN IT IS SEARCHED FOR A MATCH *)
(*)   UNTIL ONE IS FOUND OR THE END OF THE LIST IS REACHED. *)
(*)   IF A MATCH IS FOUND ITS KEY IS PASSED BACK TO THE CALLING *)
(*)   PROCEDURE OTHERWISE A NEW ENUMERITEM ENTITY IS CREATED. *)
(*) *)
(*) $COMMENTS: *)
(*)   NONE *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```

```

(** %INCLUDE BLDENT *)
(**)
  PROCEDURE BLDENT(VAR IRC : RET_REC;
                  CONST ENTITY_DATA : TRANSACTION;
                  VAR ENTITY_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE ENTITY ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   ENTITY_DATA   I    TRANSACTION DATA OF THE ENTITY ENTITY
(*   ENTITY_KEY    0    KEY TO THE ENTITY ENTITY
(*
(* $COMMONS:
(*   REF
(*   CURRENT_LIST  I/O  LIST CURRENTLY IN USE CONTAINING THE
(*                       CONSTITUENTS OF THE ENTITY ENTITY
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(*   ROUTINE MAECR IS CALLED TO CREATE THE ENTITY ENTITY.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*

```



```
(* %INCLUDE BLDENUMR *)
(**)
  PROCEDURE BLDENUMR(VAR IRC : RET_REC;
                    VAR ENUMERATION_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE ENUMERATION ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   ENUMERATION_KEY 0    KEY TO THE ENUMERATION ENTITY
(*
(* $COMMONS:
(*   REF
(*   CURRENT_LIST   I/O  A KEY TO THE LIST OF CONSTITUENTS
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE ADB DATA IS ASSIGNED, AND THEN THE KEY TO THE COMMON
(*   CURRENT_LIST, WHICH CONTAINS THE ENUMERATION CONSTITUENTS,
(*   AS WELL AS THE ADB IS PASSED TO THE MAS ROUTINE MAECR
(*   WHICH MODELS THE ENUMERATION ENTITY.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE BLDFIELD *)
(**)
PROCEDURE BLDFIELD(VAR IRC : RET_REC;
                   CONST FIELD_DATA : TRANSACTION;
                   CONST CNST_KEY : ENTKEY;
                   VAR FIELD_KEY : ENTKEY);

SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE BUILDS THE FIELD ENTITY. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC O INTERNAL RETURN CODE *)
(* FIELD_DATA I TRANSACTION DATA OF THE FIELD ENTITY *)
(* CNST_KEY I KEY TO THE CONSTITUENT *)
(* FIELD_KEY O KEY TO THE FIELD ENTITY *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* THE ADB DATA IS ASSIGNED AND THEN THE MAS ROUTINE MAECR *)
(* IS CALLED TO MODEL THE FIELD ENTITY. *)
(* *)
(* $COMMENTS: *)
(* NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(** %INCLUDE BLDGBLFD *)
(**)
PROCEDURE BLDGBLFD(VAR   IRC           : RET_REC;
                   CONST GLOBAL_DATA   : TRANSACTION;
                   VAR   GLOBAL_FIELD_KEY : ENTKEY);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE GLOBAL FIELD ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME           I/O DESCRIPTION
(*   ====           == =====
(*   IRC            0  INTERNAL RETURN CODE
(*   GLOBAL_DATA    I  GLOBAL FIELD DATA
(*   GLOBAL_FIELD_KEY 0  KEY TO THE GLOBAL FIELD ENTITY
(*
(* $COMMONS:
(*   REF
(*   CURRENT_LIST   I/O LIST CURRENTLY IN USE CONTAINING THE
(*                       CONSTITUENTS OF THE GLOBAL FIELD
(*                       ENTITY
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(*   ROUTINE MAEGR IS CALLED TO CREATE THE GLOBAL FIELD
(*   ENTITY.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*

```

```
(* %INCLUDE BLDINT *)
(**)
PROCEDURE BLDINT(VAR IRC : RET_REC;
CONST INTEGER_DATA : TRANSACTION;
VAR INTEGER_KEY : ENTKEY);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE BUILDS THE INTEGER ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(* NAME I/O DESCRIPTION
(*
(* ====
(* IRC 0 INTERNAL RETURN CODE
(* INTEGER_DATA I TRANSACTION DATA OF THE INTEGER ENTITY
(* INTEGER_KEY 0 KEY TO THE INTEGER ENTITY
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* A LIST IS MADE OF ALL THE INTEGER ENTITIES.
(* IF THE LIST IS NOT EMPTY THEN IT IS SEARCHED FOR A MATCH
(* UNTIL ONE IS FOUND OR THE END OF THE LIST IS REACHED.
(* IF A MATCH IS FOUND ITS KEY IS PASSED BACK TO THE CALLING
(* PROCEDURE OTHERWISE A NEW INTEGER ENTITY IS CREATED.
(*
(* $COMMENTS:
(* NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE BLDLOG *)
(**)
  PROCEDURE BLDLOG(VAR IRC : RET_REC;
                   VAR LOGICAL_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE BUILDS THE LOGICAL ENTITY. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ===  ===== *)
(*   IRC           0    INTERNAL RETURN CODE *)
(*   LOGICAL_KEY   0    KEY TO THE LOGICAL ENTITY *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   A LIST IS MADE OF ALL THE LOGICAL ENTITIES. *)
(*   IF THE LIST IS NOT EMPTY THEN A LOGICAL ENTITY IS CREATED, *)
(*   OTHERWISE THE KEY TO THE EXISTING LOGICAL ENTITY IS *)
(*   PASSED BACK TO THE CALLING PROCEDURE. *)
(* *)
(* $COMMENTS: *)
(*   NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE BLDPNTR *)
(**)
  PROCEDURE BLDPNTR(VAR IRC : RET_REC;
                    VAR POINTER_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE POINTER ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ===          ===  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   POINTER_KEY   0    KEY TO THE POINTER ENTITY
(*
(* $COMMONS:
(*   REF
(*   CURRENT_LIST   I/O  LIST CURRENTLY IN USE CONTAINING THE
(*                        CONSTITUENTS OF THE POINTER ENTITY
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(*   ROUTINE MAECR IS CALLED TO CREATE THE POINTER ENTITY.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE BLDREAL *)
(**)
  PROCEDURE BLDREAL(VAR IRC : RET_REC;
                    CONST REAL_DATA : TRANSACTION;
                    VAR REAL_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE REAL ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   REAL_DATA     I    TRANSACTION DATA OF THE REAL ENTITY
(*   REAL_KEY      0    KEY TO THE REAL ENTITY
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   A LIST IS MADE OF ALL THE REAL ENTITIES.
(*   IF THE LIST IS NOT EMPTY THEN IT IS SEARCHED FOR A MATCH
(*   UNTIL ONE IS FOUND OR THE END OF THE LIST IS REACHED.
(*   IF A MATCH IS FOUND ITS KEY IS PASSED BACK TO THE CALLING
(*   PROCEDURE OTHERWISE A NEW REAL ENTITY IS CREATED.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE BLDSSCMA *)
(**)
PROCEDURE BLDSSCMA(VAR IRC : RET_REC;
CONST SUBSCHEMA_DATA : TRANSACTION;
VAR SUBSCHEMA_KEY : ENTKEY);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE BUILDS THE SUBSCHEMA ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ====
(* IRC O INTERNAL RETURN CODE
(* SUBSCHEMA_DATA I TRANSACTION DATA OF THE SUBSCHEMA
(* ENTITY
(* SUBSCHEMA_KEY O KEY TO THE SUBSCHEMA ENTITY
(*
(* $COMMONS:
(* REF
(* CURRENT_LIST I/O LIST CURRENTLY IN USE CONTAINING THE
(* CONSTITUENTS OF THE SUBSCHEMA ENTITY
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(* ROUTINE MAEGR IS CALLED TO CREATE THE SUBSCHEMA ENTITY.
(*
(* $COMMENTS:
(* NONE
(*
(* $CHANGE CONTROL:
(*
```



```

(* %INCLUDE BLDSTRNG *)
(**)
  PROCEDURE BLDSTRNG(VAR IRC : RET_REC;
                    CONST STRING_DATA : TRANSACTION;
                    VAR STRING_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE STRING ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   STRING_DATA   I    TRANSACTION DATA OF THE STRING ENTITY
(*   STRING_KEY    0    KEY TO THE STRING ENTITY
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   A LIST IS MADE OF ALL THE STRING ENTITIES.
(*   IF THE LIST IS NOT EMPTY THEN IT IS SEARCHED FOR A MATCH
(*   UNTIL ONE IS FOUND OR THE END OF THE LIST IS REACHED.
(*   IF A MATCH IS FOUND ITS KEY IS PASSED BACK TO THE CALLING
(*   PROCEDURE OTHERWISE A NEW STRING ENTITY IS CREATED.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*

```

```

(*) %INCLUDE BLDSTRUC *)
(**)
  PROCEDURE BLDSTRUC(VAR IRC : RET_REG;
                    VAR STRUCTURE_KEY : ENTIKEY);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE BUILDS THE STRUCTURE ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   IRC            0    INTERNAL RETURN CODE
(*)   STRUCTURE_KEY  0    KEY TO THE STRUCTURE ENTITY
(*)
(*) $COMMONS:
(*)   REF
(*)   CURRENT_LIST   I/O  LIST CURRENTLY IN USE CONTAINING THE
(*)                      CONSTITUENTS OF THE STRUCTURE ENTITY
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(*)   ROUTINE MAECR IS CALLED TO CREATE THE STRUCTURE ENTITY.
(*)
(*) $COMMENTS:
(*)   NONE
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE BLDSUB *)
(**)
PROCEDURE BLDSUB(VAR IRC           : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     BATCH INTERFACE ROUTINE THAT CREATES A SUBSCHEMA ENTITY IN
(*     THE SCHEMA MODEL FROM A SUBSCHEMA NAME AND LIST OF
(*     CONSTITUENTS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME           I/O  DESCRIPTION
(*     ===           ===  =====
(*     IRC            0    INTERNAL RETURN CODE
(*     TRANS_STACK    I/O  TRANSACTION STACK
(*     SUBSCHEMA_FLAG I/O  INDICATES IF ENTITIES AND CLASSES ARE
(*                          DEFINED WITHIN THE SUBSCHEMA
(*     SUB_ENT_HEAD   I/O  POINTER TO LIST OF SUBSCHEMA ENTITIES
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*     INITIALIZE VARIABLES
(*     REMOVE SUBSCHEMA NAME FROM LIST AND PUSH IT ONTO TRANSACTION
(*     STACK
(*     REMOVE EACH ENTITY KEY FROM THE LIST AND PUSH THEM ONTO TRANS-
(*     ACTION STACK
(*     PUSH FINAL SUBSCHEMA TRANSACTION ONTO THE STACK AND PROCESS
(*     THE STACK
(*     REINITIALIZE THE VARIABLES
(*
```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* ORIGINATED: 03/20/87 C. H. MOHME DBMA *)
(* *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE BLDSUB *)
```

```

(*) %INCLUDE BLDSUPER *)
(**)
  PROCEDURE BLDSUPER(VAR   IRC : RET_REC;
                     CONST SUPERTYPE_DATA : TRANSACTION;
                     VAR   SUPERTYPE_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE BUILDS THE SUPERTYPE ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ===  =====
(*)   IRC             0    INTERNAL RETURN CODE
(*)   SUPERTYPE_DATA  I    TRANSACTION DATA OF THE SUPERTYPE
(*)                       ENTITY
(*)   SUPERTYPE_KEY   0    KEY TO THE SUPERTYPE ENTITY
(*)
(*) $COMMONS:
(*)   REF
(*)   CURRENT_LIST    I/O  LIST CURRENTLY IN USE CONTAINING THE
(*)                       CONSTITUENTS OF THE ENTITY ENTITY
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THE ADB DATA IS ASSIGNED TO VARIABLES THEN THE MAS
(*)   ROUTINE MAECR IS CALLED TO CREATE THE ENTITY ENTITY.
(*)
(*) $COMMENTS:
(*)   NONE
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE BLDUNRES *)
(**)
  PROCEDURE BLDUNRES(VAR IRC : RET_REC;
                    VAR UNRESOLVED_DATA : TRANSACTION;
                    VAR UNRESOLVED_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE BUILDS THE UNRESOLVED ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   UNRESOLVED_DATA 0    TRANSACTION DATA OF THE UNRESOLVED
(*                       ENTITY
(*   UNRESOLVED_KEY  0    KEY TO THE UNRESOLVED ENTITY
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   CREATE THE UNRESOLVED ENTITY BY CALLING MAS ROUTINES.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE BLEXICAL *)
(**)
PROCEDURE BLEXICAL( VAR    TOKEN           : T_TOKEN;
                    VAR    TOKEN_VALUE      : T_TOKEN_VALUE;
                    VAR    TOKEN_LOCATION   : INTEGER;
                    VAR    TOKEN_LENGTH     : INTEGER;
                    VAR    REPORT1         : TEXT);
    SUBPROGRAM;

(* *)
(* $FUNCTION: *)
(* LOCATE THE LONGEST POSSIBLE LEXEME FROM WHICH A TOKEN MAY *)
(* BE DETERMINED, EXCLUDING COMMENTS. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME      I/O  DESCRIPTION *)
(* =====  ==  ===== *)
(* TOKEN      I/O  INPUT  = TOKEN TYPE FROM PREVIOUS CALL *)
(*              OR INITIALIZATION FLAG *)
(*              OUTPUT = CURRENT TOKEN TYPE *)
(* TOKEN_VALUE  O   CURRENT TOKEN VALUE *)
(* TOKEN_LOCATION O  START LOCATION OF TOKEN IN REPORT LINE *)
(* TOKEN_LENGTH  O  LENGTH OF TOKEN IN REPORT LINE *)
(* REPORT1      O   REPORT FILE FOR ECHOING THE INPUT AND *)
(*              REPORTING ERROR MESSAGES *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* LOOP UNTIL TOKEN IS NOT A COMMENT *)
(* GET NEXT TOKEN *)
(* END LOOP *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* ORIGINATED: 03/26/87      C. H. MOHME      DBMA *)
(* *)
(* REVISED:      7 DEC 87, G. A. WHITE, ADD PARAMETERS FOR *)
```

CI PS560240032U
April 1990

(* LOCATION AND LENGTH OF TOKEN IN REPORT LINE. *)
(* *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE BLEXICAL *)


```
(* %INCLUDE BSCINCLD *)
(**)
  PROCEDURE BSCINCLD(VAR IRC          : RET_REC;
                    VAR SUBSCHEMA_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GENERATES THE PASCAL INCLUDE FILES AND WRITES
(*   THESE DEFINITIONS TO A FILE
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           I/O  RETURN CODE
(*   MSG           I/O  PANEL MESSAGE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY A PANEL CONTAINING A LIST OF ALL OF THE SUBSCHEMAS
(*                                     WITHIN THE SCHEMA MODEL.
(*   IF RETURN OR EXIT WAS NOT CHOSEN ON THE PANEL THEN
(*   IF A MULTIPLE SELECT WAS MADE ON THE PANEL THEN
(*   DISPLAY AN ERROR MESSAGE
(*   ELSE
(*   GET THE KEY OF THE SUBSCHEMA SELECTED
(*   IF THE SUBSCHEMA HAS NOT BEEN PHYSICALIZED THEN
(*   PHYSICALIZE THE SUBSCHEMA
(*   WRITE OUT TO THE FILE THE HEADING FOR THE INCLUDES
(*   MAKE AN INCLUSIVE LIST OF ENTITIES WITHIN THE SUBSCHEMA
(*   DELETE ANY DUPLICATES ON THE LIST
(*   ALPHABETICALLY SORT THE ENTITIES
(*   WRITE OUT TO THE FILE THE ENTITY KIND CONSTANTS
(*   WRITE OUT TO THE FILE THE DEFINED TYPE DECLARATIONS
(*   WRITE OUT TO THE FILE THE ENTITY DECLARATIONS
(*   WRITE OUT TO THE FILE THE MAS ENTITY DECLARATIONS
(*   WRITE OUT TO THE FILE THE KEYBLOCK DECLARATIONS
```

CI PS560240032U
April 1990

```
(*      ELSE                                     *)
(*      IF RETURN WAS SELECTED THEN             *)
(*      RETURN TO THE REPORT MENU               *)
(*      ELSE                                     *)
(*      IF EXIT WAS SELECTED THEN               *)
(*      RETURN TO THE MAIN MENU                 *)
(*      ELSE                                     *)
(*      DISPLAY AN ERROR MESSAGE FOR AN INVALID *)
(*      OPTION                                  *)
(*      END;                                    *)
(*      $COMMENTS:                             *)
(*      $CHANGE CONTROL:                       *)
(*)
```

```
(* %INCLUDE BSGTRSPR *)
(**)
PROCEDURE BSGTRSPR(VAR IRC : RET_REC;
                   VAR TRANS_STACK : TRANSPTR);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE BEGINS THE PROCESSING OF THE TRANSACTION
(* STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* IRC 0 RETURN CODE
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK
(*
(* $COMMONS:
(* REF
(* CURRENT_LIST I/O POINTS TO THE LIST OF KEYS
(* CURRENTLY IN USE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE CALLS SCPOPTR TO POP THE TRANSACTION STACK.
(* THEN EACH TRANSACTION IS PROCESSED ACCORDING TO ITS TYPE
(* BY CALLING THE APPROPRIATE ROUTINE TO MODEL THE ENTITIES.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE CLRSTK *)
(**)
  PROCEDURE CLRSTK(VAR IRC          : RET_REG;
                   VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*                                           *)
(*  $FUNCTION:                                           *)
(*    BATCH INTERFACE ROUTINE THAT CLEARS THE TRANSACTION *)
(*    PROCESSING STACK.                                   *)
(*                                           *)
(*  $DESCRIPTION OF ARGUMENTS:                           *)
(*    NAME          I/O  DESCRIPTION                     *)
(*    ====          ===  =====                       *)
(*    IRC            0   INTERNAL RETURN CODE             *)
(*    TRANS_STACK    I/O  TRANSACTION STACK               *)
(*                                           *)
(*  $COMMONS:                                           *)
(*                                           *)
(*  $ENVIRONMENT:                                       *)
(*    LANGUAGE: IBM PASCAL                               *)
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381             *)
(*                                           *)
(*  $EXECUTION PROCEDURE:                               *)
(*    INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*                                           *)
(*  $PROCESSING DESCRIPTION:                             *)
(*                                           *)
(*    INITIALIZE VARIABLES                               *)
(*    WHILE THE STACK IS NOT EMPTY, POP TRANSACTION      *)
(*                                           *)
(*  $COMMENTS:                                           *)
(*                                           *)
(*  $CHANGE CONTROL:                                     *)
(*                                           *)
(*    ORIGINATED: 03/20/87      C. H. MOHME              DBMA *)
(*-----*)
(*END-----*)
(* END %INCLUDE CLRSTK *)
```

```
(* %INCLUDE CRARRAY *)
(**)
PROCEDURE CRARRAY(VAR MESS      : MESSAGE;
                  VAR LBND      : CHAR8;
                  VAR HBND      : CHAR8;
                  VAR ATYPE     : ENTITY_TYPE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR CONTEXT   : T_CONTEXT;
                  VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE CREATE ARRAY MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==
(*   MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   LBND          O    THE LOWER BOUND OF THE ARRAY
(*   HBND          O    THE UPPER BOUND OF THE ARRAY
(*   ATYPE         O    THE ARRAY TYPE
(*   NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   CONTEXT       I    THE CONTEXT IN WHICH THE DEFINED TYPE
(*                   IS BEING CREATED
(*   RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE ARRAY PANEL (CRARRAY) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED
```

```
(*      FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.      *)
(*)                                                                    *)
(*) $COMMENTS:                                                            *)
(*)      NONE                                                            *)
(*)                                                                    *)
(*) $CHANGE CONTROL:                                                    *)
(*)                                                                    *)
(*)      REVISED: JANUARY 1988      C. H. MOHME      DBMA      *)
(*)      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE      *)
(*)                                                                    *)
(*)      REVISED: 08/13/87      C. H. MOHME      DBMA      *)
(*)      CHANGED PANEL OPTION NUMBERS; ADDED LIST AND SET.  NOTE: THE *)
(*)      LIST AND SET DATA TYPES ARE IMPLEMENTED IN THE SOFTWARE AS AN *)
(*)      ARRAY.  A FIELD WAS ADDED TO THE ARRAY ADB TO SPECIFY WHETHER *)
(*)      THE ARRAY IS A CONCEPTUAL ARRAY, LIST, OR SET.      *)
(*)                                                                    *)
(*)      REVISED: 07/02/87      C. H. MOHME      DBMA      *)
(*)      CHANGED CURSOR POSITIONING.      *)
(*)                                                                    *)
(*)      ORIGINATED: 02/04/86      G. A. TUCKER      FRMI      *)
(*)                                                                    *)
(*)-----*)
(*)                                                                    *)
(*)END-----*)
(* END %INCLUDE CRARRAY *)
```

```
(* %INCLUDE CRCLASS1 *)
(**)
PROCEDURE CRCLASS1(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR KNUM      : CHAR8;
                   VAR COMMENT   : CHAR150;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE CREATE CLASS1 PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      O    THE CLASS NAME
(*   KNUM      O    THE CLASS KIND NUMBER
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*       NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE CLASS PANEL NUMBER ONE (CRCLASS1) BY
(*   MAKING ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED
(*   INTO AN ENUMERATED TYPE. THIS DATA AS WELL AS OTHER
(*   INFORMATION GATHERED FROM THE PANEL IS PASSED BACK TO THE
(*   CALLING PROCEDURE.
(*
(* $COMMENTS:
```

CI PS560240032U
April 1990

(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)


```
(* %INCLUDE CRCLASS2 *)
(**)
PROCEDURE CRCLASS2(VAR MESS      : MESSAGE;
                   VAR KNUM      : CHAR8;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE CREATE CLASS PANEL 2 MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   KNUM      O   THE MEMBER KIND NUMBER
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISLAY THE CREATE CLASS PANEL NUMBER TWO (CRCLASS2) BY
(*   MAKING ISPLNK CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO
(*   AN ENUMERATED TYPE.  THIS DATA AS WELL AS OTHER INFORMATION
(*   GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO-
(*   CEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE CRDEFTYP *)
(**)
PROCEDURE CRDEFTYP(VAR MESS      : MESSAGE;
                   VAR NAME      : IDCHAR;
                   VAR FTYPE     : ENTITY_TYPE;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR CONTEXT   : T_CONTEXT;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
*)
*)
$FUNCTION:
*)
THIS FUNCTION:
*)
DISPLAYS THE CREATE DEFINED TYPE MENU
*)
*)
$DESCRIPTION OF ARGUMENTS:
*)
NAME      I/O  DESCRIPTION
*)
=====
*)
MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
*)
NAME      O   THE NAME OF THE DEFINED TYPE ENTERED
*)
FTYPE     O   TYPES INTEGER,STRING,REAL...ETC.
*)
NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
*)
              OPERATION
*)
CONTEXT   I   THE CONTEXT IN WHICH THE DEFINED TYPE
*)
              IS BEING CREATED
*)
RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
*)
              IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
*)
*)
$COMMONS:
*)
NONE
*)
*)
$ENVIRONMENT:
*)
LANGUAGE: IBM PASCAL
*)
HARDWARE SYSTEM: IBM 360/370/4341/4381
*)
DDNAMES USED WITH STANDARD FILES:
*)
NONE
*)
*)
$EXECUTION PROCEDURE:
*)
SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
*)
*)
$PROCESSING DESCRIPTION:
*)
DISPLAY THE CREATE DEFINED TYPE PANEL (CRDEFTYP) BY MAKING
*)
ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
*)
ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
*)
GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO-
*)
CEDURE.
*)
*)
```

```
(* $COMMENTS: *)
(* NONE *)
(* $CHANGE CONTROL: *)
(*
(* REVISED: JANUARY 1988 C. H. MOHME DBMA *)
(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE. *)
(*
(* REVISED: 08/13/87 C. H. MOHME DBMA *)
(* CHANGED PANEL OPTION NUMBERS; ADDED LIST AND SET. NOTE: THE *)
(* LIST AND SET DATA TYPES ARE IMPLEMENTED IN THE SOFTWARE AS AN *)
(* ARRAY. A FIELD WAS ADDED TO THE ARRAY ADB TO SPECIFY WHETHER *)
(* THE ARRAY IS A CONCEPTUAL ARRAY, LIST, OR SET. *)
(*
(* REVISED: 07/02/87 C. H. MOHME DBMA *)
(* CHANGED CURSOR POSITIONING. *)
(*
(* ORIGINATED: 02/04/86 G. A. TUCKER FRMI *)
(*
(*-----*)
(*
(*END-----*)
(* END %INCLUDE CRDEFTYP *)
```

```
(* %INCLUDE CRENTITY *)
(**)
PROCEDURE CRENTITY(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR KNUM      : CHAR8;
                   VAR REFR      : CHAR8;
                   VAR COMMENT   : CHAR150;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE CREATE ENTITY MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*      ====      ==  =====
(*      MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      NAME      O   THE ENTITY NAME
(*      KNUM      O   THE ENTITY KIND NUMBER
(*      REFR      O   INDICATES IF AN ENTITY REFERENCES A
(*                      SUPERTYPE
(*      NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                      OPERATION
(*      RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                      IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE CREATE ENTITY PANEL (CRENTITY) BY MAKING ISPLNK
(*      CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*      TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED FROM
(*      THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
```

CI PS560240032U
April 1990

```
(* *)
(* $COMMENTS: *)
(* NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* REVISED: 09/28/87 C. H. MOHME DBMA *)
(* CHANGED TO INCORPORATE THE SUPERTYPE DATA TYPE. *)
(* *)
(* REVISED: 08/13/87 C. H. MOHME DBMA *)
(* CHANGED PANEL OPTION NUMBERS. *)
(* *)
(* REVISED: 07/02/87 C. H. MOHME DBMA *)
(* CHANGED CURSOR POSITIONING. *)
(* *)
(* ORIGINATED: 03/14/86 G. A. TUCKER FRMI *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE CRENTITY *)
```

```

(*) %INCLUDE CRENUM *)
(**)
  PROCEDURE CRENUM(VAR MESS      : MESSAGE;
                   VAR NAME      : IDCHAR;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

    SUBPROGRAM;

(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS FUNCTION:
(*)       DISPLAYS THE CREATE ENUMERATION MENU
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O  DESCRIPTION
(*)   =====
(*)   MESS       I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*)   NAME       O    THE NAME OF THE ENUMERATION FROM PANEL
(*)   NEXT_OP    O    ENUMERATED TYPE INDICATING THE NEXT
(*)                   OPERATION
(*)   XRC        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*)                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)   DDNAMES USED WITH STANDARD FILES:
(*)   NONE
(*)
(*) $EXECUTION PROCEDURE:
(*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   DISPLAY THE CREATE ENUMERATION PANEL (CRENUM) BY MAKING
(*)   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*)   ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
(*)   GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING
(*)   PROCEDURE.
(*)
(*) $COMMENTS:
(*)   NONE
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE CRFIELD *)
(**)
```

```
PROCEDURE CRFIELD(VAR MESS      : MESSAGE;
                  VAR NAME      : T_NAME;
                  VAR POS       : CHAR8;
                  (* VAR PURP    : CHAR8; *)
                  VAR REQD      : CHAR8;
                  (* VAR DEPD    : CHAR8; *)
                  VAR COM       : CHAR50;
                  VAR FTYPE     : ENTITY_TYPE;
                  VAR FLDTYPE   : T_FIELDTYPE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);
```

SUBPROGRAM;

```
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS PROCEDURE : *)
(* DISPLAYS THE CREATE FIELD PANEL *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* MESS I THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(* NAME O THE NAME OF THE FIELD *)
(* POS O THE POSITION OF THE FIELD IN THE ADB *)
(* PURP O THE PURPOSE OF THE FIELD *)
(* REQD O REQUIREDNESS, OPTIONAL OR NOT, OF FIELD *)
(* DEPD O DEPENDENCE OF THE FIELD *)
(* FTYPE O ENTITY TYPE *)
(* FLDTYPE O THE FIELD TYPE *)
(* NEXT_OP O ENUMERATED TYPE INDICATING THE NEXT *)
(* OPERATION *)
(* RR O INDICATES IF AN ERROR HAS OCCURRED AND, *)
(* IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
```

```
(*      DDNAMES USED WITH STANDARD FILES:                                *)
(*      NONE                                                                *)
(*                                                                           *)
(* $EXECUTION PROCEDURE:                                                  *)
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE                          *)
(*                                                                           *)
(* $PROCESSING DESCRIPTION:                                              *)
(*      DISPLAY THE CREATE FIELD PANEL (CRFIELD) BY MAKING ISPLNK        *)
(*      CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED        *)
(*      TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED FROM      *)
(*      THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.                *)
(*                                                                           *)
(* $COMMENTS:                                                            *)
(*      NONE                                                                *)
(*                                                                           *)
(* $CHANGE CONTROL:                                                      *)
(*                                                                           *)
(*      REVISED: JANUARY 1988      C. H. MOHME      DBMA      *)
(*      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE                  *)
(*                                                                           *)
(*      REVISED: 09/28/87          C. H. MOHME      DBMA      *)
(*      CHANGED TO INCORPORATE THE SUPERTYPE FIELD.                      *)
(*                                                                           *)
(*      REVISED: 08/13/87          C. H. MOHME      DBMA      *)
(*      CHANGED PANEL OPTION NUMBERS; CHANGED FIELD ADB DATA; ADDED     *)
(*      LIST AND SET. NOTE: THE LIST AND SET DATA TYPES ARE IMPLE-     *)
(*      MENTED IN THE SOFTWARE AS AN ARRAY. A FIELD WAS ADDED TO THE     *)
(*      ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL ARRAY,  *)
(*      LIST, OR SET.                                                    *)
(*                                                                           *)
(*      REVISED: 07/02/87          C. H. MOHME      DBMA      *)
(*      CHANGED CURSOR POSITIONING.                                         *)
(*                                                                           *)
(*      ORIGINATED: 03/14/86       G. A. TUCKER      FRMI      *)
(*                                                                           *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE CRFIELD *)
```



```
(* %INCLUDE CRINTGR *)
(**)
PROCEDURE CRINTGR(VAR MESS      : MESSAGE;
                  VAR PREC      : CHAR8;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE CREATE INTEGER MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   PREC       O   THE PRECISION OF THE INTEGER ENTERED
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE INTEGER PANEL (CRINTGR) BY MAKING
(*   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
(*   GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING
(*   PROCEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
```

```
(*
(*   REVISED: 10/09/87           G. H. MOHME           DBMA   *)
(*   ADDED DEFAULT PRECISION.                                     *)
(*                                                                    *)
(*   REVISED: 08/13/87           G. H. MOHME           DBMA   *)
(*   CHANGED PANEL OPTION NUMBERS.                               *)
(*                                                                    *)
(*   REVISED: 07/02/87           G. H. MOHME           DBMA   *)
(*   CHANGED CURSOR POSITIONING.                                   *)
(*                                                                    *)
(*   ORIGINATED: 04/08/86         G. A. TUCKER           FRMI   *)
(*                                                                    *)
(*-----*)
(*                                                                    *)
(*END-----*)
(* END %INCLUDE CRINTGR *)
```

```
(* %INCLUDE CRLIST *)
(**)
  PROCEDURE CRLIST(VAR MESS      : MESSAGE;
                  VAR MIN       : CHAR8;
                  VAR MAX       : CHAR8;
                  VAR ATYPE     : ENTITY_TYPE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR CONTEXT    : T_CONTEXT;
                  VAR RR        : RET_REC);

    SUBPROGRAM;
  (**)
  (*-----*)
  (*
  (* $FUNCTION:
  (*   THIS FUNCTION:
  (*           DISPLAYS THE CREATE LIST PANEL
  (*
  (* $DESCRIPTION OF ARGUMENTS:
  (*   NAME      I/O  DESCRIPTION
  (*   ====      ==  =====
  (*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
  (*   MIN        0   THE MINIMUM NUMBER OF OCCURRENCES IN THE
  (*               LIST
  (*   MAX        0   THE MAXIMUM NUMBER OF OCCURRENCES IN THE
  (*               LIST
  (*   ATYPE      0   THE LIST TYPE
  (*   NEXT_OP    0   ENUMERATED TYPE INDICATING THE NEXT
  (*               OPERATION
  (*   CONTEXT    I   THE CONTEXT IN WHICH THE DEFINED TYPE
  (*               IS BEING CREATED
  (*   RR         0   INDICATES IF AN ERROR HAS OCCURRED AND,
  (*               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
  (*
  (* $COMMONS:
  (*   NONE
  (*
  (* $ENVIRONMENT:
  (*   LANGUAGE: IBM PASCAL
  (*   HARDWARE SYSTEM: IBM 360/370/4341/4381
  (*   DDNAMES USED WITH STANDARD FILES:
  (*   NONE
  (*
  (* $EXECUTION PROCEDURE:
  (*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
  (*
  (* $PROCESSING DESCRIPTION:
  (*   DISPLAY THE CREATE LIST PANEL (CRLIST) BY MAKING ISPLINK
  (*
```

CI PS560240032U
April 1990

```
(*      CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*      TYPE.  THIS DATA AS WELL AS OTHER INFORMATION GATHERED   *)
(*      FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.    *)
(*                                                                  *)
(*  $COMMENTS:                                                       *)
(*      NONE                                                         *)
(*                                                                  *)
(*  $CHANGE CONTROL:                                                 *)
(*                                                                  *)
```

```
(* %INCLUDE CRPNTR *)
(**)
PROCEDURE CRPNTR(VAR MESS      : MESSAGE;
                  VAR KNUM     : CHAR8;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE CREATE POINTER MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   KNUM       O   THE MEMBER KIND NUMBER OF THE POINTER
(*                   INTO AN INTEGER
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE POINTER PANEL (CRPNTR) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED FROM
(*   THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
```

```

(*) %INCLUDE CRREAL *)
(**)
  PROCEDURE CRREAL(VAR MESS      : MESSAGE;
                   VAR SIZE      : CHAR8;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

    SUBPROGRAM;
  (**)
  (*)-----*)
  (*)
  (*) $FUNCTION:
  (*)   THIS FUNCTION:
  (*)       DISPLAYS THE CREATE REAL PANEL
  (*)
  (*) $DESCRIPTION OF ARGUMENTS:
  (*)   NAME      I/O  DESCRIPTION
  (*)   ====      ==  =====
  (*)   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
  (*)   SIZE       O   THE PRECISION OF THE REAL ENTERED
  (*)   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
  (*)                   OPERATION
  (*)   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
  (*)                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
  (*)
  (*) $COMMONS:
  (*)   NONE
  (*)
  (*) $ENVIRONMENT:
  (*)   LANGUAGE: IBM PASCAL
  (*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
  (*)   DDNAMES USED WITH STANDARD FILES:
  (*)   NONE
  (*)
  (*) $EXECUTION PROCEDURE:
  (*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
  (*)
  (*) $PROCESSING DESCRIPTION:
  (*)   DISPLAY THE CREATE REAL PANEL (CRREAL) BY MAKING ISPLNK
  (*)   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
  (*)   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED FROM
  (*)   THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
  (*)
  (*) $COMMENTS:
  (*)   NONE
  (*)
  (*) $CHANGE CONTROL:
  (*)

```

CI PS560240032U
April 1990

```
(*  REVISED: 10/09/87      G. H. MOHME      DBMA      *)
(*  ADDED DEFAULT PRECISION.                                     *)
(*                                                                *)
(*  REVISED: 08/13/87      G. H. MOHME      DBMA      *)
(*  CHANGED PANEL OPTION NUMBERS.                               *)
(*                                                                *)
(*  REVISED: 07/02/86      G. H. MOHME      DBMA      *)
(*  CHANGED CURSOR POSITIONING.                                   *)
(*                                                                *)
(*  ORIGINATED: 04/08/86    G. A. TUCKER      FRMI      *)
(*                                                                *)
(*-----*)
(*                                                                *)
(*END-----*)
(* END %INCLUDE CRREAL*)
```

April 1990

```

(*) %INCLUDE CRSET *)
(**)
  PROCEDURE CRSET(VAR MESS      : MESSAGE;
                  VAR MIN       : CHAR8;
                  VAR MAX       : CHAR8;
                  VAR ATYPE     : ENTITY_TYPE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR CONTEXT   : T_CONTEXT;
                  VAR RR        : RET_REC);

    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS FUNCTION: *)
(*)           DISPLAYS THE CREATE SET PANEL *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O  DESCRIPTION *)
(*)   ====      ==  ===== *)
(*)   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(*)   MIN        O   THE MINIMUM NUMBER OF OCCURRENCES IN THE *)
(*)               SET *)
(*)   MAX        O   THE MAXIMUM NUMBER OF OCCURRENCES IN THE *)
(*)               SET *)
(*)   ATYPE      O   THE SET TYPE *)
(*)   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT *)
(*)               OPERATION *)
(*)   CONTEXT    I   THE CONTEXT IN WHICH THE SET IS BEING *)
(*)               CREATED *)
(*)   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND, *)
(*)               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)   DDNAMES USED WITH STANDARD FILES: *)
(*)   NONE *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   DISPLAY THE CREATE SET PANEL (CRSET) BY MAKING ISPLINK *)

```


CI PS560240032U
April 1990

```
(*      CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED  *)
(*      TYPE.  THIS DATA AS WELL AS OTHER INFORMATION GATHERED    *)
(*      FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.     *)
(*                                                                    *)
(*  $COMMENTS:                                                         *)
(*      NONE                                                           *)
(*                                                                    *)
(*  $CHANGE CONTROL:                                                  *)
(*)                                                                    *)
```

```
(* %INCLUDE CRSTRING *)
(**)
  PROCEDURE CRSTRING(VAR MESS      : MESSAGE;
                    VAR SLEN      : CHAR8;
                    VAR NEXT_OP   : OPERATIONS;
                    VAR RR        : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE CREATE STRING PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   SLEN      O   THE LENGTH OF THE STRING ENTERED
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION.
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*       NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE STRING PANEL (CRSTRING) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED FROM
(*   THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

CI PS560240032U
April 1990

```
(*  REVISED: 10/09/87          C. H. MOHME          DBMA      *)
(*  ADDED DEFAULT STRING LENGTH.                                     *)
(*                                                                    *)
(*  REVISED: 08/13/87          C. H. MOHME          DBMA      *)
(*  CHANGED PANEL OPTION NUMBERS.                                   *)
(*                                                                    *)
(*  REVISED: 07/02/87          C. H. MOHME          DBMA      *)
(*  CHANGED CURSOR POSITIONING.                                       *)
(*                                                                    *)
(*  ORIGINATED: 04/08/86        G. A. TUCKER          FRMI      *)
(*                                                                    *)
(*-----*)
(*                                                                    *)
(*END-----*)
(* END %INCLUDE GRSTRING *)
```

```
(* %INCLUDE CRSUBSCM *)
(**)
PROCEDURE CRSUBSCM(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR COMMENT    : CHAR150;
                   VAR KNUM       : CHAR8;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(* $FUNCTION: *)
(* THIS FUNCTION: *)
(* DISPLAYS THE CREATE SUBSCHEMA PANEL *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* MESS I THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(* NAME O THE SUBSCHEMA NAME *)
(* KNUM O THE SUBSCHEMA MEMBER KIND NUMBER *)
(* NEXT_OP O ENUMERATED TYPE INDICATING THE NEXT *)
(* OPERATION *)
(* RR O INDICATES IF AN ERROR HAS OCCURRED AND, *)
(* IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* DDNAMES USED WITH STANDARD FILES: *)
(* NONE *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* DISPLAY THE CREATE SUBSCHEMA PANEL (CRSUBSCM) BY MAKING *)
(* ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN *)
(* ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION *)
(* GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO- *)
(* CEDURE. *)
(* *)
(* $COMMENTS: *)
(* NONE *)
(* *)
(* $CHANGE CONTROL: *)
```

(* %INCLUDE CRSUPTYP *)

(**)

```

PROCEDURE CRSUPTYP(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR REFR      : CHAR8;
                   VAR CREATE_ONLY : BOOLEAN;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

```

SUBPROGRAM;

(**)

```

(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS PROCEDURE: *)
(*)     DISPLAYS EITHER THE CREATE/REFERENCE SUPERTYPE MENU *)
(*)     OR THE CREATE SUPERTYPE MENU *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O  DESCRIPTION *)
(*)   ====      ==  ===== *)
(*)   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(*)   NAME       O   THE ENTITY NAME *)
(*)   REFR       O   INDICATES IF THE SUPERTYPE REFERENCES *)
(*)                ANOTHER SUPERTYPE *)
(*)   CREATE_ONLY I   INDICATES I *)
(*)   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT *)
(*)                OPERATION *)
(*)   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND, *)
(*)                IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(*)
(*) $COMMONS: *)
(*)   NONE *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)   DDNAMES USED WITH STANDARD FILES: *)
(*)   NONE *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   DISPLAY THE CREATE SUPERTYPE PANEL (CRSUPTYP) BY MAKING *)
(*)   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN *)
(*)   ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION *)

```

CI PS560240032U
April 1990

(*	GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING	*)
(*	PROCEDURE.	*)
(*		*)
(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```
(* %INCLUDE CRURUL. *)
(**)
  PROCEDURE CRURUL(CONST ENTITY_TYPE:ORD_KIND;VAR GROUP:T_GROUP_ARRAY;
    VAR NUM_GROUP:LISTPSTN; VAR MIN_CNST:LISTPSTN);EXTERNAL;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   CREATES THE USER'S RULES.  RULES OF CONNECTIVITY USED TO *)
(*   DETERMINE DELETABILITY OF ENTITIES. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   ENTITY_TYPE    I   ENTITY KIND VALUE WHICH WILL HAVE THE *)
(*                   DELETED RULE *)
(*   GROUP          O   ARRAY THAT WILL BE FILLED WITH THE RULES *)
(*                   AND NUMBER OF CONSTITUENTS OF EACH *)
(*                   DIFFERENT RELATIONSHIP THAT THIS ENTITY *)
(*                   KIND CAN HAVE WITH ITS CONSTITUENTS *)
(*   NUM_GROUP      O   NUMBER OF DIFFERENT RELATIONSHIPS THIS *)
(*                   ENTITY CAN HAVE WITH ITS CONSTITUENTS *)
(*   MIN_CNST       O   MINIMUM NUMBER OF CONSTITUENTS THAT THIS *)
(*                   ENTITY CAN HAVE WHEN IT HAS A GROUP OF *)
(*                   CONSTITUENTS THAT ARE "SECONDARY" *)
(*   RC             O   EXTERNAL RETURN CODE *)
(*                   = 0  OK RETURN CODE *)
(*                   > 0  CRITICAL ERROR *)
(*                   < 0  WARNING *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   ??????ARE SET TO INDICATE IF THE RELATIONSHIP BETWEEN THE *)
(*   USER AND ITS CONSTITUTES IS DEPENDENT OR INDEPENDENT AND *)
(*   STRONG OR WEAK. *)
(*   DEFAULT RULE IS DEPENDENT/STRONG. *)
(* *)
(* $COMMENTS: *)
(* *)
```

```

(*) $CHANGE CONTROL: *)
(*) *)
(*) REVISED: 09/28/87 C. H. MOHME DBMA *)
(*) TO INCORPORATE SUPERTYPE DATA TYPE *)
(*) *)
(*) REVISED: 04/09/87 C. H. MOHME DBMA *)
(*) TO INCORPORATE LIST DATA TYPE *)
(*) *)
(*) REVISED: 09/29/86 L. J. BEHAN DBMA *)
(*) ENTERED THE NEW RULES FOR THE SCHEMA EXECUTIVE ENTITIES *)
(*) *)
(*) REVISED: 06/19/86 B. A. ULMER FRMI *)
(*) REDO LOGIC OF HOW CRURUL WORKS BASED ON THE NEW DELETE RULES *)
(*) *)
(*) REVISED: 09/ /85 B. A. ULMER FRMI *)
(*) ADD ENTITY KINDS SO AS TO TEST THE NEW DELETE RULES (2070, *)
(*) 2080, 2090) *)
(*) *)
(*) REVISED: 09/ /85 B. A. ULMER FRMI *)
(*) ADD PARAMETERS TO HANDLE THE TWO NEW DELETE RULES *)
(*) *)
(*) REVISED: 09/18/84 D. J. KERCHNER FRMI *)
(*) ADDED I/S RULE FOR THE PICK ENTITY *)
(*) *)
(*) ORIGINATED: MM/DD/YY GCWW I. M. THEORIGINATOR GROUP_ID *)
(*) *)
(*)-----*)
%PAGE *)
(*)-----*)
(*) DATA STRUCTURES/MAJOR VARIABLES: *)
(*)-----*)
(*) *)
(*)END-----*)
(**)
(*) END %INCLUDE CRURUL *)

```



```
(* %INCLUDE CSARYWRT *)
(**)
  PROCEDURE CSARYWRT(VAR IRC          : RET_REC;
                    VAR ARRAY_KEY    : ENTKEY;
                    VAR CSRFILE      : TEXT;
                    VAR PAGE_NUMBER  : INTEGER;
                    VAR LINE_COUNT   : INTEGER;
                    VAR INDENT       : INTEGER;
                    VAR PAGE_TYPE    : PAGES);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE WRITES OUT AN ARRAY DEFINITION
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0   INTERNAL RETURN CODE
(*   ARRAY_KEY     I   ARRAY KEY
(*   CSRFILE       I/O  OUTPUT FILE
(*   PAGE_NUMBER   I/O  CURRENT PAGE NUMBER
(*   LINE_COUNT    I/O  CURRENT LINE COUNT
(*   INDENT        I/O  NUMBER OF SPACES TO INDENT
(*   PAGE_TYPE     I/O  TYPE OF REPORT PAGE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(*   CREATE A NEW PAGE WITH HEADING, IF NECESSARY.
(*   GET THE ARRAY'S ADB.
(*   WRITE OUT TO THE FILE THE ARRAY BOUNDS.
(*   SET THE LIST OF CONSTITUENTS OF THE ARRAY TO BE READ IN THE
(*   FORWARD DIRECTION.
(*   READ THE ARRAY CONSTITUENT FROM THE LIST.
(*   GET THE ARRAY CONSTITUENT'S ADB.
```

```
(* CASE CONSTITUENT_ADB.ENT_KIND OF *)
(* INTEGER : WRITE OUT INTEGER DEFINITION *)
(* REAL : WRITE OUT REAL DEFINITION *)
(* STRING : WRITE OUT STRING DEFINITION *)
(* LOGICAL : WRITE OUT LOGICAL DEFINITION *)
(* ARRAY : WRITE OUT ARRAY DEFINITION *)
(* DEFINED TYPE : WRITE OUT DEFINED TYPE DEFINITION *)
(* POINTER : WRITE OUT POINTER DEFINITION *)
(*
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(*) %INCLUDE CSCLSHDG *)
(**)
  PROCEDURE CSCLSHDG(VAR CSRFILE      : TEXT;
                    VAR HEADING      : HEADING_TYPE;
                    VAR PAGE_NUMBER  : INTEGER;
                    VAR LINE_COUNT   : INTEGER);

    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE WRITES OUT A CLASS HEADING ON A NEW PAGE.
(*)
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O  DESCRIPTION
(*)   ====      ==  =====
(*)   CSRFILE    I/O  OUTPUT FILE
(*)   HEADING     I   DEFINITION OR INDEX HEADING
(*)   PAGE_NUMBER I/O  CURRENT PAGE NUMBER
(*)   LINE_COUNT  O   LINE COUNT ON THE CURRENT PAGE
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*)   CREATE A NEW PAGE.
(*)   WRITE OUT TO THE FILE THE APPROPRIATE CLASS HEADING (DEFINITION
(*)     OR INDEX).
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)

```

(* %INCLUDE CSCLSWRT *)

(**)

```
PROCEDURE CSCLSWRT(VAR IRC          : RET_REC;
                   VAR CLASS_LIST   : LISTKEY;
                   VAR CSRFILE      : TEXT;
                   VAR PAGE_NUMBER  : INTEGER;
                   VAR CURRENT_PAGE : PAGE_PTR);
```

SUBPROGRAM;

(**)

```
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE WRITES OUT CLASS DEFINITIONS TO A FILE *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 INTERNAL RETURN CODE *)
(* CLASS_LIST I ALPHABETIZED LIST OF CLASSES *)
(* CSRFILE I/O THE OUTPUT FILE *)
(* PAGE_NUMBER I/O THE CURRENT PAGE NUMBER *)
(* CURRENT_PAGE I/O POINTS TO THE CURRENT PAGE RECORD IN *)
(* THE CHAIN *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* SET THE LIST OF CLASSES TO BE READ IN THE FORWARD DIRECTION. *)
(* COUNT THE NUMBER OF CLASSES IN THE LIST. *)
(* FOR X = 1 TO THE NUMBER OF CLASSES IN THE LIST *)
(* WRITE OUT TO THE FILE THE CLASS HEADING ON A NEW PAGE. *)
(* UPDATE THE PAGE RECORD CHAIN. *)
(* READ A CLASS FROM THE LIST OF CLASSES. *)
(* GET THE CLASS' ADB. *)
(* WRITE OUT TO THE FILE THE CLASS NAME AND NUMBER. *)
(* SET THE LIST OF CONSTITUENTS OF THE CLASS TO BE READ IN THE *)
(* FORWARD DIRECTION. *)
```

```
(*      COUNT THE NUMBER OF CONSTITUENTS IN THE LIST.      *)
(*      FOR Y = 1 TO THE NUMBER OF CLASS CONSTITUENTS      *)
(*      READ A CONSTITUENT FROM THE LIST OF CONSTITUENTS.   *)
(*      GET THE CONSTITUENT'S ADB.                          *)
(*      CREATE A NEW PAGE, IF NECESSARY.                    *)
(*      WRITE OUT TO THE FILE THE CONSTITUENT (CLASS OR ENTITY) *)
(*      INCREMENT THE LINE COUNTER.                        *)
(*      WRITE OUT TO THE FILE 'END;'.                      *)
(*      *)
(*      $COMMENTS:                                          *)
(*      *)
(*      $CHANGE CONTROL:                                    *)
(*      *)
```

```
(* %INCLUDE CSDEFHDG *)
(**)
PROCEDURE CSDEFHDG(VAR CSRFILE      : TEXT;
                   VAR PAGE_NUMBER : INTEGER;
                   VAR LINE_COUNT  : INTEGER);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE WRITES OUT A DEFINED TYPE HEADING ON A NEW
(*     PAGE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ===  =====
(*     CSRFILE       I/O  OUTPUT FILE
(*     PAGE_NUMBER   I/O  CURRENT PAGE NUMBER
(*     LINE_COUNT    O    LINE COUNT ON THE CURRENT PAGE
(*
(* $COMMONS:
(*     NONE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(*     CREATE A NEW PAGE.
(*     WRITE OUT TO THE FILE THE DEFINED TYPE DEFINITION HEADING.
(*     INCREMENT THE LINE COUNTER.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE CSDEFWRT *)
(**)
  PROCEDURE CSDEFWRT(VAR IRC           : RET_REC;
                    VAR DEFINED_TYPE_KEY : ENTKEY;
                    VAR CSRFILE          : TEXT;
                    VAR LINE_COUNT       : INTEGER);

    SUBPROGRAM;

(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE WRITES OUT A DEFINED TYPE NAME
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   IRC            0    INTERNAL RETURN CODE
(*)   DEFINED_TYPE_KEY  I    DEFINED TYPE KEY
(*)   CSRFILE        I/O  THE OUTPUT FILE
(*)   LINE_COUNT      I/O  CURRENT LINE COUNT
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*)   GET THE DEFINED TYPE'S ADB.
(*)   WRITE OUT TO THE FILE THE DEFINED TYPE NAME.
(*)   INCREMENT THE LINE COUNTER.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE CSENMWRT *)
(**)
PROCEDURE CSENMWRT(VAR IRC           : RET_REC;
                   VAR ENUMERATION_KEY : ENTKEY;
                   VAR CSRFILE         : TEXT;
                   VAR PAGE_NUMBER     : INTEGER;
                   VAR LINE_COUNT      : INTEGER;
                   VAR PAGE_TYPE       : PAGES);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE WRITES OUT AN ENUMERATION DEFINITION
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC           0    INTERNAL RETURN CODE
(*     ENUMERATION_KEY I    ENUMERATION KEY
(*     CSRFILE       I/O  THE OUTPUT FILE
(*     PAGE_NUMBER   I/O  CURRENT PAGE NUMBER
(*     LINE_COUNT    I/O  CURRENT LINE COUNT
(*     PAGE_TYPE     I/O  TYPE OF REPORT PAGE
(*
(* $COMMONS:
(*     NONE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(* GET THE ENUMERATION'S ADB.
(* WRITE OUT TO THE FILE 'ENUMERATION OF ('.
(* INCREMENT THE LINE COUNTER.
(* SET THE LIST OF CONSTITUENTS OF THE ENUMERATION TO BE READ IN
(* THE FORWARD DIRECTION.
(* COUNT THE NUMBER OF CONSTITUENTS IN THE LIST.
(* FOR Y = 1 TO THE NUMBER OF CONSTITUENTS
(* READ A CONSTITUENT FROM THE LIST OF CONSTITUENTS.
(* GET THE CONSTITUENT'S ADB.
```


CI PS560240032U
April 1990

(*	CREATE A NEW PAGE, IF NECESSARY.	*)
(*	WRITE OUT TO THE FILE THE CONSTITUENT.	*)
(*	INCREMENT THE LINE COUNTER.	*)
(*	WRITE OUT TO THE FILE ')'	*)
(*		*)
(*	\$COMMENTS:	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

(* %INCLUDE CSENTHDG *)

(**)

```
PROCEDURE CSENTHDG(VAR CSRFILE      : TEXT;  
                   VAR HEADING      : HEADING_TYPE;  
                   VAR PAGE_NUMBER  : INTEGER;  
                   VAR LINE_COUNT   : INTEGER);
```

SUBPROGRAM;

(**)

```
(*-----*)  
(* *)  
(* $FUNCTION: *)  
(* THIS ROUTINE WRITES OUT AN ENTITY HEADING ON A NEW PAGE. *)  
(* *)  
(* *)  
(* $DESCRIPTION OF ARGUMENTS: *)  
(* NAME I/O DESCRIPTION *)  
(* ==== == ===== *)  
(* CSRFILE I/O OUTPUT FILE *)  
(* HEADING I DEFINITION OR INDEX HEADING *)  
(* PAGE_NUMBER I/O CURRENT PAGE NUMBER *)  
(* LINE_COUNT 0 LINE COUNT ON THE CURRENT PAGE *)  
(* *)  
(* $COMMONS: *)  
(* NONE *)  
(* *)  
(* $ENVIRONMENT: *)  
(* LANGUAGE: IBM PASCAL *)  
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)  
(* *)  
(* $EXECUTION PROCEDURE: *)  
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)  
(* *)  
(* $PROCESSING DESCRIPTION: *)  
(* *)  
(* CREATE A NEW PAGE. *)  
(* WRITE OUT TO THE FILE THE APPROPRIATE ENTITY HEADING (DEFINITION *)  
(* OR INDEX). *)  
(* INCREMENT THE LINE COUNTER. *)  
(* *)  
(* $COMMENTS: *)  
(* *)  
(* $CHANGE CONTROL: *)  
(* *)
```

(* %INCLUDE CSENTWRT *)

(**)

```
PROCEDURE CSENTWRT(VAR IRC           : RET_REC;  
                   VAR ENTITY_LIST   : LISTKEY;  
                   VAR CSRFILE       : TEXT;  
                   VAR PAGE_NUMBER   : INTEGER;  
                   VAR CURRENT_PAGE  : PAGE_PTR);
```

SUBPROGRAM;

(**)

```
(*-----*)  
(* *)  
(* $FUNCTION: *)  
(* THIS ROUTINE WRITES OUT ENTITY DEFINITIONS TO A FILE *)  
(* *)  
(* $DESCRIPTION OF ARGUMENTS: *)  
(* NAME I/O DESCRIPTION *)  
(* ==== === *)  
(* IRC 0 INTERNAL RETURN CODE *)  
(* ENTITY_LIST I ALPHABETIZED LIST OF ENTITIES *)  
(* CSRFILE I/O THE OUTPUT FILE *)  
(* PAGE_NUMBER I/O THE CURRENT PAGE NUMBER *)  
(* CURRENT_PAGE I/O POINTS TO THE CURRENT PAGE RECORD IN *)  
(* THE CHAIN. *)  
(* *)  
(* $COMMONS: *)  
(* NONE *)  
(* *)  
(* $ENVIRONMENT: *)  
(* LANGUAGE: IBM PASCAL *)  
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)  
(* *)  
(* $EXECUTION PROCEDURE: *)  
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)  
(* *)  
(* $PROCESSING DESCRIPTION: *)  
(* *)  
(* SET THE LIST OF ENTITIES TO BE READ IN THE FORWARD DIRECTION. *)  
(* COUNT THE NUMBER OF ENTITIES IN THE LIST. *)  
(* FOR X = 1 TO THE NUMBER OF ENTITIES IN THE LIST *)  
(* WRITE OUT TO THE FILE THE ENTITY HEADING. *)  
(* UPDATE THE PAGE RECORD CHAIN. *)  
(* READ AN ENTITY FROM THE LIST OF ENTITIES. *)  
(* GET THE ENTITY'S ADB. *)  
(* WRITE OUT TO THE FILE THE ENTITY NAME AND NUMBER. *)  
(* SET THE LIST OF CONSTITUENTS OF THE ENTITY TO BE READ IN THE *)  
(* FORWARD DIRECTION. *)
```

```

(*) COUNT THE NUMBER OF CONSTITUENTS IN THE LIST. *)
(*) FOR Y = 1 TO THE NUMBER OF CONSTITUENTS IN THE LIST *)
(*) READ A CONSTITUENT FROM THE LIST. *)
(*) GET THE CONSTITUENT'S ADB. *)
(*) CREATE A NEW PAGE, IF NECESSARY. *)
(*) WRITE OUT TO THE FILE A FIELD NAME. *)
(*) SET THE TYPE OF FIELD TO BE READ IN THE FORWARD DIRECTION. *)
(*) GET THE TYPE'S KEY. *)
(*) GET THE TYPE'S ADB. *)
(*) CASE TYPE_ADB.ENT_KIND OF *)
(*) INTEGER : WRITE OUT THE INTEGER DEFINITION *)
(*) REAL : WRITE OUT THE REAL DEFINITION *)
(*) STRING : WRITE OUT THE STRING DEFINITION *)
(*) LOGICAL : WRITE OUT THE LOGICAL DEFINITION *)
(*) ARRAY : WRITE OUT THE ARRAY DEFINITION *)
(*) DEFINED TYPE : WRITE OUT THE DEFINED TYPE DEFINITION *)
(*) POINTER : WRITE OUT THE POINTER DEFINITION *)
(*) WRITE OUT TO THE FILE 'END;'. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*) REVISED: 03/30/88 C. H. MOHME DBMA *)
(*) INCORPORATED THE FIELD POSITION NUMBER. *)
(*) *)
(*) REVISED: 09/28/87 C. H. MOHME DBMA *)
(*) INCORPORATED THE SUPERTYPE DATA TYPE. *)
(*) *)
(*) REVISED: 08/13/87 C. H. MOHME DBMA *)
(*) INCORPORATED OPTIONAL FIELD CHARACTERISTIC IN THE CONCEPTUAL *)
(*) SCHEMA REPORT. *)
(*) *)
(*) REVISED: 06/24/87 C. H. MOHME DBMA *)
(*) CHANGED TO PUT INTO BATCH INPUT FORMAT. *)
(*) *)
(*) ORIGINATED: 10/28/86 C. H. MOHME DBMA *)
(*) *)
(*)-----*)
(*) *)
(*)END-----*)
(*) END %INCLUDE CSENTWRT *)

```

(* %INCLUDE CSGBLHDG *)

(**)

```
PROCEDURE CSGBLHDG(VAR CSRFILE      : TEXT;
                   VAR PAGE_NUMBER : INTEGER;
                   VAR LINE_COUNT  : INTEGER);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*)
(*) $FUNCTION: (*)
(*) THIS ROUTINE WRITES OUT A GLOBAL FIELD HEADING ON A NEW (*)
(*) PAGE. (*)
(*) (*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== (*)
(*) CSRFILE I/O OUTPUT FILE (*)
(*) PAGE_NUMBER I/O CURRENT PAGE NUMBER (*)
(*) LINE_COUNT 0 LINE COUNT ON THE CURRENT PAGE (*)
(*) (*)
(*) $COMMONS: (*)
(*) NONE (*)
(*) (*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*) (*)
(*) $EXECUTION PROCEDURE: (*)
(*) INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT (*)
(*) (*)
(*) $PROCESSING DESCRIPTION: (*)
(*) (*)
(*) CREATE A NEW PAGE. (*)
(*) WRITE OUT TO THE FILE THE GLOBAL FIELD DEFINITION HEADING. (*)
(*) INCREMENT THE LINE COUNTER. (*)
(*) (*)
(*) $COMMENTS: (*)
(*) (*)
(*) $CHANGE CONTROL: (*)
(*) (*)
```

```
(* %INCLUDE CSGBLWRT *)
(**)
PROCEDURE CSGBLWRT(VAR IRC           : RET_REC;
                   VAR GLOBAL_FIELD_LIST : LISTKEY;
                   VAR CSRFILE          : TEXT;
                   VAR PAGE_NUMBER      : INTEGER);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE WRITES OUT GLOBAL FIELD DEFINITIONS TO A FILE
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* IRC 0 INTERNAL RETURN CODE
(* GLOBAL_FIELD_LIST I LIST OF GLOBAL FIELDS
(* CSRFILE I/O THE OUTPUT FILE
(* PAGE_NUMBER I/O THE CURRENT PAGE NUMBER
(* PAGE_MARK I/O ARRAY OF BOOLEAN: TRUE IF PAGE MARKS
(* THE BEGINNING OF A NEW ENTITY, FALSE
(* OTHERWISE.
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(* SET THE LIST OF GLOBAL FIELDS TO BE READ IN THE FORWARD
(* DIRECTION.
(* COUNT THE NUMBER OF GLOBAL FIELDS IN THE LIST.
(* CREATE A LIST FOR THE ALPHABETIZED GLOBAL FIELDS.
(* FOR X = 1 TO THE NUMBER OF GLOBAL FIELDS IN THE LIST
(* READ A GLOBAL FIELD FROM THE LIST OF GLOBAL FIELDS.
(* SET THE CONSTITUENT OF THE GLOBAL FIELD TO BE READ IN THE
(* FORWARD DIRECTION.
(* READ A FIELD KEY FROM THE CONSTITUENT LIST OF THE GLOBAL FIELD.
(* ATTACH THE FIELD KEY TO THE LIST FOR THE ALPHABETIZED GLOBAL
```

```
(*      FIELDS. *)
(*) SORT THE LIST OF GLOBAL FIELDS INTO ALPHABETICAL ORDER. *)
(*) SET THE LIST OF GLOBAL FIELDS TO BE READ IN THE FORWARD *)
(*) DIRECTION. *)
(*) WRITE OUT TO THE FILE THE GLOBAL FIELD DEFINITION HEADING. *)
(*) FOR X = 1 TO THE NUMBER OF GLOBAL FIELDS *)
(*)   READ A GLOBAL FIELD FROM THE LIST OF GLOBAL FIELDS. *)
(*)   GET THE FIELD'S ADB (THE GLOBAL FIELD'S CONSTITUENT). *)
(*)   CREATE A NEW PAGE, IF NECESSARY. *)
(*)   WRITE OUT TO THE FILE THE GLOBAL FIELD NAME. *)
(*)   SET THE TYPE OF THE FIELD TO BE READ IN THE FORWARD DIRECTION. *)
(*)   GET THE TYPE KEY. *)
(*)   GET THE TYPE'S ADB. *)
(*)   CASE TYPE_ADB.ENT_KIND OF *)
(*)     INTEGER      : WRITE OUT THE INTEGER DEFINITION *)
(*)     REAL         : WRITE OUT THE REAL DEFINITION *)
(*)     STRING       : WRITE OUT THE STRING DEFINITION *)
(*)     LOGICAL      : WRITE OUT THE LOGICAL DEFINITION *)
(*)     ARRAY        : WRITE OUT THE ARRAY DEFINITION *)
(*)     DEFINED TYPE : WRITE OUT THE DEFINED TYPE DEFINITION *)
(*)     POINTER      : WRITE OUT THE POINTER DEFINITION *)
(*)   WRITE OUT TO THE FILE A BLANK LINE. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
```

(* %INCLUDE CSHDGWRT *)

(**)

```
PROCEDURE CSHDGWRT(VAR CSRFILE      : TEXT;
                   VAR PAGE_NUMBER  : INTEGER;
                   VAR LINE_COUNT   : INTEGER;
                   VAR HEADING      : HEADING_TYPE;
                   VAR PAGE_TYPE    : PAGES);
```

SUBPROGRAM;

(**)

```
-----*)
*)
*)
*) $FUNCTION: *)
*) THIS ROUTINE CALLS THE APPROPRIATE HEADING ROUTINE. *)
*) *)
*) *)
*) $DESCRIPTION OF ARGUMENTS: *)
*) NAME I/O DESCRIPTION *)
*) ==== == ===== *)
*) CSRFILE I/O OUTPUT FILE *)
*) PAGE_NUMBER I/O CURRENT PAGE NUMBER *)
*) LINE_COUNT 0 LINE COUNT ON THE CURRENT PAGE *)
*) HEADING I DEFINITION OR INDEX HEADING *)
*) PAGE_TYPE I TYPE OF HEADING TO PRINT *)
*) *)
*) $COMMONS: *)
*) NONE *)
*) *)
*) $ENVIRONMENT: *)
*) LANGUAGE: IBM PASCAL *)
*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
*) *)
*) $EXECUTION PROCEDURE: *)
*) INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
*) *)
*) $PROCESSING DESCRIPTION: *)
*) *)
*) CASE TYPE OF PAGE OF *)
*) ENTITY PAGE : WRITE OUT TO THE FILE THE ENTITY HEADING. *)
*) CLASS PAGE : WRITE OUT TO THE FILE THE CLASS HEADING. *)
*) DEFINED TYPE PAGE : WRITE OUT TO THE FILE THE DEFINED TYPE *)
*) HEADING. *)
*) SUBSCHEMA PAGE : WRITE OUT TO THE FILE THE SUBSCHEMA *)
*) HEADING. *)
*) *)
*) $COMMENTS: *)
*) *)
*) $CHANGE CONTROL: *)
```


(* %INCLUDE CSINDWRT *)

(**)

```
PROCEDURE CSINDWRT(VAR IRC          : RET_REC;
                   VAR INDEX_TYPE    : PAGES;
                   VAR LIST           : LISTKEY;
                   VAR CSRFILE        : TEXT;
                   VAR PAGE_NUMBER    : INTEGER;
                   VAR CURRENT_PAGE   : PAGE_PTR);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) THIS ROUTINE WRITES OUT TO A FILE AN INDEX FOR AN ENTITY, (*)
(*) CLASS, OR SUBSCHEMA. (*)
(*) (*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC 0 INTERNAL RETURN CODE (*)
(*) INDEX_TYPE I TYPE OF INDEX TO BE PRINTED (*)
(*) LIST I ALPHABETIZED LIST OF ENTITIES (*)
(*) CSRFILE I/O THE OUTPUT FILE (*)
(*) PAGE_NUMBER I/O THE CURRENT PAGE NUMBER (*)
(*) CURRENT_PAGE I/O POINTS TO A PAGE RECORD WHICH CONTAINS (*)
(*) THE PAGE NUMBER IN THE REPORT FOR THE (*)
(*) ENTITY. (*)
(*) (*)
(*) $COMMONS: (*)
(*) NONE (*)
(*) (*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*) (*)
(*) $EXECUTION PROCEDURE: (*)
(*) INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT (*)
(*) (*)
(*) $PROCESSING DESCRIPTION: (*)
(*) (*)
(*) SET THE LIST OF ENTITIES TO BE READ IN THE FORWARD DIRECTION. (*)
(*) COUNT THE NUMBER OF ENTITIES IN THE LIST. (*)
(*) WRITE OUT THE PROPER PAGE HEADING. (*)
(*) FOR X = 1 TO THE NUMBER OF ENTITIES (*)
(*) WRITE OUT THE PROPER PAGE HEADING, IF NEW PAGE IS NEEDED. (*)
(*) READ AN ENTITY FROM THE LIST OF ENTITIES. (*)
```

CI PS560240032U
April 1990

```
(*  GET THE ENTITY'S ADB.
(*  WRITE OUT TO THE FILE THE ENTITY NAME (AND NUMBER).
(*  WRITE OUT TO THE FILE THE PAGE NUMBER THAT THE ENTITY BEGINS
(*  ON.
(*  INCREMENT THE LINE COUNT.
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*
```

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

```
(* %INCLUDE CSINTWRT *)
(**)
PROCEDURE CSINTWRT(VAR IRC          : RET_REC;
                   VAR INT_KEY      : ENTKEY;
                   VAR CSRFILE      : TEXT;
                   VAR LINE_COUNT   : INTEGER);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE WRITES OUT AN INTEGER DEFINITION
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC           0    INTERNAL RETURN CODE
(*     INT_KEY       I    INTEGER KEY
(*     CSRFILE       I/O  THE OUTPUT FILE
(*     LINE_COUNT    I/O  CURRENT LINE COUNT
(*
(* $COMMONS:
(*     NONE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(*     GET THE INTEGER'S ADB.
(*     WRITE OUT TO THE FILE THE INTEGER DEFINITION.
(*     INCREMENT THE LINE COUNTER.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*)
```

(* %INCLUDE CSLOGWRT *)

(**)

PROCEDURE CSLOGWRT(VAR IRC : RET_REC;
VAR CSRFILE : TEXT;
VAR LINE_COUNT : INTEGER);

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) THIS ROUTINE WRITES OUT A LOGICAL DEFINITION (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC 0 RETURN RECORD (*)
(*) CSRFILE I/O THE OUTPUT FILE (*)
(*) LINE_COUNT I/O CURRENT LINE COUNT (*)
(*)
(*) $COMMONS: (*)
(*) NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*) INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)
(*) WRITE OUT TO THE FILE 'LOGICAL;'. (*)
(*) INCREMENT THE LINE COUNTER. (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)
```

```
(* %INCLUDE CSMAIN *)
(**)
PROCEDURE CSMAIN(VAR IRC      : RET_REC;
                  VAR MSG      : MESSAGE);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE SERVES AS THE MAIN DRIVER FOR THE CONCEPTUAL
(* SCHEMA REPORT.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* IRC 0 INTERNAL RETURN CODE
(* MSG 0 MESSAGE RETURNED INDICATING IF A REPORT
(* HAS BEEN PRODUCED
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(* INITIALIZE THE VARIABLES USED WITHIN THIS ROUTINE.
(* WRITE OUT TO THE FILE THE REPORT COVER.
(* INITIALIZE THE PAGE NUMBER AND PAGE CHAIN.
(* MAKE A LIST OF DEFINED TYPES WITHIN THE SCHEMA.
(* ALPHABETIZE THE LIST OF DEFINED TYPES.
(* WRITE OUT TO THE FILE THE DEFINED TYPES WITHIN THE SCHEMA.
(* DELETE THE LIST OF DEFINED TYPES WITHIN THE SCHEMA.
(* MAKE A LIST OF GLOBAL FIELDS WITHIN THE SCHEMA.
(* WRITE OUT TO THE FILE THE GLOBAL FIELDS WITHIN THE SCHEMA.
(* DELETE THE LIST OF GLOBAL FIELDS WITHIN THE SCHEMA.
(* MAKE A LIST OF ENTITIES WITHIN THE SCHEMA.
(* ALPHABETIZE THE LIST OF ENTITIES.
(* WRITE OUT TO THE FILE THE ENTITIES WITHIN THE SCHEMA.
(* MAKE A LIST OF CLASSES WITHIN THE SCHEMA.
(* ALPHABETIZE THE LIST OF CLASSES.
```

```
(*  WRITE OUT TO THE FILE THE CLASSES WITHIN THE SCHEMA.          *)
(*  MAKE A LIST OF SUBSCHEMAS WITHIN THE SCHEMA.                   *)
(*  ALPHABETIZE THE LIST OF SUBSCHEMAS.                             *)
(*  WRITE OUT TO THE FILE THE SUBSCHEMAS WITHIN THE SCHEMA.       *)
(*  WRITE OUT TO THE FILE THE ENTITY INDEX.                         *)
(*  DELETE THE LIST OF ENTITIES.                                    *)
(*  WRITE OUT TO THE FILE THE CLASS INDEX.                          *)
(*  DELETE THE LIST OF CLASSES.                                     *)
(*  WRITE OUT TO THE FILE THE SUBSCHEMA INDEX.                     *)
(*  DELETE THE LIST OF SUBSCHEMAS.                                  *)
(*  IF NO MODEL EXISTS, WRITE APPROPRIATE MESSAGE.                 *)
(*  DISPOSE OF POINTERS USED IN THIS ROUTINE.                      *)
(*                                                                    *)
(*  $COMMENTS:                                                       *)
(*                                                                    *)
(*    WITHIN THE INCLUDE FILE 'SCECON', ONE CAN SET THE MAXIMUM    *)
(*    NUMBER OF LINES PER PAGE IN THE REPORT.                      *)
(*                                                                    *)
(*  $CHANGE CONTROL:                                                 *)
(*                                                                    *)
```

```

(*) %INCLUDE CSNEWPG *)
(**)
  PROCEDURE CSNEWPG(VAR CSRFILE      : TEXT;
                    VAR PAGE_NUMBER : INTEGER;
                    VAR LINE_COUNT  : INTEGER);

    SUBPROGRAM;

(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS ROUTINE CREATES A NEW PAGE IN THE CONCEPTUAL SCHEMA *)
(*)   REPORT. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O  DESCRIPTION *)
(*)   ====      ==  ===== *)
(*)   CSRFILE    I/O  OUTPUT FILE *)
(*)   PAGE_NUMBER I/O  CURRENT PAGE NUMBER *)
(*)   LINE_COUNT  O   LINE COUNT ON THE CURRENT PAGE *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) *)
(*)   INCREMENT THE PAGE NUMBER. *)
(*)   CREATE A NEW PAGE IN THE FILE. *)
(*)   WRITE OUT TO THE FILE THE CURRENT PAGE NUMBER. *)
(*)   INCREMENT THE LINE COUNTER. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```

(* %INCLUDE CSPTRWRT *)

(**)

```
PROCEDURE CSPTRWRT(VAR IRC          : RET_REC;
                   VAR POINTER_KEY : ENIKEY;
                   VAR CSRFILE     : TEXT;
                   VAR PAGE_NUMBER : INTEGER;
                   VAR LINE_COUNT  : INTEGER;
                   VAR INDENT      : INTEGER;
                   VAR PAGE_TYPE   : PAGES);
```

SUBPROGRAM;

(**)

```
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE WRITES OUT A POINTER DEFINITION *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 INTERNAL RETURN CODE *)
(* POINTER_KEY I POINTER KEY *)
(* CSRFILE I/O OUTPUT FILE *)
(* PAGE_NUMBER I/O CURRENT PAGE NUMBER *)
(* LINE_COUNT I/O CURRENT LINE COUNT *)
(* INDENT I/O NUMBER OF SPACES TO INDENT *)
(* PAGE_TYPE I/O TYPE OF REPORT PAGE *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* GET THE POINTER'S ADB. *)
(* WRITE OUT TO THE FILE 'POINTER TO ('. *)
(* CREATE A NEW PAGE, IF NECESSARY. *)
(* SET THE LIST OF CONSTITUENTS OF THE POINTER TO BE READ IN THE *)
(* FORWARD DIRECTION. *)
(* COUNT THE NUMBER OF CONSTITUENTS IN THE LIST. *)
(* FOR Y = 1 TO THE NUMBER OF CONSTITUENTS IN THE LIST *)
(* *)
```


CI PS560240032U
April 1990

```
(*      CREATE A NEW PAGE, IF NECESSARY.                *)
(*      READ A CONSTITUENT FROM THE LIST OF CONSTITUENTS. *)
(*      GET THE CONSTITUENT'S ADB.                        *)
(*      WRITE OUT TO THE FILE A CONSTITUENT (ENTITY OR CLASS) *)
(*      INCREMENT THE LINE COUNTER.                      *)
(*      WRITE OUT TO THE FILE ')'.                        *)
(*                                                        *)
(*      $COMMENTS:                                        *)
(*                                                        *)
(*      $CHANGE CONTROL:                                  *)
(*                                                        *)
```

```
(* %INCLUDE CSRELWRT *)
(**)
  PROCEDURE CSRELWRT(VAR IRC          : RET_REC;
                    VAR REAL_KEY      : ENTKEY;
                    VAR CSRFILE       : TEXT;
                    VAR LINE_COUNT    : INTEGER);

  SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE WRITES OUT A REAL DEFINITION
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0   INTERNAL RETURN CODE
(*   REAL_KEY      I   REAL KEY
(*   CSRFILE       I/O  THE OUTPUT FILE
(*   LINE_COUNT    I/O  CURRENT LINE COUNT
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(* GET THE INTEGER'S ADB.
(* WRITE OUT TO THE FILE THE REAL DEFINITION.
(* INCREMENT THE LINE COUNTER.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE CSRMAIN *)
(**)
  PROCEDURE CSRMAIN(VAR IRC      : RET_REC;
                    VAR MSG      : MESSAGE);
    SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    THIS ROUTINE PROVIDES THE CAPABILITIES TO GENERATE, BROWSE,
(*    AND PRINT THE CONCEPTUAL SCHEMA REPORT.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME      I/O  DESCRIPTION
(*    ====      ==  =====
(*    IRC        0   RETURN CODE
(*    MSG        I/O  PANEL MESSAGE
(*
(*  $COMMONS:
(*    NONE
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(*  $PROCESSING DESCRIPTION:
(*
(*    DISPLAY THE CONCEPTUAL SCHEMA REPORT MAIN MENU
(*    DETERMINE WHICH OPTION WAS CHOSEN AND ACT ACCORDINGLY:
(*      -- GENERATE THE CONCEPTUAL SCHEMA REPORT
(*      -- BROWSE THE CONCEPTUAL SCHEMA REPORT
(*      -- PRINT THE CONCEPTUAL SCHEMA REPORT
(*      -- RETURN TO THE MAIN MENU
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*
```

```

(** %INCLUDE CSRMENU *)
(**)
PROCEDURE CSRMENU(VAR MESS      : MESSAGE;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   DISPLAYS THE CONCEPTUAL SCHEMA REPORT MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   INITIALIZE THE VARIABLES
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
(*   DISPLAY THE CONCEPTUAL SCHEMA REPORT MENU
(*   DETERMINE THE ACTION SELECTED FROM THE MENU
(*   REMOVE THE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*

```

```
(* %INCLUDE CSRPTCVR *)
(**)
  PROCEDURE CSRPTCVR(VAR CSRFILE : TEXT);

    SUBPROGRAM;
  (**)
  (*-----*)
  (*
  (* $FUNCTION:
  (*   THIS ROUTINE PRINTS OUT THE REPORT COVER FOR THE CONCEPTUAL
  (*   SCHEMA.
  (*
  (*
  (* $DESCRIPTION OF ARGUMENTS:
  (*   NAME          I/O  DESCRIPTION
  (*   ====          ==
  (*   CSRFILE       I/O  OUTPUT FILE
  (*
  (* $COMMONS:
  (*   NONE
  (*
  (* $ENVIRONMENT:
  (*   LANGUAGE: IBM PASCAL
  (*   HARDWARE SYSTEM: IBM 360/370/4341/4381
  (*
  (* $EXECUTION PROCEDURE:
  (*   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
  (*
  (* $PROCESSING DESCRIPTION:
  (*
  (*   CREATE A NEW PAGE.
  (*   WRITE 'CONCEPTUAL SCHEMA REPORT'.
  (*
  (* $COMMENTS:
  (*
  (* $CHANGE CONTROL:
  (*
```

```
(* %INCLUDE CSSTGWRT *)
(**)
    PROCEDURE CSSTGWRT(VAR IRC          : RET_REC;
                      VAR STRING_KEY   : ENTKEY;
                      VAR CSRFILE      : TEXT;
                      VAR LINE_COUNT   : INTEGER);

    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*     THIS ROUTINE WRITES OUT A STRING DEFINITION *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* *)
(*     NAME          I/O  DESCRIPTION *)
(*     ====          ===  ===== *)
(*     IRC           O    INTERNAL RETURN CODE *)
(*     STRING_KEY    I    STRING KEY *)
(*     CSRFILE       I/O  THE OUTPUT FILE *)
(*     LINE_COUNT    I/O  CURRENT LINE COUNT *)
(* *)
(* $COMMONS: *)
(*     NONE *)
(* *)
(* $ENVIRONMENT: *)
(*     LANGUAGE: IBM PASCAL *)
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*     INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(*     GET THE STRING'S ADR. *)
(*     WRITE OUT TO THE FILE THE STRING DEFINITION. *)
(*     INCREMENT THE LINE COUNTER. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

(* %INCLUDE GSSTRWRT *)

(**)

```
PROCEDURE GSSTRWRT(VAR IRC          : RET_REC;
                   VAR STRUCTURE_KEY : ENTKEY;
                   VAR CSRFILE       : TEXT;
                   VAR PAGE_NUMBER   : INTEGER;
                   VAR LINE_COUNT    : INTEGER;
                   VAR PAGE_TYPE     : PAGES);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) THIS ROUTINE WRITES OUT A STUCTURE DEFINITION (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC 0 INTERNAL RETURN CODE (*)
(*) STRUCTURE_KEY I STRUCTURE KEY (*)
(*) CSRFILE I/O THE OUTPUT FILE (*)
(*) PAGE_NUMBER I/O CURRENT PAGE NUMBER (*)
(*) LINE_COUNT I/O CURRENT LINE COUNT (*)
(*) PAGE_TYPE I/O TYPE OF REPORT PAGE (*)
(*)
(*) $COMMONS: (*)
(*) NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*) INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)
(*) GET THE STRUCTURE'S ADB. (*)
(*) WRITE OUT TO THE FILE 'STRUCTURE'. (*)
(*) INCREMENT THE LINE COUNTER. (*)
(*) SET THE LIST OF CONSTITUENTS TO BE READ IN THE FORWARD DIRECTION. (*)
(*) COUNT THE NUMBER OF CONSTITUENTS IN THE LIST. (*)
(*) FOR X = 1 TO THE NUMBER OF CONSTITUENTS IN THE LIST (*)
(*) READ A CONSTITUENT FROM THE LIST. (*)
(*) GET THE CONSTITUENT'S ADB. (*)
(*) CREATE A NEW PAGE, IF NECESSARY. (*)
```

```
(*      WRITE OUT TO THE FILE THE FIELD NAME.                      *)
(*      SET THE TYPE LIST TO BE READ IN THE FORWARD DIRECTION.      *)
(*      GET THE TYPE'S KEY.                                           *)
(*      GET THE TYPE'S ADB.                                           *)
(*      CASE TYPE_ADB.ENT_KIND OF                                     *)
(*          INTEGER          : WRITE OUT THE INTEGER DEFINITION.      *)
(*          REAL             : WRITE OUT THE REAL DEFINITION.         *)
(*          STRING           : WRITE OUT THE STRING DEFINITION.       *)
(*          LOGICAL          : WRITE OUT THE LOGICAL DEFINITION.      *)
(*          ARRAY            : WRITE OUT THE ARRAY DEFINITION.        *)
(*          DEFINED TYPE     : WRITE OUT THE DEFINED TYPE DEFINITION.  *)
(*          POINTER          : WRITE OUT THE POINTER DEFINITION.      *)
(*      CREATE A NEW PAGE, IF NECESSARY.                              *)
(*      WRITE OUT TO THE FILE 'END;'.                                  *)
(*                                                                    *)
(*      $COMMENTS:                                                    *)
(*                                                                    *)
(*      $CHANGE CONTROL:                                              *)
(*                                                                    *)
```


(* %INCLUDE CSSUBHDG *)

(**)

```
PROCEDURE CSSUBHDG(VAR CSRFILE      : TEXT;
                   VAR HEADING      : HEADING_TYPE;
                   VAR PAGE_NUMBER  : INTEGER;
                   VAR LINE_COUNT   : INTEGER);
```

SUBPROGRAM;

(**)

```
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE WRITES OUT A SUBSCHEMA HEADING ON A NEW PAGE. *)
(* *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* CSRFILE I/O OUTPUT FILE *)
(* HEADING I DEFINITION OR INDEX HEADING *)
(* PAGE_NUMBER I/O CURRENT PAGE NUMBER *)
(* LINE_COUNT O LINE COUNT ON THE CURRENT PAGE *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* CREATE A NEW PAGE. *)
(* WRITE OUT TO THE FILE THE APPROPRIATE HEADING (DEFINITION OR *)
(* INDEX). *)
(* INCREMENT THE LINE COUNTER. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE CSSUBWRT *)
(**)
  PROCEDURE CSSUBWRT(VAR IRC          : RET_REG;
                    VAR SUBSCHEMA_LIST : LISTKEY;
                    VAR CSRFILE        : TEXT;
                    VAR PAGE_NUMBER    : INTEGER;
                    VAR CURRENT_PAGE   : PAGE_PTR);

    SUBPROGRAM;
(**)
(*-----*)
(*
(*
(* $FUNCTION:
(*   THIS ROUTINE WRITES OUT SUBSCHEMA DEFINITIONS TO A FILE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   SUBSCHEMA_LIST I    ALPHABETIZED LIST OF CLASSES
(*   CSRFILE       I/O  THE OUTPUT FILE
(*   PAGE_NUMBER   I/O  THE CURRENT PAGE NUMBER
(*   CURRENT_PAGE  I/O  POINTS TO THE CURRENT PAGE RECORD IN
(*                       THE CHAIN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(* SET THE LIST OF SUBSCHEMAS TO BE READ IN THE FORWARD DIRECTION.
(* COUNT THE NUMBER OF SUBSCHEMAS IN THE LIST.
(* FOR X = 1 TO THE NUMBER OF SUBSCHEMAS IN THE LIST
(*   WRITE OUT TO THE FILE THE SUBSCHEMA HEADING.
(*   UPDATE THE PAGE RECORD CHAIN.
(*   READ A SUBSCHEMA FROM THE LIST OF SUBSCHEMAS.
(*   GET THE SUBSCHEMA'S ADB.
(*   WRITE OUT TO THE FILE THE SUBSCHEMA NAME.
(*   SET THE LIST OF CONSTITUENTS TO BE READ IN THE FORWARD
(*   DIRECTION.
(*)
```

```
(*      COUNT THE NUMBER OF CONSTITUENTS IN THE LIST.      *)
(*      FOR Y = 1 TO THE NUMBER OF CONSTITUENTS IN THE LIST  *)
(*      READ A CONSTITUENT FROM THE LIST OF CONSTITUENTS     *)
(*      GET THE CONSTITUENT'S ADB.                            *)
(*      WRITE OUT TO THE FILE THE HEADING ON A NEW PAGE, IF   *)
(*      NECESSARY.                                             *)
(*      WRITE OUT TO THE FILE THE CONSTITUENT (ENTITY OR CLASS). *)
(*      WRITE OUT TO THE FILE 'END;'.                          *)
(*                                                            *)
(* $COMMENTS:                                                 *)
(*                                                            *)
(* $CHANGE CONTROL:                                           *)
(*)
```

```
(* %INCLUDE CSSUPHDG *)
(**)
  PROCEDURE CSSUPHDG(VAR CSRFILE      : TEXT;
                     VAR HEADING     : HEADING_TYPE;
                     VAR PAGE_NUMBER : INTEGER;
                     VAR LINE_COUNT  : INTEGER);

    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE WRITES OUT A SUPERTYPE HEADING ON A NEW PAGE. *)
(* *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   CSRFILE       I/O  OUTPUT FILE *)
(*   HEADING        I   DEFINITION OR INDEX HEADING *)
(*   PAGE_NUMBER    I/O  CURRENT PAGE NUMBER *)
(*   LINE_COUNT     O   LINE COUNT ON THE CURRENT PAGE *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(*   CREATE A NEW PAGE. *)
(*   WRITE OUT TO THE FILE THE APPROPRIATE SUPERTYPE HEADING *)
(*   DEFINITION OR INDEX. *)
(*   INCREMENT THE LINE COUNTER. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(*) %INCLUDE CSSUPWRT *)
(**)
PROCEDURE CSSUPWRT(VAR IRC          : RET_REC;
                   VAR SUPERTYPE_LIST : LISTKEY;
                   VAR CSRFILE        : TEXT;
                   VAR PAGE_NUMBER    : INTEGER;
                   VAR CURRENT_PAGE   : PAGE_PTR);

SUBPROGRAM;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION: *)
(*) THIS ROUTINE WRITES OUT SUPERTYPE DEFINITIONS TO A FILE *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) IRC 0 INTERNAL RETURN CODE *)
(*) SUPERTYPE_LIST I ALPHABETIZED LIST OF SUPERTYPES *)
(*) CSRFILE I/O THE OUTPUT FILE *)
(*) PAGE_NUMBER I/O THE CURRENT PAGE NUMBER *)
(*) CURRENT_PAGE I/O POINTS TO THE CURRENT PAGE RECORD IN *)
(*) THE CHAIN. *)
(*) *)
(*) $COMMONS: *)
(*) NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) *)
(*) SET THE LIST OF ENTITIES TO BE READ IN THE FORWARD DIRECTION. *)
(*) COUNT THE NUMBER OF ENTITIES IN THE LIST. *)
(*) FOR X = 1 TO THE NUMBER OF ENTITIES IN THE LIST *)
(*) WRITE OUT TO THE FILE THE ENTITY HEADING. *)
(*) UPDATE THE PAGE RECORD CHAIN. *)
(*) READ AN ENTITY FROM THE LIST OF ENTITIES. *)
(*) GET THE ENTITY'S ADB. *)
(*) WRITE OUT TO THE FILE THE ENTITY NAME AND NUMBER. *)
(*) SET THE LIST OF CONSTITUENTS OF THE ENTITY TO BE READ IN THE

```

```
(*      FORWARD DIRECTION.                                *)
(*) COUNT THE NUMBER OF CONSTITUENTS IN THE LIST.          *)
(*) FOR Y = 1 TO THE NUMBER OF CONSTITUENTS IN THE LIST    *)
(*)   READ A CONSTITUENT FROM THE LIST.                     *)
(*)   GET THE CONSTITUENT'S ADB.                             *)
(*)   CREATE A NEW PAGE, IF NECESSARY.                       *)
(*)   WRITE OUT TO THE FILE A FIELD NAME.                   *)
(*)   SET THE TYPE OF FIELD TO BE READ IN THE FORWARD DIRECTION. *)
(*)   GET THE TYPE'S KEY.                                    *)
(*)   GET THE TYPE'S ADB.                                    *)
(*)   CASE TYPE_ADB.ENT_KIND OF                             *)
(*)     INTEGER      : WRITE OUT THE INTEGER DEFINITION    *)
(*)     REAL         : WRITE OUT THE REAL DEFINITION       *)
(*)     STRING       : WRITE OUT THE STRING DEFINITION     *)
(*)     LOGICAL      : WRITE OUT THE LOGICAL DEFINITION    *)
(*)     ARRAY        : WRITE OUT THE ARRAY DEFINITION      *)
(*)     DEFINED TYPE : WRITE OUT THE DEFINED TYPE DEFINITION *)
(*)     POINTER      : WRITE OUT THE POINTER DEFINITION    *)
(*)   WRITE OUT TO THE FILE 'END;'.                          *)
(*)                                                         *)
(*) $COMMENTS:                                              *)
(*)                                                         *)
(*) $CHANGE CONTROL:                                       *)
(*)                                                         *)
```

```
(* %INCLUDE CSTYPWRT *)
(**)
  PROCEDURE CSTYPWRT(VAR IRC          : RET_REC;
                    VAR DEF_TYP_LIST : LISTKEY;
                    VAR CSRFILE      : TEXT;
                    VAR PAGE_NUMBER  : INTEGER);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE WRITES OUT THE DEFINED TYPE DEFINITIONS TO
(*   A FILE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   DEF_TYP_LIST  I    ALPHABETIZED LIST OF DEFINED TYPES
(*   CSRFILE       I/O  THE OUTPUT FILE
(*   PAGE_NUMBER   I/O  THE CURRENT PAGE NUMBER
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE CONCEPTUAL SCHEMA REPORT
(*
(* $PROCESSING DESCRIPTION:
(*
(* SET THE LIST OF DEFINED TYPES TO BE READ IN THE FORWARD
(*   DIRECTION.
(* COUNT THE NUMBER OF DEFINED TYPES IN THE LIST.
(* WRITE OUT TO THE FILE THE DEFINED TYPE DEFINITION HEADING.
(* FOR X = 1 TO THE NUMBER OF DEFINED TYPES IN THE LIST
(*   READ A DEFINED TYPE FROM THE LIST.
(*   GET THE DEFINED TYPE'S ADB.
(*   WRITE OUT THE DEFINED TYPE HEADING, IF NECESSARY.
(*   WRITE OUT TO THE FILE THE DEFINED TYPE NAME.
(*   SET THE LIST OF CONSTITUENTS TO BE READ IN THE FORWARD
(*   DIRECTION.
(*   READ THE CONSTITUENT FROM THE LIST OF CONSTITUENTS.
```

```
(* GET THE CONSTITUENT'S ADB. *)
(* CASE CONSTITUENT_ADB.ENT_KIND OF *)
(* INTEGER : WRITE OUT THE INTEGER DEFINITION. *)
(* REAL : WRITE OUT THE REAL DEFINITION. *)
(* STRING : WRITE OUT THE STRING DEFINITION. *)
(* LOGICAL : WRITE OUT THE LOGICAL DEFINITION. *)
(* ARRAY : WRITE OUT THE ARRAY DEFINITION. *)
(* DEFINED TYPE : WRITE OUT THE DEFINED TYPE DEFINITION. *)
(* POINTER : WRITE OUT THE POINTER DEFINITION. *)
(* ENUMERATION : WRITE OUT THE ENUMERATION DEFINITION. *)
(* STRUCTURE : WRITE OUT THE STRUCTURE DEFINITION. *)
(* INCREMENT THE LINE COUNTER. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```



```
(* BEGIN %INCLUDE DDABNDS *****)
(*)
PROCEDURE DDABNDS ( VAR   DDFILE      : T_FILE_VARIANT;
                   CONST NO_OF_DIMEN  : INTEGER;
                   CONST STARTING_ARRAY_POSITION : INTEGER;
                   VAR   POINTER      : T_VARIANT_POINTER;
                   VAR   ENTITY_SIZE   : INTEGER;
                   VAR   ENTITY_POSITION : INTEGER;
                   VAR   ENUM_INDEX    : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE LOW-BOUND AND UPPER-BOUND FOR THE ARRAY ATTRIBUTE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     DDFILE        O    DATA DICTIONARY SEQUENTIAL FILE
(*)     NO_OF_DIMEN    I    NUMBER OF ARRAY DIMENSION
(*)     STARTING_ARRAY_PO I  STARTING POSITION IN THE ARRAY TABLE
(*)     POINTER        I    POINTER TO ARRAY TABLE
(*)     ENTITY_SIZE    O    NUMBER OF RECORDS IN THE DEFINITION
(*)     ENTITY_POSITION O    FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH THE NUMBER OF DIMENSIONS
(*)     WRITE LOW-BOUND AND UPPER-BOUND
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 23 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDABNDS *****)
```

```

(* BEGIN %INCLUDE DDADB *****)
(*)
PROCEDURE DDADB ( VAR   DDFILE           : T_FILE_VARIANT;
                  CONST RUNTIME          : T_RUN_TIME;
                  CONST ENTRY            : INTEGER;
                  CONST PS_ORDER         : T_PS_ORDER;
                  VAR   ENTITY_SIZE      : INTEGER;
                  VAR   ENTITY_POSITION  : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   WRITE THE BASIC RECORD OF AN ENTITY TO A SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ==  =====
(*)   DDFILE          I    DATA DICTIONARY SEQUENTIAL FILE
(*)   RUNTIME          I    CONTAINS THE ENTITY DEFINITION
(*)   ENTRY            I    ENTRY ORDER IN THE DEFINITION
(*)   PS_ORDER         I    LIST OF PHYSICAL SCHEMA ORDER
(*)   ENTITY_SIZE      0    NUMBER OF RECORDS IN THE DEFINITION
(*)   ENTITY_POSITION  0    FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   WRITE BASIC RECORD
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   ORIGINATED: 17 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDADB *****

```

```
(* BEGIN %INCLUDE DDARRAY *****)
(*)
PROCEDURE DDARRAY ( VAR   DDFILE      : T_FILE_VARIANT;
                    CONST RUNTIME     : T_RUN_TIME;
                    CONST ENTRY       : INTEGER;
                    CONST PS_ORDER    : T_PS_ORDER;
                    VAR   ENTITY_SIZE  : INTEGER;
                    VAR   ENTITY_POSITION : INTEGER;
                    VAR   ATT_SIZE     : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE ARRAY ATTRIBUTE OF AN ENTITY TO A SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   DDFILE        O    DATA DICTIONARY SEQUENTIAL FILE
(*)   RUNTIME        I    CONTAINS THE ENTITY DEFINITION
(*)   ENTRY          I    ENTRY ORDER IN THE DEFINITION
(*)   PS_ORDER       I    LIST OF PHYSICAL SCHEMA ORDER
(*)   ENTITY_SIZE    O    NUMBER OF RECORDS IN THE DEFINITION
(*)   ENTITY_POSITION O    FIRST RECORD OF THE DEFINITION
(*)   ATT_SIZE       O    SIZE OF ARRAY ATTRIBUTE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   OBTAIN THE NUMBER OF DIMENSIONS
(*)   OBTAIN THE STARTING POSITION OF ARRAY TABLE
(*)   CASE DATA TYPE OF
(*)     IN-ADB : WRITE BASIC DEFINITION
(*)             DDALIST ( EXTERNAL SUBPROGRAM FOR LOW-BOUND AND
(*)                       UPPER-BOUND )
(*)     IN-CL  : WRITE BASIC DEFINITION
(*)             DDCL ( EXTERNAL SUBPROGRAM FOR CONSTITUENT LIST)
(*)             DDALIST ( EXTERNAL SUBPROGRAM FOR LOW-BOUND AND
(*)                       UPPER-BOUND )
(*)   END CASE
(*)
```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* ORIGINATED: 17 MARCH 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE DDARRAY *****)
```

```

(* BEGIN %INCLUDE DDCL *****)
(*)
PROCEDURE DDCL ( VAR   DDFILE           : T_FILE_VARIANT;
                  CONST RUNTIME         : T_RUN_TIME;
                  CONST ENTRY           : INTEGER;
                  VAR   ENTITY_SIZE     : INTEGER;
                  VAR   ENTITY_POSITION : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   WRITE THE CONSTITUENT REFERENCES OF AN ENTITY TO A
(*)   SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O DESCRIPTION
(*)   ====           === =====
(*)   DDFILE         0   DATA DICTIONARY SEQUENTIAL FILE
(*)   RUNTIME        I   CONTAINS THE ENTITY DEFINITION
(*)   ENTRY          I   ENTRY ORDER IN THE DEFINITION
(*)   ENTITY_SIZE    0   NUMBER OF RECORDS IN THE DEFINITION
(*)   ENTITY_POSITION 0   FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   IF NOT ARRAY ATTRIBUTE THEN
(*)     WRITE BASIC DEFINITION
(*)   END IF
(*)   OBTAIN THE NUMBER OF ELIGIBLE KINDS
(*)   OBTAIN STARTING POSITION OF CONSTITUENT LIST TABLE
(*)   LOOP THROUGH THE NUMBER OF ELIGIBLE KINDS
(*)     WRITE ELIGIBLE KIND IN THE CONSTITUENT LIST TABLE
(*)   END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   ORIGINATED: 17 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDCL *****)

```

```

(* BEGIN %INCLUDE DDCLASS *****)
(*)
PROCEDURE DDCLASS ( CONST LIST_OF_CLASS : LISTKEY;
                    VAR  DDFILE          : T_FILE_VARIANT;
                    VAR  DDINX           : T_INX_FILE;
                    VAR  ENTITY_POSITION: INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE CLASS KINDS TO A SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O DESCRIPTION
(*)     ====          == =====
(*)     DDFILE        I   DATA DICTIONARY SEQUENTIAL FILE
(*)     RUNTIME        I   CONTAINS THE ENTITY DEFINITION
(*)     ENTRY          I   ENTRY ORDER IN THE DEFINITION
(*)     PS_ORDER       I   LIST OF PHYSICAL SCHEMA ORDER
(*)     ENTITY_SIZE    0   NUMBER OF RECORDS IN THE DEFINITION
(*)     ENTITY_POSITION 0   FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     WRITE BASIC RECORD
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 08 MAY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDCLASS *****)

```

```

(*) BEGIN %INCLUDE DENTITY *****
(*)
PROCEDURE DENTITY ( CONST LIST_OF_ENTITIES : LISTKEY;
                    VAR  DDFILE           : T_FILE_VARIANT;
                    VAR  DDINX            : T_INX_FILE;
                    VAR  ENTITY_POSITION: INTEGER;
                    VAR  IRC              : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION:
(*)     WRITE THE ENTITY DEFINITIONS TO A SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ===  =====
(*)     DDFILE          I    DATA DICTIONARY SEQUENTIAL FILE
(*)     RUNTIME          I    CONTAINS THE ENTITY DEFINITION
(*)     ENTRY            I    ENTRY ORDER IN THE DEFINITION
(*)     PS_ORDER         I    LIST OF PHYSICAL SCHEMA ORDER
(*)     ENTITY_SIZE      0    NUMBER OF RECORDS IN THE DEFINITION
(*)     ENTITY_POSITION  0    FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     WRITE ENTITY DEFINITION TO A FILE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 08 MAY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DENTITY *****

```

```
(* BEGIN %INCLUDE DDENUM *****)
PROCEDURE DDENUM ( VAR   DDFILE           : T_FILE_VARIANT;
                   CONST RUNTIME          : T_RUN_TIME;
                   CONST ENTRY            : INTEGER;
                   CONST PS_ORDER         : T_PS_ORDER;
                   VAR   ENTITY_SIZE      : INTEGER;
                   VAR   ENTITY_POSITION  : INTEGER;
                   CONST ENUM_TABLE_INDEX : INTEGER;
                   CONST NO_OF_ENUM      : INTEGER );
    SUBPROGRAM;

(* *)
(* $FUNCTION: *)
(* WRITE THE ENUMERATION ATTRIBUTE OF AN ENTITY TO A *)
(* SEQUENTIAL FILE *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME          I/O  DESCRIPTION *)
(* ====          ===  ===== *)
(* DDFILE         0    DATA DICTIONARY SEQUENTIAL FILE *)
(* RUNTIME        I    CONTAINS THE ENTITY DEFINITION *)
(* ENTRY          I    ENTRY ORDER IN THE DEFINITION *)
(* PS_ORDER       I    LIST OF PHYSICAL SCHEMA ORDER *)
(* ENTITY_SIZE    0    NUMBER OF RECORDS IN THE DEFINITION *)
(* ENTITY_POSITION 0    FIRST RECORD OF THE DEFINITION *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* WRITE BASIC DEFINITION *)
(* OBTAIN THE NUMBER OF ENUMERATION VALUES *)
(* OBTAIN THE STARTING POSITION OF ENUMERATION VALUE TABLE *)
(* LOOP THROUGH THE NUMBER OF ENUMERATION VALUES *)
(* WRITE THE ENUMERATION VALUE FROM THE TABLE *)
(* END LOOP *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* ORIGINATED: 17 MARCH 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE DDENUM *****)
```



```

(* BEGIN %INCLUDE DDREPORT *****
(*)
PROCEDURE DDREPORT ( VAR  SUBSCHEMA_KEY    : ENTKEY;
                     VAR  IRC              : RET_REC );
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     WRITE THE DATA DICTIONARY IN A CHARACTER FORM.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     SUBSCHEMA_KEY    I   INPUT VALUE OF ARBITRARY SIZE
(*)     IRC              0   RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     SORT ENTITIES BY KIND NUMBER
(*)     LOOP THROUGH THE LIST OF ENTITIES
(*)     REQUEST ENTITY DEFINITION
(*)     LOOP THROUGH THE NUMBER OF ATTRIBUTES IN THE DEFINITION
(*)     CASE DATA TYPE OF
(*)         IN-ADB      : DDADB ( EXTERNAL SUBPROGRAM )
(*)         IN-ENUM     : DDENUM ( EXTERNAL SUBPROGRAM )
(*)         IN-CL       : DDCL ( EXTERNAL SUBPROGRAM )
(*)         IN-ARRAY    : DDARRAY ( EXTERNAL SUBPROGRAM )
(*)     END CASE
(*)     END LOOP
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 23 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDREPORT *****

```

```

(*) BEGIN %INCLUDE DDSTRUC *****
(*)
PROCEDURE DDSTRUC ( VAR DDFILE          : T_FILE_VARIANT;
                   CONST RUNTIME        : T_RUN_TIME;
                   VAR ENTRY            : INTEGER;
                   CONST PS_ORDER       : T_PS_ORDER;
                   VAR ENTITY_SIZE      : INTEGER;
                   VAR ENTITY_POSITION  : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE STRUCTURE ATTRIBUTE OF AN ENTITY TO A
(*)     SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     DDFILE        I    DATA DICTIONARY SEQUENTIAL FILE
(*)     RUNTIME        I    CONTAINS THE ENTITY DEFINITION
(*)     ENTRY          I    ENTRY ORDER IN THE DEFINITION
(*)     PS_ORDER       I    LIST OF PHYSICAL SCHEMA ORDER
(*)     ENTITY_SIZE    0    NUMBER OF RECORDS IN THE DEFINITION
(*)     ENTITY_POSITION 0    FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 15 JANUARY 1988, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDSTRUC *****

```

```

(*) BEGIN %INCLUDE DDWRITE *****
(*)
PROCEDURE DDWRITE ( CONST RUNTIME      : T_RUN_TIME;
                    VAR  DDFILE        : T_FILE_VARIANT;
                    VAR  DDINX         : T_INX_FILE;
                    VAR  ENTITY_POSITION: INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE ENTITY DEFINITIONS TO A SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     DDFILE        I    DATA DICTIONARY SEQUENTIAL FILE
(*)     RUNTIME        I    CONTAINS THE ENTITY DEFINITION
(*)     ENTRY          I    ENTRY ORDER IN THE DEFINITION
(*)     PS_ORDER       I    LIST OF PHYSICAL SCHEMA ORDER
(*)     ENTITY_SIZE    0    NUMBER OF RECORDS IN THE DEFINITION
(*)     ENTITY_POSITION 0    FIRST RECORD OF THE DEFINITION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     WRITE BASIC RECORD
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 08 MAY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE DDWRITE *****

```

```

(*) %INCLUDE DEFADD *)
(**)
  PROCEDURE DEFADD(VAR IRC          : RET_REC;
                   VAR IDENTIFIER   : T_NAME;
                   VAR KIND         : INTEGER;
                   VAR REFERENCE_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   BATCH INTERFACE ROUTINE THAT ADDS AN UNRESOLVED ENTITY
(*)   REFERENCE TO THE LIST OF BACKPATCH ENTITIES.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   IRC           0    INTERNAL RETURN CODE
(*)   IDENTIFIER    I    NAME OF ENTITY OR DEFINED TYPE
(*)   KIND          I    KIND NUMBER OF ENTITY
(*)   REFERENCE_KEY I    KEY OF "ENTITY" WITH UNRESOLVED
(*)                   REFERENCE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*)   PERFORM INITIALIZATIONS.
(*)   DETERMINE IF ANY BACKPATCH ENTITIES EXIST IN THE MODEL.
(*)   IF NO BACKPATCH ENTITIES EXIST, CREATE ONE.
(*)   IF BACKPATCH ENTITIES EXIST, DETERMINE IF ONE HAS THE SAME
(*)   NAME OR KIND AS THE ENTITY INPUT.
(*)   IF A BACKPATCH ENTITY HAS THE SAME NAME OR KIND, THEN ADD A
(*)   REFERENCE AS A CONSTITUENT OF THAT BACKPATCH ENTITY.
(*)   IF NO EXISTING BACKPATCH ENTITY HAS THE SAME NAME OR KIND,
(*)   THEN CREATE A NEW BACKPATCH ENTITY.
(*)
(*) $COMMENTS:
(*)

```

CI PS560240032U
April 1990

```
(* $CHANGE CONTROL: *)
(*) *)
(*) ORIGINATED: 04/22/87 C. H. MOHME DBMA *)
(*) *)
(*)-----*)
(*) *)
(*)END-----*)
(*) END %INCLUDE DEFADD *)
```

```
(* %INCLUDE DEFARR *)
(**)
  PROCEDURE DEFARR(VAR IRC          : RET_REC;
                   VAR ENT_KIND     : INTEGER;
                   VAR TRANS_STACK  : TRANSPTR;
                   VAR TOKEN        : T_TOKEN;
                   VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                   VAR TOKEN_LOCATION : INTEGER;
                   VAR TOKEN_LENGTH  : INTEGER;
                   VAR REPORT1      : TEXT);

  SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   BATCH INTERFACE ROUTINE THAT PROCESSES AN ARRAY DEFINITION. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   IRC           0    INTERNAL RETURN CODE *)
(*   ENT_KIND      I    THE KIND OF ENTITY BEING CONSTRUCTED *)
(*   TRANS_STACK   I/O  TRANSACTION STACK *)
(*   TOKEN         I/O  TOKEN FROM BATCH INPUT *)
(*   TOKEN_VALUE   I/O  TOKEN VALUE FROM BATCH INPUT *)
(*   TOKEN_LOCATION I/O  LOCATION OF TOKEN IN INPUT LINE *)
(*   TOKEN_LENGTH  I/O  LENGTH OF TOKEN *)
(*   TEXT         I/O  OUTPUT FILE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(*   INITIALIZE VARIABLES *)
(*   GET ARRAY LOWER BOUND AND VERIFY RANGE *)
(*   GET ARRAY UPPER BOUND AND VERIFY RANGE *)
(*   PUSH ARRAY TRANSACTION ONTO STACK *)
(*   GET THE ARRAY TYPE *)
(*   IF THE ARRAY TYPE IS BASIC TYPE THEN *)
(*     CREATE BASIC CONSTITUENT *)
```

CI PS560240032U
April 1990

```
(* ELSE *)
(* IF THE ARRAY TYPE IS DEFINED TYPE THEN *)
(* CREATE DEFINED TYPE CONSTITUENT *)
(* PRINT ERRORS AS APPROPRIATE *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* ORIGINATED: 03/20/87 C. H. MOHME DBMA *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE DEFARR *)
```

(* %INCLUDE DEFATT *)

(**)

```
PROCEDURE DEFATT(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR TOKEN        : T_TOKEN;
                  VAR TOKEN_VALUE  : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH  : INTEGER;
                  VAR FIELDTYPE    : T_FIELDTYPE;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR CLASS_FLAG    : BOOLEAN;
                  VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR REPORT1       : TEXT);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) BATCH INTERFACE ROUTINE THAT PROCESSES AN ATTRIBUTE (*)
(*) DEFINITION. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC 0 INTERNAL RETURN CODE (*)
(*) TRANS_STACK I/O TRANSACTION STACK (*)
(*) TOKEN I/O TOKEN FROM BATCH INPUT (*)
(*) TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT (*)
(*) TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE (*)
(*) TOKEN_LENGTH I/O LENGTH OF TOKEN (*)
(*) FIELDTYPE I/O TYPE OF ATTRIBUTE (ENTITY OR GLOBAL) (*)
(*) SUBSCHEMA_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN (*)
(*) THE SUBSCHEMA (*)
(*) SUB_ENT_HEAD I/O POINTS TO LIST OF SUBSCHEMA ENTITIES (*)
(*) SUB_ENT_LIST I/O POINTS TO CURRENT ENTITY IN THE LIST OF (*)
(*) SUBSCHEMA ENTITIES (*)
(*) CLASS_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN (*)
(*) THE CLASS (*)
(*) CLS_ENT_HEAD I/O POINTS TO THE LIST OF CLASS ENTITIES (*)
(*) CLS_ENT_LIST I/O POINTS TO CURRENT ENTITY IN THE LIST OF (*)
(*) CLASS ENTITIES (*)
(*) REPORT1 I/O OUTPUT FILE (*)
(*)
(*) $COMMONS: (*)
```



```
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* $PROCESSING DESCRIPTION: *)
(*
(* INITIALIZE VARIABLES *)
(* STORE ATTRIBUTE NAME *)
(* DETERMINE IF THE ATTRIBUTE NAME IS UNIQUE *)
(* IF THE ATTRIBUTE TYPE IS GLOBAL OR STRUCTURE, THEN WE WILL WANT *)
(* A LIST OF ALL OF THE ATTRIBUTES SO THAT WE CAN VERIFY THAT *)
(* THE ATTRIBUTE NAME IS UNIQUE AMONG ALL MODELED ATTRIBUTES. *)
(* IF THE ATTRIBUTE TYPE IS ENTITY THEN WE WILL WANT A LIST OF *)
(* ALL OF THE GLOBAL AND STRUCTURE ATTRIBUTES SO THAT WE CAN *)
(* VERIFY THAT THE ATTRIBUTE NAME IS UNIQUE AMONG ALL MODELED *)
(* GLOBAL AND STRUCTURE ATTRIBUTES. *)
(* VERIFY THAT THE ATTRIBUTE NAME IS UNIQUE AMONG THOSE ATTRIBUTES *)
(* ON THE STACK. *)
(* VERIFY THAT THE ATTRIBUTE NAME IS UNIQUE AMONG THOSE ATTRIBUTES *)
(* ALREADY MODELED. *)
(* IF THE ATTRIBUTE NAME IS UNIQUE THEN PUT THE ATTRIBUTE DATA ON- *)
(* TO THE STACK AND GET NEXT TOKEN. *)
(* DETERMINE IF ANOTHER ATTRIBUTE FOLLOWS THE CURRENT ONE *)
(* (SEPARATED BY A COMMA) OR IF THE TYPE DEFINITION FOR *)
(* THE ATTRIBUTE(S) IS NEXT. *)
(* IF ANOTHER ATTRIBUTE FOLLOWS THEN *)
(* DEFINE THE ATTRIBUTE *)
(* ELSE *)
(* IF ATTRIBUTE TYPE IS BASIC TYPE THEN *)
(* CREATE BASIC TYPE *)
(* ELSE *)
(* IF ATTRIBUTE TYPE IS DEFINED TYPE THEN *)
(* CREATE DEFINED TYPE CONSTITUENT *)
(* PRINT ERRORS AS APPROPRIATE *)
(* $COMMENTS: *)
(* $CHANGE CONTROL: *)
(*
(* REVISED: 30 MARCH 1988 C. H. MOHME DBMA *)
(* MODIFIED TO INCORPORATE ATTRIBUTE PHYSICAL POSITION NUMBER. *)
(*
(* REVISED: JANUARY 1988 C. H. MOHME DBMA *)
```

CI PS560240032U
April 1990

(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE *)
(* *)
(* ORIGINATED: 03/20/87 C. H. MOHME DBMA *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE DEFATT *)

(* %INCLUDE DEFBAS *)
(**)

```

PROCEDURE DEFBAS(VAR IRC           : RET_REC;
                  VAR ENT_KIND      : INTEGER;
                  VAR TRANS_STACK   : TRANSPTR;
                  VAR TOKEN         : T_TOKEN;
                  VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH  : INTEGER;
                  VAR REPORT1       : TEXT);

```

SUBPROGRAM;

```

(**)
(*-----*)
(*
(* $FUNCTION:
(*     BATCH INTERFACE ROUTINE THAT PROCESSES A PRIMITIVE DATA
(*     TYPE DEFINITION.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME           I/O  DESCRIPTION
(*     ====          ===  =====
(*     IRC             0    INTERNAL RETURN CODE
(*     ENT_KIND        0    THE KIND OF ENTITY BEING CONSTRUCTED
(*     TRANS_STACK     I/O  TRANSACTION STACK
(*     TOKEN           I/O  TOKEN FROM BATCH INPUT
(*     TOKEN_VALUE     I/O  TOKEN VALUE FROM BATCH INPUT
(*     TOKEN_LOCATION  I/O  TOKEN VALUE FROM BATCH INPUT
(*     TOKEN_LENGTH    I/O  TOKEN VALUE FROM BATCH INPUT
(*     REPORT1        I/O  OUTPUT FILE
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*     INITIALIZE VARIABLES
(*     IF BASIC TYPE IS INTEGER THEN
(*         GET INTEGER PRECISION
(*     PUSH INTEGER DATA ONTO TRANSACTION STACK
(*     IF BASIC TYPE IS REAL THEN
(*         GET REAL PRECISION

```

```
(*      PUSH REAL DATA ONTO TRANSACTION STACK      *)
(*      IF BASIC TYPE IS STRING THEN                  *)
(*      GET STRING PRECISION                          *)
(*      PUSH STRING DATA ONTO TRANSACTION STACK      *)
(*      IF BASIC TYPE IS LOGICAL THEN                 *)
(*      PUSH LOGICAL DATA ONTO TRANSACTION STACK    *)
(*      IF BASIC TYPE IS ARRAY THEN                  *)
(*      DEFINE ARRAY                                  *)
(*      IF BASIC TYPE IS POINTER THEN                 *)
(*      DEFINE POINTER                                *)
(*      WRITE APPROPRIATE ERROR MESSAGES             *)
(*      $COMMENTS:                                     *)
(*      $CHANGE CONTROL:                              *)
(*      ORIGINATED: 03/20/87      C. H. MOHME      DBMA *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE DEFBAS *)
```

(* %INCLUDE DEFCLS *)

(**)

```
PROCEDURE DEFCLS(VAR IRC           : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR TOKEN        : T_TOKEN;
                  VAR TOKEN_VALUE  : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH : INTEGER;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR CLASS_FLAG    : BOOLEAN;
                  VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR REPORT1       : TEXT);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION:
(*)   BATCH INTERFACE ROUTINE THAT PROCESSES A CLASS DEFINITION.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ==  =====
(*)   IRC            0    INTERNAL RETURN CODE
(*)   TRANS_STACK    I/O  TRANSACTION STACK
(*)   TOKEN          I/O  TOKEN FROM BATCH INPUT
(*)   TOKEN_VALUE    I/O  TOKEN VALUE FROM BATCH INPUT
(*)   TOKEN_LOCATION I/O  LOCATION OF TOKEN IN INPUT LINE
(*)   TOKEN_LENGTH   I/O  LENGTH OF TOKEN
(*)   SUBSCHEMA_FLAG I/O  INDICATES IF ENTITIES OR CLASSES ARE
(*)                       DEFINED WITHIN A SUBSCHEMA
(*)   SUB_ENT_HEAD   I/O  POINTS TO LIST OF SUBSCHEMA ENTITIES
(*)   SUB_ENT_LIST   I/O  POINTS TO CURRENT ENTITY IN LIST OF
(*)                       SUBSCHEMA ENTITIES
(*)   CLASS_FLAG     I/O  INDICATES IF ENTITIES OR CLASSES ARE
(*)                       DEFINED WITHIN A CLASS
(*)   CLS_ENT_HEAD   I/O  POINTS TO LIST OF CLASS ENTITIES
(*)   CLS_ENT_LIST   I/O  POINTS TO CURRENT ENTITY IN LIST OF
(*)                       CLASS ENTITIES
(*)   REPORT1        I/O  OUTPUT FILE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
```

```

(*)      HARDWARE SYSTEM: IBM 360/370/4341/4381      *)
(*)                                                    *)
(*) $EXECUTION PROCEDURE:                             *)
(*)      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*)                                                    *)
(*) $PROCESSING DESCRIPTION:                           *)
(*)                                                    *)
(*)      INITIALIZE VARIABLES                           *)
(*)      GET CLASS NAME AND KIND NUMBER                 *)
(*)      VERIFY UNIQUENESS OF CLASS NAME AND KIND NUMBER *)
(*)      IF THE NEXT TOKEN IS AN IDENTIFIER OR AN INTEGER, THEN WE KNOW *)
(*)      THAT WE WILL BE GETTING A LIST OF ENTITIES AND CLASSES THAT *)
(*)      ARE TO BE MEMBERS OF THE CLASS.  IF THE SUBSCHEMA FLAG OR *)
(*)      CLASS FLAG IS SET, THEN ADD THE CLASS NAME TO THE APPROPRIATE *)
(*)      LIST(S).  PUSH THE CLASS NAME AND KIND NUMBER ONTO THE PRO- *)
(*)      CESSING STACK.                                  *)
(*)      DETERMINE IF THE ENTITY OR CLASS EXISTS         *)
(*)      IF WE HAVE AN ENTITY OR CLASS KEY THEN WE VERIFY THAT THE *)
(*)      ENTITY OR CLASS IS NOT ALREADY A MEMBER OF THE CLASS.  IF *)
(*)      IT IS NOT A MEMBER, WE ATTACH THE KEY TO THE LIST AND PUSH *)
(*)      A TRANSACTION ONTO THE STACK.                   *)
(*)      IF THE ENTITY OR CLASS CONSTITUENT DOES NOT ALREADY EXIST, *)
(*)      CREATE AN UNRESOLVED ENTITY AND MAKE IT A CONSTITUENT OF *)
(*)      THE CLASS.  IF THE ENTITY OR CLASS CONSTITUENT IS LATER *)
(*)      CREATED, THEN IT WILL REPLACE THE UNRESOLVED ENTITY IN THE *)
(*)      CLASS CONSTITUENT LIST.                         *)
(*)      IF WE HAVE READ IN ALL OF THE ENTITY AND CLASS NAMES, WE PUSH *)
(*)      THE FINAL CLASS TRANSACTION ONTO THE STACK AND PROCESS THE *)
(*)      TRANSACTION STACK.                              *)
(*)      IF WE HAVE ENCOUNTERED AN ENTITY OR CLASS DEFINITION, THEN WE *)
(*)      SET A FLAG TO INDICATE THAT AFTER MODELING THESE ENTITIES *)
(*)      AND CLASSES, WE MUST THEN MODEL THE CLASS.     *)
(*)      ADD THE CLASS TO THE SUBSCHEMA AND CLASS LIST, AS NECESSARY *)
(*)      WRITE APPROPRIATE ERROR MESSAGES               *)
(*)                                                    *)
(*) $COMMENTS:                                           *)
(*)                                                    *)
(*) $CHANGE CONTROL:                                     *)
(*)                                                    *)
(*)      ORIGINATED: 03/20/87      C. H. MOHME      DBMA      *)
(*)                                                    *)
(*)-----*)
(*)-----*)
(*)END-----*)
(*) END %INCLUDE DEFCLS *)

```

```
(* %INCLUDE DEFDEF *)
(**)
PROCEDURE DEFDEF(VAR IRC          : RET_REC;
                  VAR TRANS_STACK : TRANSPTR;
                  VAR TOKEN       : T_TOKEN;
                  VAR TOKEN_VALUE : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH : INTEGER;
                  VAR REPORT1     : TEXT);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     BATCH INTERFACE ROUTINE THAT PROCESSES A DEFINED TYPE
(*     REFERENCE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     =====
(*     IRC           0    INTERNAL RETURN CODE
(*     TRANS_STACK   I/O  TRANSACTION STACK
(*     TOKEN         I/O  TOKEN FROM BATCH INPUT
(*     TOKEN_VALUE   I/O  TOKEN VALUE FROM BATCH INPUT
(*     TOKEN_LOCATION I   LOCATION OF TOKEN IN INPUT LINE
(*     TOKEN_LENGTH  I   LENGTH OF TOKEN
(*     REPORT1       I/O  OUTPUT FILE
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*     INITIALIZE THE VARIABLES
(*     DETERMINE IF THE DEFINED TYPE EXISTS
(*     IF DEFINED TYPE EXISTS THEN PUT DEFINED TYPE TRANSACTION ONTO
(*     THE TRANSACTION STACK
(*     BUILD POINTER REFERENCING EXISTING ENTITY OR CLASS OR BUILD
(*     UNRESOLVED ENTITY.
(*     WRITE APPROPRIATE ERROR MESSAGES
(*
```

CI PS560240032U
April 1990

(* COMMENTS: *)
(* *)
(* \$CHANGE CONTROL: *)
(* *)
(* ORIGINATED: 03/20/87 C. H. MOHME DBMA *)
(* *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE DEFDEF *)


```
(* %INCLUDE DEFENM *)
(**)
  PROCEDURE DEFENM(VAR IRC           : RET_REC;
                   VAR ENT_KIND      : INTEGER;
                   VAR TRANS_STACK   : TRANSPTR;
                   VAR TOKEN         : T_TOKEN;
                   VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                   VAR TOKEN_LOCATION : INTEGER;
                   VAR TOKEN_LENGTH  : INTEGER;
                   VAR REPORT1       : TEXT);

  SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   BATCH INTERFACE ROUTINE THAT PROCESSES AN ENUMERATION
(*   DEFINITION.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   IRC       0   INTERNAL RETURN CODE
(*   ENT_KIND   I   THE KIND OF ENTITY BEING CONSTRUCTED
(*   TRANS_STACK I/O TRANSACTION STACK
(*   TOKEN      I/O TOKEN FROM BATCH INPUT
(*   TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT
(*   TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE
(*   TOKEN_LENGTH I/O LENGTH OF TOKEN
(*   REPORT1    I/O OUTPUT FILE
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*   INITIALIZE VARIABLES
(*   PUSH ENUMERATION TRANSACTION ONTO THE STACK
(*   GET EACH ENUMERATION ITEM
(*   VERIFY THE UNIQUENESS OF EACH ENUMERATION ITEM NAME AMONG THE
(*   ENTITIES CURRENTLY MODELED AND THOSE ON THE STACK
(*   IF THE ENUMERATION ITEM NAME IS UNIQUE THEN PUSH A TRANSACTION
```

CI PS560240032U
April 1990

```
(*      ONTO THE STACK                                *)
(*      WRITE APPROPRIATE ERROR MESSAGES              *)
(*                                                    *)
(*      $COMMENTS:                                     *)
(*                                                    *)
(*      $CHANGE CONTROL:                               *)
(*                                                    *)
(*      ORIGINATED: 03/20/87      C. H. MOHME          DBMA *)
(*                                                    *)
(*-----*)
(*                                                    *)
(*END-----*)
(* END %INCLUDE DEFENM *)
```

(* %INCLUDE DEFENT *)

(**)

```
PROCEDURE DEFENT(VAR IRC           : RET_REC;
                 VAR TRANS_STACK   : TRANSPTR;
                 VAR TOKEN         : T_TOKEN;
                 VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                 VAR TOKEN_LOCATION : INTEGER;
                 VAR TOKEN_LENGTH  : INTEGER;
                 VAR SUBSCHEMA_FLAG : BOOLEAN;
                 VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                 VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                 VAR CLASS_FLAG    : BOOLEAN;
                 VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                 VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                 VAR REPORT1       : TEXT);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) BATCH INTERFACE ROUTINE THAT PROCESSES AN ENTITY DEFINITION. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC 0 INTERNAL RETURN CODE (*)
(*) TRANS_STACK I/O TRANSACTION STACK (*)
(*) TOKEN I/O TOKEN FROM BATCH INPUT (*)
(*) TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT (*)
(*) TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE (*)
(*) TOKEN_LENGTH I/O LENGTH OF TOKEN (*)
(*) SUBSCHEMA_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN (*)
(*) THE SUBSCHEMA (*)
(*) SUB_ENT_HEAD I/O POINTS TO LIST OF SUBSCHEMA ENTITIES (*)
(*) SUB_ENT_LIST I/O POINTS TO CURRENT ENTITY IN LIST OF (*)
(*) SUBSCHEMA ENTITIES (*)
(*) CLASS_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN (*)
(*) THE CLASS (*)
(*) CLS_ENT_HEAD I/O POINTS TO LIST OF CLASS ENTITIES (*)
(*) CLS_ENT_LIST I/O POINTS TO CURRENT ENTITY IN LIST OF (*)
(*) CLASS ENTITIES (*)
(*) REPORT1 I/O OUTPUT FILE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
```

```
(*      LANGUAGE: IBM PASCAL                                *)
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381              *)
(*                                                         *)
(* $EXECUTION PROCEDURE:                                     *)
(*      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*                                                         *)
(* $PROCESSING DESCRIPTION:                                  *)
(*                                                         *)
(*      INITIALIZE VARIABLES                                *)
(*      GET THE ENTITY NAME AND KIND NUMBER                  *)
(*      VERIFY THE UNIQUENESS OF THE ENTITY NAME AND KIND NUMBER AMONG *)
(*      MODELED ENTITIES AND THOSE ON THE STACK              *)
(*      IF THE ENTITY NAME AND KIND NUMBER ARE UNIQUE THEN CHECK IF THE *)
(*      SUBSCHEMA FLAG OR CLASS FLAG ARE SET. IF SO, ADD THE ENTITY *)
(*      NAME TO THE APPROPRIATE LIST(S). PUSH ENTITY TRANSACTION ON- *)
(*      TO THE STACK.                                        *)
(*      DEFINE EACH ATTRIBUTE OF THE ENTITY                  *)
(*      WHEN THE END OF THE ENTITY IS ENCOUNTERED, THEN PUSH FINAL *)
(*      ENTITY TRANSACTION ONTO THE TRANSACTION STACK AND PROCESS *)
(*      THE STACK.                                           *)
(*      WRITE ERROR MESSAGES AS APPROPRIATE                  *)
(*                                                         *)
(* $COMMENTS:                                                *)
(*                                                         *)
(* $CHANGE CONTROL:                                          *)
(*                                                         *)
(*      ORIGINATED: 03/20/87      C. H. MOHME              DBMA *)
(*                                                         *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE DEFENT *)
```

April 1990

```
(* %INCLUDE DEFGBL *)
```

```
(**)
```

```
PROCEDURE DEFGBL(VAR IRC           : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR TOKEN        : T_TOKEN;
                  VAR TOKEN_VALUE  : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH  : INTEGER;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR CLASS_FLAG    : BOOLEAN;
                  VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR REPORT1       : TEXT);
```

```
SUBPROGRAM;
```

```
(**)
```

```
(*-----*)
(* *)
(* $FUNCTION: *)
(* BATCH INTERFACE ROUTINE THAT PROCESSES A GLOBAL ATTRIBUTE *)
(* DEFINITION. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 INTERNAL RETURN CODE *)
(* TRANS_STACK I/O TRANSACTION STACK *)
(* TOKEN I/O TOKEN FROM BATCH INPUT *)
(* TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT *)
(* TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE *)
(* TOKEN_LENGTH I/O LENGTH OF TOKEN *)
(* SUBSCHEMA_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN *)
(* THE SUBSCHEMA *)
(* SUB_ENT_HEAD I/O POINTS TO LIST OF SUBSCHEMA ENTITIES *)
(* SUB_ENT_LIST I/O POINTS TO CURRENT ENTITY IN LIST OF *)
(* SUBSCHEMA ENTITIES *)
(* CLASS_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN *)
(* THE CLASS *)
(* CLS_ENT_HEAD I/O POINTS TO LIST OF CLASS ENTITIES *)
(* CLS_ENT_LIST I/O POINTS TO CURRENT ENTITY IN LIST OF *)
(* CLASS ENTITIES *)
(* REPORT1 I/O OUTPUT FILE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
```

```
(*      LANGUAGE: IBM PASCAL                                *)
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381              *)
(*                                                         *)
(* $EXECUTION PROCEDURE:                                     *)
(*      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*                                                         *)
(* $PROCESSING DESCRIPTION:                                  *)
(*                                                         *)
(*      PERFORM INITIALIZATIONS                             *)
(*      GET GLOBAL ATTRIBUTE NAME                           *)
(*      PUSH INITIAL GLOBAL ATTRIBUTE TRANSACTION ONTO THE STACK *)
(*      DEFINE THE GLOBAL ATTRIBUTE                         *)
(*      IF GLOBAL ATTRIBUTE ACCEPTED, THEN PUSH FINAL GLOBAL ATTRIBUTE *)
(*      TRANSACTION ONTO THE STACK AND PROCESS.  IF THE GLOBAL ATTRI- *)
(*      BUTE NOT ACCEPTED THEN RECOVER.                     *)
(*      WRITE APPROPRIATE ERROR MESSAGES                    *)
(*                                                         *)
(* $COMMENTS:                                                *)
(*                                                         *)
(* $CHANGE CONTROL:                                          *)
(*                                                         *)
(*      ORIGINATED: 05/15/87      C. H. MOHME              DBMA *)
(*                                                         *)
(*-----*)
(*
(*END-----*)
(* END %INCLUDE DEFGBL *)
```

```
(* %INCLUDE DEFPRE *)
(**)
  PROCEDURE DEFPRE(VAR IRC           : RET_REC;
                   VAR ENT_KIND      : INTEGER;
                   VAR TRANS_STACK   : TRANSPTR;
                   VAR TOKEN         : T_TOKEN;
                   VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                   VAR TOKEN_LOCATION : INTEGER;
                   VAR TOKEN_LENGTH  : INTEGER;
                   VAR DATA         : TRANSACTION;
                   VAR PRECISION     : INTEGER;
                   VAR REPORT1       : TEXT);

  SUBPROGRAM;

(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   BATCH INTERFACE ROUTINE THAT PROCESSES AN INTEGER PRECISION, *)
(*   REAL PRECISION, OR STRING LENGTH. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  ===== *)
(*   IRC        0   INTERNAL RETURN CODE *)
(*   ENT_KIND    I   THE KIND OF ENTITY BEING CONSTRUCTED *)
(*   TRANS_STACK I/O  TRANSACTION STACK *)
(*   TOKEN       I/O  TOKEN FROM BATCH INPUT *)
(*   TOKEN_VALUE I/O  TOKEN VALUE FROM BATCH INPUT *)
(*   TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE *)
(*   TOKEN_LENGTH I/O LENGTH OF TOKEN *)
(*   DATA       I/O  CONTAINS TRANSACTION INFORMATION *)
(*   PRECISION   0   INTEGER PRECISION, REAL PRECISION, OR *)
(*                   STRING LENGTH *)
(*   REPORT1     I/O  OUTPUT FILE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(*   INITIALIZE VARIABLES *)
```

```
(*  GET THE PRECISION, IF PRESENT                                *)
(*  IF THE PRECISION IS SPECIFIED, THEN VERIFY THAT IT IS IN THE *)
(*  PROPER RANGE.                                                *)
(*  IF THE PRECISION IS NOT SPECIFIED, THEN ASSIGN THE DEFAULT  *)
(*  PRECISION.                                                    *)
(*  WRITE APPROPRIATE ERROR MESSAGES                             *)
(*                                                                *)
(*  $COMMENTS:                                                    *)
(*                                                                *)
(*  $CHANGE CONTROL:                                              *)
(*                                                                *)
(*  ORIGINATED: 03/20/87      C. H. MOHME                        DBMA *)
(*                                                                *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE DEFPRE *)
```



```
(* %INCLUDE DEFPTR *)
(**)
PROCEDURE DEFPTR(VAR IRC           : RET_REC;
                  VAR ENT_KIND      : INTEGER;
                  VAR TRANS_STACK   : TRANSPTR;
                  VAR TOKEN         : T_TOKEN;
                  VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH  : INTEGER;
                  VAR REPORT1       : TEXT);

SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* BATCH INTERFACE ROUTINE THAT PROCESSES A POINTER DEFINITION.*)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 INTERNAL RETURN CODE *)
(* ENT_KIND I THE KIND OF ENTITY BEING CONSTRUCTED *)
(* TRANS_STACK I/O TRANSACTION STACK *)
(* TOKEN I/O TOKEN FROM BATCH INPUT *)
(* TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT *)
(* TOKEN_LOCATION I/O TOKEN VALUE FROM BATCH INPUT *)
(* TOKEN_LENGTH I/O TOKEN VALUE FROM BATCH INPUT *)
(* REPORT1 I/O OUTPUT FILE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* INITIALIZE VARIABLES *)
(* PUSH INITIAL POINTER TRANSACTION ONTO THE TRANSACTION STACK *)
(* GET EACH POINTER CONSTITUENT NAME OR KIND NUMBER. *)
(* DETERMINE IF THE ENTITY OR CLASS EXISTS *)
(* VERIFY THAT THE ENTITY OR CLASS IS NOT ALREADY A CONSTITUENT. *)
(* IF IT IS NOT, THEN PUSH ENTITY KEY TRANSACTION ONTO THE *)
(* TRANSACTION STACK. *)
```

```
(*      IF THE POINTER REFERENCES AN ENTITY THAT IS NOT DEFINED, THEN *)
(*      CREATE AN UNRESOLVED ENTITY.  IF THE UNDEFINED ENTITY IS      *)
(*      LATER DEFINED, IT WILL REPLACE THE UNRESOLVED ENTITY IN THE   *)
(*      POINTER'S CONSTITUENT LIST.                                     *)
(*      PUSH THE FINAL POINTER TRANSACTION ONTO THE TRANSACTION STACK  *)
(*      WRITE APPROPRIATE ERROR MESSAGES                               *)
(*                                                                    *)
(* $COMMENTS:                                                           *)
(*                                                                    *)
(* $CHANGE CONTROL:                                                     *)
(*                                                                    *)
(*      ORIGINATED: 03/20/87      C. H. MOHME      DBMA      *)
(*                                                                    *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE DEFPTR *)
```

```
(* %INCLUDE DEFQUERY *)
(**)
PROCEDURE DEFQUERY(VAR IRC                : RET_REC;
                   VAR IDENTIFIER          : T_NAME;
                   VAR KIND                 : INTEGER;
                   VAR DEFINITION_KEY_KIND : INTEGER;
                   VAR DEFINITION_KEY      : ENTKEY);

SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* BATCH INTERFACE ROUTINE THAT DETERMINES IF A NEWLY MODELED *)
(* ENTITY SATISFIES ANY UNRESOLVED ENTITY REFERENCES ON THE *)
(* BACKPATCH LIST. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 INTERNAL RETURN CODE *)
(* IDENTIFIER I NAME OF ENTITY, CLASS, OR DEFINED TYPE *)
(* KIND I KIND NUMBER OF ENTITY OR CLASS *)
(* ENT_KIND I THE KIND OF ENTITY (DEFINED TYPE, ENTITY *)
(* OR CLASS) *)
(* DEFINITION_KEY I KEY OF "ENTITY" WITH UNRESOLVED *)
(* REFERENCE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* INITIALIZE VARIABLES *)
(* DETERMINE IF ANY BACKPATCH ENTITIES EXIST IN THE MODEL *)
(* IF BACKPATCH ENTITIES EXIST, DETERMINE IF ONE MATCHES THE NAME *)
(* OR KIND OF THE NEWLY CREATED ENTITY. *)
(* IF A MATCH HAS BEEN FOUND, THEN WE CAN RESOLVE EACH REFERENCE *)
(* OF THE BACKPATCH ENTITY. *)
(* IF THE NEWLY CREATED ENTITY IS A DEFINED TYPE, THEN WE REPLACE *)
(* THE CONSTITUENT OF THE REFERENCE KEY WITH THE KEY OF THE *)
(* NEWLY DEFINED ENTITY. *)
```

```
(* IF THE REFERENCE KEY IS A CONSTITUENT OF A CLASS, THEN WE PUT *)
(* THE KEY OF THE NEWLY DEFINED ENTITY IN THE CLASS' CONSTITU- *)
(* ENT LIST IN PLACE OF THE REFERENCE KEY. *)
(* IF THE REFERENCE KEY IS A CONSTITUENT OF A SUBSCHEMA, THEN WE *)
(* PUT THE KEY OF THE NEWLY DEFINED ENTITY IN THE SUBSCHEMA'S *)
(* CONSTITUENT LIST IN PLACE OF THE REFERENCE KEY. *)
(* IF THE REFERENCE KEY IS A CONSTITUENT OF A POINTER, THEN WE *)
(* PUT THE KEY OF THE NEWLY DEFINED ENTITY IN THE POINTER'S *)
(* CONSTITUENT LIST IN PLACE OF THE REFERENCE KEY. *)
(* IF THE NEWLY DEFINED ENTITY IS NOT A DEFINED TYPE AND IF THE *)
(* REFERENCE KEY IS NOT A CONSTITUENT OF A SUBSCHEMA, CLASS OR *)
(* POINTER THEN WE BUILD A POINTER (WHICH REFERENCES THE *)
(* NEWLY DEFINED ENTITY) AND MAKE IT A CONSTITUENT OF THE *)
(* REFERENCE KEY. *)
(* DELETE THE BACKPATCH ENTITY AND READ THE NEXT ONE *)
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* ORIGINATED: 04/22/87 C. H. MOHME DBMA
(*
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE DEFQUERY *)
```

```
(* %INCLUDE DEFSTC *)
(**)
PROCEDURE DEFSTC(VAR IRC          : RET_REC;
                 VAR TRANS_STACK  : TRANSPTR;
                 VAR TOKEN        : T_TOKEN;
                 VAR TOKEN_VALUE  : T_TOKEN_VALUE;
                 VAR TOKEN_LOCATION : INTEGER;
                 VAR TOKEN_LENGTH : INTEGER;
                 VAR SUBSCHEMA_FLAG : BOOLEAN;
                 VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                 VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                 VAR CLASS_FLAG    : BOOLEAN;
                 VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                 VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                 VAR REPORT1      : TEXT);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*     BATCH INTERFACE ROUTINE THAT PROCESSES A STRUCTURE
(*     DEFINITION.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC           0    INTERNAL RETURN CODE
(*     TRANS_STACK   I/O  TRANSACTION STACK
(*     TOKEN         I/O  TOKEN FROM BATCH INPUT
(*     TOKEN_VALUE   I/O  TOKEN VALUE FROM BATCH INPUT
(*     TOKEN_LOCATION I/O  LOCATION OF TOKEN IN INPUT LINE
(*     TOKEN_LENGTH  I/O  LENGTH OF TOKEN
(*     SUBSCHEMA_FLAG I/O  INDICATES IF ENTITIES ARE DEFINED WITHIN
(*                          THE SUBSCHEMA
(*     SUB_ENT_HEAD  I/O  POINTS TO LIST OF SUBSCHEMA ENTITIES
(*     SUB_ENT_LIST  I/O  POINTS TO CURRENT ENTITY IN LIST OF
(*                          SUBSCHEMA ENTITIES
(*     CLASS_FLAG    I/O  INDICATES IF ENTITIES ARE DEFINED WITHIN
(*                          THE CLASS
(*     CLS_ENT_HEAD  I/O  POINTS TO LIST OF CLASS ENTITIES
(*     CLS_ENT_LIST  I/O  POINTS TO CURRENT ENTITY IN LIST OF
(*                          CLASS ENTITIES
(*     REPORT1      I/O  OUTPUT FILE
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
```

```
(*      LANGUAGE: IBM PASCAL                      *)
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381      *)
(*      *)                                           *)
(* $EXECUTION PROCEDURE:                          *)
(*      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*      *)                                           *)
(* $PROCESSING DESCRIPTION:                        *)
(*      *)                                           *)
(*      PERFORM INITIALIZATIONS                    *)
(*      PUSH INITIAL STRUCTURE TRANSACTION ONTO THE STACK *)
(*      DEFINE STRUCTURE ATTRIBUTES                *)
(*      RECOVER FROM ERROR, AS NECESSARY           *)
(*      PUSH FINAL STRUCTURE TRANSACTION ONTO THE STACK AND GET NEXT *)
(*      TOKEN                                       *)
(*      WRITE APPROPRIATE ERROR MESSAGES           *)
(*      *)                                           *)
(* $COMMENTS:                                      *)
(*      *)                                           *)
(* $CHANGE CONTROL:                               *)
(*      *)                                           *)
(*      ORIGINATED: 05/18/87      C. H. MOHME      DBMA      *)
(*      *)                                           *)
(*-----*)
(*                                           *)
(*END-----*)
(* END %INCLUDE DEFSTC *)
```

(* %INCLUDE DEFSUB *)

(**)

```
PROCEDURE DEFSUB(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR TOKEN        : T_TOKEN;
                  VAR TOKEN_VALUE   : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH  : INTEGER;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR CLASS_FLAG    : BOOLEAN;
                  VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR REPORT1       : TEXT);
```

SUBPROGRAM;

(**)

```
(*-----*)
(* *)
(* $FUNCTION: *)
(* BATCH INTERFACE ROUTINE THAT PROCESSES A SUBSCHEMA *)
(* DEFINITION. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* === === *)
(* IRC 0 INTERNAL RETURN CODE *)
(* TRANS_STACK I/O TRANSACTION STACK *)
(* TOKEN I/O TOKEN FROM BATCH INPUT *)
(* TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT *)
(* TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE *)
(* TOKEN_LENGTH I/O LENGTH OF TOKEN *)
(* SUBSCHEMA_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN *)
(* THE SUBSCHEMA *)
(* SUB_ENT_HEAD I/O POINTS TO THE LIST OF SUBSCHEMA ENTITIES *)
(* SUB_ENT_LIST I/O POINTS TO THE CURRENT ENTITY IN THE LIST *)
(* OF SUBSCHEMA ENTITIES *)
(* CLASS_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN *)
(* THE CLASS *)
(* CLS_ENT_HEAD I/O POINTS TO THE LIST OF CLASS ENTITIES *)
(* CLS_ENT_LIST I/O POINTS TO THE CURRENT ENTITY IN THE LIST *)
(* OF CLASS ENTITIES *)
(* REPORT1 I/O OUTPUT FILE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
```

```

(*)      LANGUAGE: IBM PASCAL                                *)
(*)      HARDWARE SYSTEM: IBM 360/370/4341/4381              *)
(*)                                                    *)
(*) $EXECUTION PROCEDURE:                                    *)
(*)      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*)                                                    *)
(*) $PROCESSING DESCRIPTION:                                  *)
(*)                                                    *)
(*)      INITIALIZE VARIABLES                                *)
(*)      GET SUBSCHEMA NAME AND VERIFY ITS UNIQUENESS         *)
(*)      IF SUBSCHEMA NAME IS UNIQUE THEN GET NEXT MEANINGFUL TOKEN *)
(*)      IF THE TOKEN IS AN IDENTIFIER OR AN INTEGER, THEN WE KNOW THAT *)
(*)      WE WILL BE GETTING A LIST OF ENTITIES AND CLASSES THAT ARE TO *)
(*)      BE MEMBERS OF THE SUBSCHEMA.  PUSH THE SUBSCHEMA NAME ONTO *)
(*)      THE PROCESSING STACK.                                *)
(*)      DETERMINE IF THE IDENTIFIER IS AN ENTITY OR A CLASS  *)
(*)      IF WE HAVE AN ENTITY OR CLASS KEY, THEN WE VERIFY THAT IT IS *)
(*)      NOT ALREADY A MEMBER OF THE SUBSCHEMA.  IF IT IS NOT, THEN *)
(*)      WE ATTACH THE ENTITY KEY TO THE LIST AND PUSH A TRANS- *)
(*)      ACTION ONTO THE TRANSACTION STACK.                    *)
(*)      IF THE SUBSCHEMA'S CONSTITUENT IS NOT YET DEFINED, CREATE AN *)
(*)      UNRESOLVED ENTITY.  WHEN THE CONSTITUENT IS LATER DEFINED, *)
(*)      IF WILL REPLACE THE UNRESOLVED ENTITY IN THE CONSTITUENT *)
(*)      LIST OF THE SUBSCHEMA.                                *)
(*)      IF WE HAVE READ IN ALL OF THE ENTITY AND CLASS NAMES, WE PUSH *)
(*)      THE FINAL SUBSCHEMA TRANSACTION ONTO THE STACK AND PROCESS *)
(*)      THE TRANSACTION STACK.                                *)
(*)      IF WE HAVE ENCOUNTERED AN ENTITY OR CLASS DEFINITION, THEN WE *)
(*)      SET A FLAG TO INDICATE THAT AFTER MODELING THESE ENTITIES *)
(*)      AND CLASSES, WE MUST THEN MODEL THE SUBSCHEMA.      *)
(*)      WRITE APPROPRIATE ERROR MESSAGES.                    *)
(*)                                                    *)
(*) $COMMENTS:                                                *)
(*)                                                    *)
(*) $CHANGE CONTROL:                                          *)
(*)                                                    *)
(*)      ORIGINATED: 03/20/87      C. H. MOHME      DBMA      *)
(*)                                                    *)
(*)-----*)
(*)                                                    *)
(*)END-----*)
(*) END %INCLUDE DEFSUB *)

```


(* %INCLUDE DEFSUP *)

(**)

```
PROCEDURE DEFSUP(VAR IRC           : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR TOKEN        : T_TOKEN;
                  VAR TOKEN_VALUE  : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH : INTEGER;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR CLASS_FLAG    : BOOLEAN;
                  VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR REPORT1       : TEXT);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) BATCH INTERFACE ROUTINE THAT PROCESSES A SUPERTYPE (*)
(*) DEFINITION. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC 0 INTERNAL RETURN CODE (*)
(*) TRANS_STACK I/O TRANSACTION STACK (*)
(*) TOKEN I/O TOKEN FROM BATCH INPUT (*)
(*) TOKEN_VALUE I/O TOKEN VALUE FROM BATCH INPUT (*)
(*) TOKEN_LOCATION I/O LOCATION OF TOKEN IN INPUT LINE (*)
(*) TOKEN_LENGTH I/O LENGTH OF TOKEN (*)
(*) SUBSCHEMA_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN (*)
(*) THE SUBSCHEMA (*)
(*) SUB_ENT_HEAD I/O POINTS TO LIST OF SUBSCHEMA ENTITIES (*)
(*) SUB_ENT_LIST I/O POINTS TO CURRENT ENTITY IN LIST OF (*)
(*) SUBSCHEMA ENTITIES (*)
(*) CLASS_FLAG I/O INDICATES IF ENTITIES ARE DEFINED WITHIN (*)
(*) THE CLASS (*)
(*) CLS_ENT_HEAD I/O POINTS TO LIST OF CLASS ENTITIES (*)
(*) CLS_ENT_LIST I/O POINTS TO CURRENT ENTITY IN LIST OF (*)
(*) CLASS ENTITIES (*)
(*) REPORT1 I/O OUTPUT FILE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
```

```
(*      LANGUAGE: IBM PASCAL                                *)
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381              *)
(*      $EXECUTION PROCEDURE:                                *)
(*          INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*      $PROCESSING DESCRIPTION:                              *)
(*      INITIALIZE VARIABLES                                  *)
(*      GET THE ENTITY NAME AND KIND NUMBER                  *)
(*      VERIFY THE UNIQUENESS OF THE ENTITY NAME AND KIND NUMBER AMONG *)
(*      MODELED ENTITIES AND THOSE ON THE STACK              *)
(*      IF THE ENTITY NAME AND KIND NUMBER ARE UNIQUE THEN CHECK IF THE *)
(*      SUBSCHEMA FLAG OR CLASS FLAG ARE SET. IF SO, ADD THE ENTITY *)
(*      NAME TO THE APPROPRIATE LIST(S). PUSH ENTITY TRANSACTION ON- *)
(*      TO THE STACK.                                         *)
(*      DEFINE EACH ATTRIBUTE OF THE ENTITY                  *)
(*      WHEN THE END OF THE ENTITY IS ENCOUNTERED, THEN PUSH FINAL *)
(*      ENTITY TRANSACTION ONTO THE TRANSACTION STACK AND PROCESS *)
(*      THE STACK.                                           *)
(*      WRITE ERROR MESSAGES AS APPROPRIATE                  *)
(*      $COMMENTS:                                           *)
(*      $CHANGE CONTROL:                                      *)
(*      ORIGINATED: 09/29/87      C. H. MOHME              DBMA *)
(*      -----*)
(*      *)
(*END-----*)
(* END %INCLUDE DEFSUP *)
```

```
(* %INCLUDE DEFTYP *)
(**)
PROCEDURE DEFTYP(VAR IRC           : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR TOKEN        : T_TOKEN;
                  VAR TOKEN_VALUE  : T_TOKEN_VALUE;
                  VAR TOKEN_LOCATION : INTEGER;
                  VAR TOKEN_LENGTH  : INTEGER;
                  VAR SUBSCHEMA_FLAG : BOOLEAN;
                  VAR SUB_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR SUB_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR CLASS_FLAG    : BOOLEAN;
                  VAR CLS_ENT_HEAD  : ENTITY_LIST_PTR;
                  VAR CLS_ENT_LIST  : ENTITY_LIST_PTR;
                  VAR REPORT1      : TEXT);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     BATCH INTERFACE ROUTINE THAT PROCESSES A DEFINED TYPE
(*     DEFINITION.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC           0    INTERNAL RETURN CODE
(*     TRANS_STACK   I/O  TRANSACTION STACK
(*     TOKEN         I/O  TOKEN FROM BATCH INPUT
(*     TOKEN_VALUE   I/O  TOKEN VALUE FROM BATCH INPUT
(*     TOKEN_LOCATION I/O  LOCATION OF TOKEN IN INPUT LINE
(*     TOKEN_LENGTH  I/O  LENGTH OF TOKEN
(*     SUBSCHEMA_FLAG I/O  INDICATES IF ENTITIES ARE DEFINED WITHIN
(*                          THE SUBSCHEMA
(*     SUB_ENT_HEAD   I/O  POINTS TO LIST OF SUBSCHEMA ENTITIES
(*     SUB_ENT_LIST   I/O  POINTS TO CURRENT ENTITY IN LIST OF
(*                          SUBSCHEMA ENTITIES
(*     CLASS_FLAG     I/O  INDICATES IF ENTITIES ARE DEFINED WITHIN
(*                          THE CLASS
(*     CLS_ENT_HEAD   I/O  POINTS TO LIST OF CLASS ENTITIES
(*     CLS_ENT_LIST   I/O  POINTS TO CURRENT ENTITY IN LIST OF
(*                          CLASS ENTITIES
(*     REPORT1       I/O  OUTPUT FILE
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
```

```
(*      LANGUAGE: IBM PASCAL                                *)
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381              *)
(*                                                         *)
(* $EXECUTION PROCEDURE:                                    *)
(*      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(*                                                         *)
(* $PROCESSING DESCRIPTION:                                  *)
(*                                                         *)
(*      INITIALIZE VARIABLES                                *)
(*      GET THE DEFINED TYPE NAME AND VERIFY ITS UNIQUENESS AMONG *)
(*      ENTITIES IN THE MODEL AND ON THE STACK              *)
(*      IF THE DEFINED TYPE NAME IS UNIQUE, THEN PUSH DEFINED TYPE *)
(*      TRANSACTION ONTO THE TRANSACTION STACK.             *)
(*      GET THE TYPE DEFINITION. THE TYPE MAY BE AN ENUMERATION, A *)
(*      BASIC TYPE, OR A DEFINED TYPE.                      *)
(*      IF THE TYPE DEFINITION IS ENUMERATION THEN          *)
(*      DEFINE ENUMERATION                                  *)
(*      ELSE                                                 *)
(*      IF THE TYPE DEFINITION IS STRUCTURE THEN            *)
(*      DEFINE STRUCTURE                                     *)
(*      ELSE                                                 *)
(*      IF THE TYPE DEFINITION IS A BASICA TYPE THEN       *)
(*      DEFINE BASIC TYPE                                    *)
(*      ELSE                                                 *)
(*      IF THE TYPE DEFINITION IS A DEFINED TYPE THEN      *)
(*      DEFINED DEFINED TYPE                                *)
(*      PROCESS THE TRANSACTION STACK                       *)
(*      WRITE APPROPRIATE ERROR MESSAGES                   *)
(*      GET NEXT DEFINED TYPE NAME IF PRESENT               *)
(*                                                         *)
(* $COMMENTS:                                              *)
(*                                                         *)
(* $CHANGE CONTROL:                                        *)
(*                                                         *)
(*      ORIGINATED: 03/20/87      C. H. MOHME              DBMA *)
(*      -----*)
(*      -----*)
(*END-----*)
(* END %INCLUDE DEFTYP *)
```

```
(* %INCLUDE DISPLIST *)
(**)
PROCEDURE DISPLIST(VAR MESS          : MESSAGE;
                   VAR MEMBERLIST    : T_ARRAYTV;
                   VAR MAX_ARRAYSIZE : INTEGER;
                   VAR NAME           : T_NAME;
                   VAR NEXT_OP        : OPERATIONS;
                   VAR RR              : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE DISPLAYS A LIST OF ENTITIES. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* MESS I THE ERROR MESSAGE RECEIVED FROM MAINLINE *)
(* MEMBERLIST I THE ARRAY OF MEMBERS TO SELECT FROM *)
(* MAX_ARRAYSIZE I THE SIZE OF THE ARRAY OF MEMBERS *)
(* NAME O THE MEMBER SELECTED *)
(* NEXT_OP O TELLS THE MAINLINE WHAT PANEL TO CALL *)
(* NEXT *)
(* RR O TELLS THE MAINLINE IF THERE IS AN ERROR *)
(* AND IN WHAT ROUTINE IT OCCURS *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* DDNAMES USED WITH STANDARD FILES: *)
(* NONE *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* DISPLAY THE DISPLAY LIST PANEL (DISPLIST) BY MAKING ISPLNK *)
(* CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(* TYPE. THE MEMBER SELECTED IS RETURNED TO THE CALLING PRO- *)
(* CEDURE. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE ENTCLS *)
(**)
  PROCEDURE ENTCLS(VAR IRC          : RET_REC;
                   VAR DATA_REC    : BLKDATA);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   BATCH INTERFACE ROUTINE THAT DETERMINES IF A SPECIFIED
(*   IDENTIFIER IS A MODELED ENTITY OR CLASS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    INTERNAL RETURN CODE
(*   DATA_REC     I/O  CONTAINS ENTITY DATA
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*   INITIALIZE VARIABLES
(*   DETERMINE IF THE NAME IS AN ENTITY.  IF IT IS, GET THE KEY.
(*   IF THE NAME IS NOT AN ENTITY, DETERMINE IF IT IS A CLASS.  IF
(*   IT IS, GET THE KEY.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   ORIGINATED: 03/20/87          C. H. MOHME          DBMA
(*
(*-----*)
(*
(*END-----*)
(* END %INCLUDE ENTCLS *)
```

```
(* %INCLUDE ERRMSG *)
(**)
PROCEDURE ERRMSG(VAR ENT_KIND      : INTEGER;
                  VAR REPORT1      : TEXT);
SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* BATCH INTERFACE ROUTINE THAT WRITES APPROPRIATE ERROR *)
(* MESSAGES TO THE REPORT FILE. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== *)
(* ENT_KIND I THE KIND OF ENTITY BEING CREATED *)
(* REPORT1 I/O THE BATCH OUTPUT FILE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* $COMMENTS: *)
(* WRITE APPROPRIATE ERROR MESSAGES FOR THE SUBSCHEMA, CLASS, *)
(* ENTITY, SUPERTYPE, DEFINED TYPE, STRUCTURE, AND GLOBAL *)
(* ATTRIBUTE KINDS. *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* ORIGINATED: 12/12/87 C. H. MOHME DBMA *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE ERRMSG *)
```

```

(*) %INCLUDE ERRREC *)
(**)
  PROCEDURE ERRREC(VAR IRC          : RET_REC;
                   VAR ENT_KIND     : INTEGER;
                   VAR TRANS_STACK  : TRANSPTR;
                   VAR SUBSCHEMA_FLAG : BOOLEAN;
                   VAR SUB_ENT_HEAD : ENTITY_LIST_PTR;
                   VAR SUB_ENT_LIST : ENTITY_LIST_PTR;
                   VAR CLASS_FLAG   : BOOLEAN;
                   VAR CLS_ENT_HEAD : ENTITY_LIST_PTR;
                   VAR CLS_ENT_LIST : ENTITY_LIST_PTR);

    SUBPROGRAM;
  (**)
  (*)-----*)
  (*)
  (*) $FUNCTION: *)
  (*)   BATCH INTERFACE ROUTINE THAT PERFORMS THE NECESSARY ACTIONS *)
  (*)   TO RECOVER FROM AN INPUT ERROR. *)
  (*)
  (*) $DESCRIPTION OF ARGUMENTS: *)
  (*)   NAME          I/O  DESCRIPTION *)
  (*)   ====          ==  ===== *)
  (*)   IRC           I/O  INTERNAL RETURN CODE *)
  (*)   ENT_KIND      I    KIND OF ENTITY *)
  (*)   TRANS_STACK   I/O  TRANSACTION STACK *)
  (*)   SUBSCHEMA_FLAG I/O  INDICATES IF ENTITIES ARE DEFINED WITHIN *)
  (*)                   THE SUBSCHEMA *)
  (*)   SUB_ENT_HEAD   I/O  POINTS TO THE LIST OF SUBSCHEMA ENTITIES *)
  (*)   SUB_ENT_LIST   I/O  POINTS TO THE CURRENT ENTITY IN THE LIST *)
  (*)                   OF SUBSCHEMA ENTITIES *)
  (*)   CLASS_FLAG     I/O  INDICATES IF ENTITIES ARE DEFINED WITHIN *)
  (*)                   THE CLASS *)
  (*)   CLS_ENT_HEAD   I/O  POINTS TO THE LIST OF CLASS ENTITIES *)
  (*)   CLS_ENT_LIST   I/O  POINTS TO THE CURRENT ENTITY IN THE LIST *)
  (*)                   OF SUBSCHEMA ENTITIES *)
  (*)
  (*) $COMMONS: *)
  (*)
  (*) $ENVIRONMENT: *)
  (*)   LANGUAGE: IBM PASCAL *)
  (*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
  (*)
  (*) $EXECUTION PROCEDURE: *)
  (*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT *)
  (*)
  (*) $PROCESSING DESCRIPTION: *)

```



```

(*)
(*) INITIALIZE VARIABLES (*)
(*) CLEAR TRANSACTION STACK (*)
(*) CASE TYPE OF RECOVERY OF (*)
(*) SUBSCHEMA : DISCARD TOKENS UNTIL END_SCHEMA ENCOUNTERED (*)
(*) CLASS : DISCARD TOKENS UNTIL END_CLASS ENCOUNTERED (*)
(*) IF CLASS IS IN ANOTHER CLASS, DISCARD TOKENS (*)
(*) UNTIL END_CLASS IS ENCOUNTERED (*)
(*) IF CLASS IS IN A SUBSCHEMA, DISCARD TOKENS UNTIL (*)
(*) END_SCHEMA ENCOUNTERED (*)
(*) ENTITY : DISCARD TOKENS UNTIL END_ENTITY ENCOUNTERED (*)
(*) IF ENTITY IS IN A CLASS, DISCARD TOKENS UNTIL (*)
(*) END_CLASS ENCOUNTERED (*)
(*) IF ENTITY IS IN A SUBSCHEMA, DISCARD TOKENS (*)
(*) UNTIL END_SCHEMA ENCOUNTERED (*)
(*) DEFINED (*)
(*) TYPE : DISCARD TOKENS UNTIL SEMICOLON ENCOUNTERED (*)
(*) IF DEFINED TYPE IS IN SUBSCHEMA DISCARD TOKENS (*)
(*) UNTIL END_SCHEMA ENCOUNTERED (*)
(*) SET FLAGS TO FALSE (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)
(*) ORIGINATED: 03/20/87 C. H. MOHME DBMA (*)
(*)
(*)-----(*)
(*)
(*)END-----(*)
(*) END %INCLUDE ERRREC *)

```

```
(* BEGIN %INCLUDE GETDD *****)
(*)
PROCEDURE GETDD ( CONST KIND          : INTEGER;
                  CONST MAX_AVAIL     : INTEGER;
                  CONST ATTRIBUTE_ORDER : CHAR;
                  VAR  USER_ARRAY     : T_USER_ARRAY;
                  VAR  MAX_ACTUAL      : INTEGER;
                  VAR  RETURN_CODE     : INTEGER );
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   READ THE DATA DICTIONARY INTO THE APPLICATION PROGRAM.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ===  =====
(*)   KIND           I    A KIND NUMBER OF ENTITY
(*)   MAX_ACTUAL      O    AN ACTUAL NUMBER OF RECORDS IN
(*)                       ENTITY DEFINITION
(*)   MAX_AVAIL       I    A NUMBER OF 80 CHARACTER RECORDS
(*)                       AVAILABLE IN CALLER TO HOLE
(*)                       ENTITY DEFINITION
(*)   USER_ARRAY      O    AN ENTITY DEFINITION
(*)   RETURN_CODE     O    RETURN CODE
(*)                       -1 = ACTUAL SIZE GREATER THAN
(*)                       SPACE AVAILABLE
(*)                       0 = SUCCESS
(*)                       1 = KIND NOT IN DATA DICTIONARY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   LOOP THROUGH DATA DICTIONARY INDEX FILE
(*)   IF KIND IN DATA DICTIONARY THEN
(*)       GET ENTITY DEFINITION FROM DDFILE
(*)       FILL UP THE ARRAY OF ENTITY DEFINITIONS UP TO NUMBER
(*)       OF RECORDS AVAILABLE IN CALLER
(*)   END IF
(*)   END LOOP
(*)
```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) ORIGINATED: 23 MARCH 1987, M. H. CHOI, DBMA *)
(*) *)
(*) END %INCLUDE GETDD *****)
```

```
(* BEGIN %INCLUDE LEXICAL *****)
(*)
PROCEDURE LEXICAL ( VAR    TOKEN           : T_TOKEN;
                   VAR    TOKEN_VALUE      : T_TOKEN_VALUE;
                   VAR    TOKEN_LOCATION   : INTEGER;
                   VAR    TOKEN_LENGTH     : INTEGER;
                   VAR    REPORT1          : TEXT);
    SUBPROGRAM;
(*)
(*) $FUNCTION:
(*) LOCATE THE LONGEST POSSIBLE LEXEME FROM WHICH A TOKEN MAY
(*) BE DETERMINED.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME          I/O DESCRIPTION
(*) =====
(*) TOKEN          I/O INPUT  = TOKEN TYPE FROM PREVIOUS CALL
(*)                  OR INITIALIZATION FLAG
(*)                  OUTPUT = CURRENT TOKEN TYPE
(*) TOKEN_VALUE    0  CURRENT TOKEN VALUE
(*) TOKEN_LOCATION 0  START LOCATION OF TOKEN IN REPORT LINE
(*) TOKEN_LENGTH   0  LENGTH OF TOKEN IN REPORT LINE
(*) REPORT1        I/O
(*)
(*) $COMMONS:
(*) NONE
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL SEGMENT SUBPROGRAM
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*) DDNAMES USED WITH STANDARD FILES:
(*) SOURCE : THE INPUT STREAM OF CHARACTERS
(*) REPORT1 : FORMATTED REPORT OF THE INPUT
(*)
(*) $EXECUTION PROCEDURE:
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*)
(*) $PROCESSING DESCRIPTION:
(*) ?
(*)
(*) $COMMENTS:
(*) ?
(*)
(*) $CHANGE CONTROL:
(*)
(*) ORIGINATED: 18 MAR 87, G. A. WHITE
(*)
```

CI PS560240032U
April 1990

(* REVISED: 7 DEC 87, G. A. WHITE, ADD PARAMETERS FOR *)
(* LOCATION AND LENGTH OF TOKEN IN REPORT LINE. *)
(* *)
(* END %INCLUDE LEXICAL *****)

```
(* %INCLUDE LMEM23 *)
(**)
PROCEDURE LMEM23(VAR MESS      : MESSAGE;
                  VAR MEMBERS  : T_ARRAYTV;
                  VAR SIZE     : INTEGER;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE LIST MEMBERS (LMEM23) PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MEMBERS   I    THE ARRAY OF MEMBERS TO DISPLAY
(*   SIZE      I    THE SIZE OF THE ARRAY OF MEMBERS
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE LIST MEMBERS PANEL (LMEM23) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE MCREATE *)
(**)
PROCEDURE MCREATE(VAR MESS      : MESSAGE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*           DISPLAYS THE CREATE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE PANEL (MCREATE) BY MAKING ISPLNK CALLS.
(*   THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE MFILMOD *)
(**)
PROCEDURE MFILMOD(VAR MESS      : MESSAGE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*           DISPLAYS THE FILE/RETRIEVE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE FILE MODEL PANEL (MFILMOD) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```



```
(* %INCLUDE MINCLUD *)
(**)
PROCEDURE MINCLUD(VAR MESS      : MESSAGE;
                  VAR REPORT_TYPE : OPERATIONS;
                  VAR MEMBERS     : T_ARRAYTV;
                  VAR SIZE        : INTEGER;
                  VAR MEMBER      : T_NAME;
                  VAR NEXT_OP     : OPERATIONS;
                  VAR RR          : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE LIST OF SUBSCHEMAS
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   REPORT_TYPE I   THE TYPE OF REPORT TO BE GENERATED
(*   MEMBERS     I   THE ARRAY OF MEMBERS TO SELECT FROM
(*   SIZE        I   THE SIZE OF THE ARRAY OF MEMBERS
(*   MEMBER      O   THE MEMBER SELECTED
(*   NEXT_OP     O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR          O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE SUBSCHEMA PANEL (MINCLUD) (FROM WHICH A SUB-
(*   SCHEMA MAY BE SELECTED) BY MAKING ISPLNK CALLS.  THE OPTION
(*   CHOSEN IS TRANSLATED INTO AN ENUMERATED TYPE.
(*
(*
```

CI PS560240032U
April 1990

(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```

(* %INCLUDE MMAIN *)
(**)
  PROCEDURE MMAIN(VAR MESS      : MESSAGE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*               DISPLAYS THE MAIN MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   MESS           I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP        O   ENUMERATED TYPE INDICATING THE NEXT
(*                       OPERATION
(*   RR             O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                       IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE MAIN MENU PANEL (MMAIN) BY MAKING ISPLNK CALLS.
(*   THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*

```

```
(* %INCLUDE MNEWMOD *)
(**)
PROCEDURE MNEWMOD(VAR MESS      : MESSAGE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*           DISPLAYS THE FILE/RETRIEVE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE NEW MODEL PANEL (MNEWMOD) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```

(* BEGIN %INCLUDE MQBHALL *****)
(*)
PROCEDURE MQBHALL ( CONST NO_OF_KINDS : INTEGER;
                    VAR  NO_OF_ENTRY  : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     PRINT ALL ENTITIES IN THE MODEL
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ==  =====
(*)     NO_OF_KINDS    I    NUMBER OF KINDS IN THE MODEL
(*)     NO_OF_ENTRY    I/O  NUMBER OF ENTITIES PRINTED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH LIST OF ENTITIES IN THE MODEL
(*)     PRINT ENTITY
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQBHALL *****)

```

```
(* BEGIN %INCLUDE MQBHATT *****)
(*)
PROCEDURE MQBHATT ( CONST KIND      : INTEGER;
                    CONST INST_NO   : INTEGER;
                    VAR  NO_OF_ENTRY : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     PRINT INDIVIDUAL INSTANCES OF A SPECIFIC KIND
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ==  =====
(*)     KIND           I    A KIND NUMBER OF ENTITY
(*)     INST_NO        I    INSTANCE NUMBER
(*)     NO_OF_ENTRY    I/O  NUMBER OF ENTITIES PRINTED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     PRINT ADB, CONSTITUENT LIST, AND USER LIST.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQBHATT *****)
```

```

(*) BEGIN %INCLUDE MQBHATTS *****
(*)
PROCEDURE MQBHATTS ( CONST KIND      : INTEGER;
                     VAR  NO_OF_ENTRY : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     PRINT ALL ENTITIES OF A SPECIFIC KIND
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     KIND           I   A KIND NUMBER OF ENTITY
(*)     NO_OF_ENTRY    I/O  NUMBER OF ENTITIES PRINTED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH LIST OF ENTITIES OF A SPECIFIC KIND
(*)     PRINT ADB, CONSTITUENT LIST, AND USER LIST
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQBHATTS *****

```

```

(*) BEGIN %INCLUDE MQBHENT *****
(*)
PROCEDURE MQBHENT ( CONST MEMBER      : T_CHAR56;
                    VAR  NO_OF_ENTRY  : INTEGER;
                    VAR  ACTION       : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DISPLAY BATCH ENTITY MENU ( SELECT TO PRINT ALL ENTITIES OF
(*)     A SPECIFIC KIND OR TO PRINT INDIVIDUAL INSTANCES OF A
(*)     SPECIFIC KIND )
(*)
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     =====
(*)     MEMBER        I    ENTITY OF A SPECIFIC KIND
(*)     NO_OF_ENTRY   I/O  NUMBER OF ENTITIES PRINTED
(*)     ACTION        O    ENUMERATED TYPE INDICATING THE NEXT
(*)                       OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     DISPLAY BATCH ENTITY MENU
(*)     SELECT TO PRINT ALL ENTITIES OF A SPECIFIC KIND OR
(*)     TO PRINT INDIVIDUAL INSTANCES OF A SPECIFIC KIND
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQBHENT *****

```



```
(* BEGIN %INCLUDE MQBHMAIN *****)
(*)
PROCEDURE MQBHMAIN ( CONST MEMBERLIST : T_MEMBER;
                     CONST NO_OF_KINDS : INTEGER;
                     VAR  ACTION       : OPERATIONS );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*)   DISPLAY BATCH MAIN MENU ( SELECT TO PRINT ALL ENTITIES IN (*)
(*)   THE MODEL OR TO PRINT ALL ENTITIES OF A SPECIFIC KIND ) (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME           I/O  DESCRIPTION (*)
(*)   ====          ===  ===== (*)
(*)   MEMBERLIST     I    LIST OF ENTITIES IN THE MODEL (*)
(*)   NO_OF_KINDS    I    NUMBER OF KINDS IN THE MODEL (*)
(*)   ACTION         O    ENUMERATED TYPE INDICATING THE NEXT (*)
(*)                   OPERATION (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)   NAME/VALUE INTERFACE (*)
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)   DISPLAY BATCH MAIN MENU (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)   REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) (*)
(*)   ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE MQBHMAIN *****)
```

```

(*) BEGIN %INCLUDE MQCLMU *****
(*)
PROCEDURE MQCLMU ( CONST LIST_OF_CNSTS : LISTKEY;
                  CONST CL_NAME       : T_CL_NAME;
                  CONST NO_OF_CL      : INTEGER;
                  VAR  MEMBER         : T_CHAR56;
                  VAR  ACTION         : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DISPLAY A LIST OF CONSTITUENTS FOR A SPECIFIC KIND
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ==  =====
(*)     LIST_OF_CNSTS   I    POINTER TO CONSTITUENT LIST
(*)     CL_NAME         I    POINTER ATTRIBUTE NAME
(*)     NO_OF_CL        I    NUMBER OF CONSTITUENT LIST
(*)     MEMBER          0    SELECTED CONSTITUENT FOR DEFINITIONS
(*)     ACTION          0    ENUMERATED TYPE TO INDICATE THE NEXT
(*)                        OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     ?
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQCLMU *****

```

```

(* BEGIN %INCLUDE MQGETVAL *****)
(*)
PROCEDURE MQGETVAL ( CONST DATA_TYPE      : INTEGER;
                     CONST SIZE            : INTEGER;
                     CONST ATTRIBUTE_VALUE  : T_ATTRIBUTE_VALUE;
                     VAR  VAL              : STRING(16) );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*)   CONVERT ATTRIBUTE VALUE TO A STRING VALUE (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME          I/O  DESCRIPTION (*)
(*)   ====          ==  ===== (*)
(*)   DATA_TYPE     I    TYPE OF THE VALUE (*)
(*)   SIZE           I    SIZE OF THE VALUE (*)
(*)   ATTRIBUTE_VALUE I    DEPENDS ON THE TYPE (*)
(*)   VAL            0    STRING VALUE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)   NAME/VALUE INTERFACE (*)
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)   ? (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)   REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) (*)
(*)   ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE MQGETVAL *****)

```

```
(* BEGIN %INCLUDE MQGTDEFN *****)
(*)
PROCEDURE MQGTDEFN ( CONST KIND          : INTEGER;
                     CONST ENTITY_KEY    : ENTIKEY;
                     VAR  MEMBERLIST     : T_MEMBERLIST;
                     VAR  TOTAL_MEMBER   : INTEGER;
                     VAR  CL_NAME        : T_CL_NAME;
                     VAR  NO_OF_CL       : INTEGER );

    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     GET ENTITY DEFINITIONS OF A SPECIFIC KIND
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     KIND          I    A KIND NUMBER OF ENTITY
(*)     ENTITY_KEY    I    POINTER TO ENTITY INSTANCE
(*)     MEMBERLIST    0    LIST OF ATTRIBUTE NAMES WITH A VALUE
(*)     TOTAL_MEMBER  0    TOTAL NUMBER OF MEMBERS IN THE LIST
(*)     CL_NAME       0    LIST OF ATTRIBUTE NAMES IN THE
(*)                     CONSTITUENT LIST
(*)     NO_OF_CL      0    TOTAL NUMBER OF CONSTITUENT LIST
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     OBTAIN ENTITY DEFINITIONS FROM THE DATA DICTIONARY
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 10 DECEMBER, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQGTDEFN *****)
```

```

(*) BEGIN %INCLUDE MQIAATT *****
(*)
PROCEDURE MQIAATT ( CONST KIND      : T_CHAR48;
                    CONST INST_NO   : INTEGER;
                    VAR  TEMP_MEMBER : T_CHAR56;
                    VAR  ACTION      : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     PRINT INDIVIDUAL INSTANCES OF A SPECIFIC KIND
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME      I/O  DESCRIPTION
(*)     ====      ==  =====
(*)     KIND      I    A KIND NUMBER OF ENTITY
(*)     INST_NO   I    INSTANCE NUMBER
(*)     TEMP_MEMBER  O   SELECTED ENTITY
(*)     ACTION    O    ENUMERATED TYPE INDICATING THE NEXT
(*)                   OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     ?
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQIAATT *****

```

```
(* BEGIN %INCLUDE MQIAENT *****)
(*)
PROCEDURE MQIAENT ( CONST MEMBER      : T_CHAR56;
                   VAR  ACTION       : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DISPLAY INTERACTIVE ENTITY MENU
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     MEMBER        I    ENTITY OF A SPECIFIC KIND
(*)     ACTION        O    ENUMERATED TYPE INDICATING THE NEXT
(*)                      OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     ?
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQIAENT *****)
```

```

(*) BEGIN %INCLUDE MQIAMAIN *****
(*)
PROCEDURE MQIAMAIN ( CONST MEMBERLIST : T_MEMBER;
                     CONST NO_OF_KINDS : INTEGER;
                     VAR  ACTION       : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DISPLAY MAIN INTERACTIVE MENU
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====          ===  =====
(*)     MEMBERLIST     I    LIST OF ENTITIES IN THE MODEL
(*)     NO_OF_KINDS    I    NUMBER OF KINDS IN THE MODEL
(*)     ACTION         O    ENUMERATED TYPE INDICATING THE NEXT
(*)                      OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     DISPLAY INTERACTIVE MAIN MENU
(*)     SELECT AN ENTITY FOR THE DEFINITIONS
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQIAMAIN *****

```

```

(*) BEGIN %INCLUDE MQNCLMU *****
(*)
PROCEDURE MQNCLMU ( VAR ACTION    : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DISPLAY A MENU INDICATING NO CONSTITUENTS FOR A SPECIFIC
(*)     ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     ACTION        0    ENUMERATED TYPE INDICATING THE NEXT
(*)                      OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     ?
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQNCLMU *****

```



```
(* BEGIN %INCLUDE MQNUSRMU *****)
(*)
PROCEDURE MQNUSRMU ( VAR ACTION : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION: (*)
(*)     DISPLAY A MENU INDICATING NO USERS FOR A SPECIFIC ENTITY (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)     NAME           I/O  DESCRIPTION (*)
(*)     ====           ==  ===== (*)
(*)     ACTION         0    ENUMERATED TYPE INDICATING THE NEXT (*)
(*)                      OPERATION (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)     NAME/VALUE INTERFACE (*)
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)     ? (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) (*)
(*)     ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE MQNUSRMU *****)
```

```
(* %INCLUDE MQUDVR *)
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS IS THE MAINLINE PROGRAM WHICH DRIVES THE MODEL QUERY *)
(* UTILITY OF THE SCHEMA MANAGER. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NONE *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* DDNAMES USED WITH STANDARD FILES: *)
(* DDFILE = THE DATA DICTIONARY FILE DEFINITIONS. *)
(* *)
(* DDINX = THE DATA DICTIONARY FILE INDEX. *)
(* *)
(* FT08F001 = PART MODEL MASTER FILE (PID). *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* BEGIN %INCLUDE MQUSRMU ***** .***** *)
(*)
PROCEDURE MQUSRMU ( CONST LIST_OF_USERS : LISTKEY;
                    VAR  TEMP_MEMBER   : T_CHAR56;
                    VAR  ACTION        : OPERATIONS );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DISPLAY LIST OF USERS FOR A SPECIFIC ENTITY
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ==  =====
(*)     LIST_OF_USERS   I    LIST OF USERS FOR A SPECIFIC ENTITY
(*)     TEMP_MEMBER     0    SELECTED USER FOR THE DEFINITIONS
(*)     ACTION          0    ENUMERATED TYPE INDICATING THE NEXT
(*)                        OPERATION
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     ?
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 15 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE MQUSRMU ***** .***** *)
```

```
(* %INCLUDE MREPORT *)
(**)
PROCEDURE MREPORT(VAR MESS      : MESSAGE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*       DISPLAYS THE REPORT MENU
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*       NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REPORT MENU (MREPORT) BY MAKING ISPLNK CALLS.
(*   THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE MREVIEW *)
(**)
PROCEDURE MREVIEW(VAR MESS      : MESSAGE;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*       DISPLAYS THE REVIEW MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW PANEL (MREVIEW) BY MAKING ISPLNK CALLS.
(*   THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE MUPDATE *)
(**)
  PROCEDURE MUPDATE(VAR MESS      : MESSAGE;
                    VAR NEXT_OP   : OPERATIONS;
                    VAR RR        : RET_REC);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE:
(*           DISPLAYS THE UPDATE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE UPDATE PANEL (MUPDATE) BY MAKING ISPLNK CALLS.
(*   THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```

(*) BEGIN %INCLUDE NVRTVRS *****
(*)
PROCEDURE NVRTVRS ( CONST KIND          : INTEGER;
                   VAR  RUNTIME         : T_RUN_TIME;
                   VAR  RUNTIME_SIZE    : INTEGER;
                   VAR  RETURN_CODE     : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     RETRIEVE ENTITY DEFINITIONS FROM THE FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME      I/O  DESCRIPTION
(*)     ====      ==  =====
(*)     KIND      I    THE KIND NUMBER OF THE ENTITY
(*)                DEFINITION TO BE READ
(*)     RUNTIME    O    RUN-TIME SUBSCHEMA WHICH CONTAINS
(*)                THE ENTITY DEFINITION, ALONG WITH
(*)                ANY ENUMERATION VALUES, ANY ARRAY
(*)                INFORMATIONS, AND CONSTITUENT LIST
(*)                INFORMATIONS, IN A COMPACTED FORM.
(*)     RUNTIME_SIZE  O  THE NUMBER OF BYTES ACTUALLY REQUIRED
(*)                FOR THE COMPACTED RUN-TIME SUBSCHEMA.
(*)     RETURN_CODE  O  RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH INXFILE
(*)     IF KIND FOUND IN INXFILE THEN
(*)     LOOP THROUGH DATAFILE
(*)     IF KIND FOUND IN DATAFILE THEN
(*)     STORE ENTITY DEFINITION IN TEMPORARY WORK AREA
(*)     END IF
(*)     END IF
(*)     END LOOP
(*)     STORE ENTITY DEFINITION INTO RUN-TIME SUBSCHEMA
(*)     STORE SIZE OF ENTITY DEFINITION INTO RUN-TIME SUBSCHEMA
(*)

```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) ORIGINATED: 21 OCTOBER 1986, M. H. CHOI, DBMA *)
(*) *)
(*) END %INCLUDE NVRTVRS *****)
```



```
(* BEGIN %INCLUDE PHALFLD *****)
(*)
PROCEDURE PHALFLD ( VAR   FIELD_KEY       : ENTKEY;
                   CONST OFFSET_LIST      : T_OFFSET_LIST;
                   CONST OFFSET_LIST_COUNT : INTEGER;
                   VAR   S_ROOT           : T_STRUC_POINTER;
                   VAR   ARRAY_TABLE_POSITION : INTEGER;
                   VAR   CL_POSITION       : INTEGER;
                   VAR   ENUM_TABLE_POSITION : INTEGER;
                   VAR   IRC               : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   PHYSICALIZE THE ATTRIBUTE OF AN ENTITY
(*)   ( ASSIGN ATTRIBUTE SIZE AND LOCATION )
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   FIELD_KEY      I    FIELD KEY TO BE PHYSICALIZE
(*)   OFFSET_LIST     I    LIST OF ATTRIBUTES OF AN ENTITY
(*)                   ACCORDING TO BOUNDARY ALIGNMENT
(*)   OFFSET_LIST_COUNT I  NUMBER OF ATTRIBUTES IN THE LIST
(*)   S_ROOT          I/O  LIST OF STRUCTURE ATTRIBUTES
(*)   ARRAY_TABLE_POSIT 0  NUMBER OF ARRAYS IN THE ENTITY
(*)   CL_POSITION      0  NUMBER OF POINTERS IN THE ENTITY
(*)   ENUM_TABLE_POSIT 0  NUMBER OF ENUMERATIONS IN THE ENTITY
(*)   IRC              0  RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)   LOOP THROUGH ATTRIBUTES IN THE LIST OF BOUNDARY ALIGNMENT
(*)   IF ATTRIBUTE NAME IS IN THE LIST THEN
(*)       OBTAIN SIZE AND OFFSET FROM THE LIST
(*)       PHYSICALIZE THE ATTRIBUTE
(*)   END IF
(*)   END LOOP
(*)
(*) $COMMENTS:
(*)
```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* REVISED: 15 JANUARY 1988, M. H. CHOI, DBMA *)
(* INCORPORATED THE STRUCTURE DATA TYPE *)
(* ORIGINATED: 26 NOVEMBER 1986, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PHALFLD *****)

```
(* BEGIN %INCLUDE PHALST *****)
(*)
PROCEDURE PHALST ( CONST ENTITY_KEY      : ENTKY;
                   CONST STRUCTURE_NAME  : T_NAME;
                   CONST S_ROOT          : T_STRUC_POINTER;
                   VAR  ARRAY_TABLE_POSITION : INTEGER;
                   VAR  CL_POSITION       : INTEGER;
                   VAR  ENUM_TABLE_POSITION : INTEGER;
                   VAR  IRC               : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*)     PHYSICALIZE THE STRUCTURE ATTRIBUTE OF AN ENTITY (*)
(*)     ( ASSIGN ATTRIBUTE SIZE AND LOCATION ) (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)     NAME          I/O  DESCRIPTION (*)
(*)     ====          ===  ===== (*)
(*)     ENTITY_KEY     I    ENTITY KEY TO BE PHYSICALIZE (*)
(*)     STRUCTURE_NAME I    NAME OF THE STRUTURE ATTRIBUTE (*)
(*)     S_ROOT         I/O  LIST OF STRUCTURE ATTRIBUTES (*)
(*)     ARRAY_TABLE_POSIT 0  NUMBER OF ARRAYS IN THE ENTITY (*)
(*)     CL_POSITION     0  NUMBER OF POINTERS IN THE ENTITY (*)
(*)     ENUM_TABLE_POSIT 0  NUMBER OF ENUMERATIONS IN THE ENTITY (*)
(*)     IRC             0  RETURN_CODE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)     LOOP THROUGH ATTRIBUTES IN THE LIST OF BOUNDARY ALIGNMENT (*)
(*)     IF ATTRIBUTE NAME IS IN THE LIST THEN (*)
(*)         OBTAIN SIZE AND OFFSET FROM THE LIST (*)
(*)         PHYSICALIZE THE ATTRIBUTE (*)
(*)     END IF (*)
(*)     END LOOP (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)     ORIGINATED: 15 JANUARY 1988, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE PHALST *****)
```

```

(* BEGIN %INCLUDE PHBNST ***** *)
(*)
PROCEDURE PHBNST ( CONST NAME           : T_NAME;
                   CONST ALIGNMENT      : T_ALIGN;
                   CONST SIZE           : INTEGER;
                   VAR  DW_ROOT         : T_DW_POINTER;
                   VAR  FW_ROOT         : T_FW_POINTER;
                   VAR  HW_ROOT         : T_HW_POINTER;
                   VAR  BY_ROOT         : T_BY_POINTER;
                   VAR  PNTR_ROOT       : T_PNTR_POINTER;
                   VAR  IRC              : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   BUILD A LIST OF ATTRIBUTES IN THE STRUCTURE OF AN ENTITY
(*)   ACCORDING TO BOUNDARY ALIGNMENT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME           I/O  DESCRIPTION
(*)   ====          ===  =====
(*)   NAME           I    NAME OF STRUCTURE ATTRIBUTE
(*)   ALIGNMENT      I    ALIGNMENT OF STRUCTURE
(*)   SIZE           I    SIZE OF THE STRUCTURE
(*)   DW_ROOT        O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                       DOUBLE WORD ALIGNMENT
(*)   FW_ROOT        O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                       FULL WORD ALIGNMENT
(*)   HW_ROOT        O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                       HALF WORD ALIGNMENT
(*)   BY_ROOT        O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                       BYTES
(*)   PNTR_ROOT      O    LIST OF POINTER ATTRIBUTES
(*)   IRC            O    RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)   PROCEDURE (1) : DW_ALIGNMENT
(*)   BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY
(*)   REQUIRE FOR DOUBLE WORD ALIGNMENT
(*)
(*)

```

CI PS560240032U
April 1990

```
(*  PROCEDURE (2) : FW_ALIGNMENT *)
(*  BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*  REQUIRE FOR FULL WORD ALIGNMENT *)
(*  *)
(*  PROCEDURE (3) : HW_ALIGNMENT *)
(*  BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*  REQUIRE FOR HALF WORD ALIGNMENT *)
(*  *)
(*  PROCEDURE (4) : BY_ALIGNMENT *)
(*  BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*  REQUIRE FOR BYTES *)
(*  *)
(*  PROCEDURE (5) : PNTR_ALIGNMENT *)
(*  BUILD A LIST OF POINTER ATTRIBUTES OF AN ENTITY *)
(*  *)
(*  $COMMENTS: *)
(*  *)
(*  $CHANGE CONTROL: *)
(*  REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*  ORIGINATED: 15 JANUARY 1988, M. H. CHOI, DBMA *)
(*  *)
(* END %INCLUDE PHBNST *****)
```

```
(* BEGIN %INCLUDE PHBYFPOS *****)
(*)
PROCEDURE PHBYFPOS ( VAR ORDER_INDEX : INTEGER;
                    VAR ORDER_REC : T_GLOBAL_FIELD;
                    VAR ARRAY_TABLE_POSITION : INTEGER;
                    VAR ENUM_TABLE_POSITION : INTEGER;
                    VAR CL_POSITION : INTEGER;
                    VAR OFFSET_LIST_COUNT : INTEGER;
                    VAR OFFSET_LIST : T_OFFSET_LIST;
                    VAR IRC : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION:
(*) PHYSICALIZE THE ATTRIBUTES THAT SPECIFIED THE FIELD
(*) POSITION ORDER ( DETERMINE ATTRIBUTE SIZE AND LOCATION )
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*) ORDER_INDEX I NUMBER OF ATTRIBUTES THAT SPECIFIED
(*) THE ORDER
(*) ORDER_REC I LIST OF ATTRIBUTES THAT SPECIFIED
(*) THE ORDER
(*) ARRAY_TABLE_POS O NUMBER OF ARRAYS
(*) ENUM_TABLE_POSR O NUMBER OF ENUMERATIONS
(*) CL_POSITION O NUMBER OF POINTERS
(*) OFFSET_LIST_COUNT O NUMBER OF ATTRIBUTES IN THE LIST
(*) OFFSET_LIST O LIST OF ATTRIBUTES OF AN ENTITY
(*) IRC O RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*) SORT ATTRIBUTES BY THE FIELD POSITION NUMBER
(*) LOOP THROUGH THE ORDER LIST
(*) PHYSICALIZE THE ATTRIBUTE
(*) END LOOP
(*)
(*) $COMMENTS:
(*)
```

CI PS560240032U

April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 14 OCTOBER 1986, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PHBYFPOS *****)

```
(* BEGIN %INCLUDE PHDECBYT *****)
(*)
PROCEDURE PHDECBYT ( CONST CNST_ADB      : ENTBLOCK;
                     CONST NAME         : T_NAME;
                     CONST ARRAY_LENGTH : INTEGER;
                     VAR  DW_ROOT       : T_DW_POINTER;
                     VAR  FW_ROOT       : T_FW_POINTER;
                     VAR  HW_ROOT       : T_HW_POINTER;
                     VAR  BY_ROOT       : T_BY_POINTER;
                     VAR  PNTR_ROOT     : T_PNTR_POINTER;
                     VAR  IRC           : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   TRANSLATE DECIMAL DIGIT PRECISION INTO BYTE PRECISION AND
(*)   BUILD A LIST OF ATTRIBUTES OF AN ENTITY ACCORDING TO
(*)   BOUNDARY ALIGNMENT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   CNST_ADB      I    ADB OF CONSTITUENT
(*)   NAME          I    ATTRIBUTE NAME
(*)   ARRAY_LENGTH  I    NUMBER OF ARRAY DIMENSIONS
(*)   DW_ROOT       0    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   DOUBLE WORD ALIGNMENT
(*)   FW_ROOT       0    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   FULL WORD ALIGNMENT
(*)   HW_ROOT       0    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   HALF WORD ALIGNMENT
(*)   BY_ROOT       0    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   BYTES
(*)   PNTR_ROOT     0    LIST OF POINTER ATTRIBUTES
(*)   IRC           0    RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CASE ATTRIBUTE TYPE OF
(*)   INTEGER : CASE DECIMAL DIGIT OF INTEGER SIZE OF
(*)           1, 2 : BYTES_PRECISION = 1
```



```

(*)          SIZE = BYTES_PRECISION * ARRAY_LENGTH*)
(*)          BY_ALIGNMENT, PROCEDURE (4)          *)
(*)
(*)          3, 4 : BYTES_PRECISION = 2          *)
(*)          SIZE = BYTES_PRECISION * ARRAY_LENGTH*)
(*)          HW_ALIGNMENT, PROCEDURE (3)          *)
(*)
(*)          5, 6, : BYTES_PRECISION = 4          *)
(*)          7, 8, SIZE = BYTES_PRECISION * ARRAY_LENGTH*)
(*)          9      FW_ALIGNMENT, PROCEDURE (2)    *)
(*)
(*)          REAL      : CASE DECIMAL DIGIT OF REAL SIZE OF      *)
(*)          1,2,3: BYTES_PRECISION = 4          *)
(*)          4,5,6 SIZE = BYTES_PRECISION * ARRAY_LENGTH*)
(*)          7      FW_ALIGNMENT, PROCEDURE (2)    *)
(*)
(*)          8, 9, 10, 11,          *)
(*)          12, 13, 14, 15,          *)
(*)          16 : BYTES_PRECISION = 8          *)
(*)          SIZE = BYTES_PRECISION * ARRAY_LENGTH *)
(*)          DW_ALIGNMENT, PROCEDURE (1)          *)
(*)
(*)          STRING : BYTES_PRECISION = DECIMAL SIZE OF STRING *)
(*)          SIZE = BYTES_PRECISION * ARRAY_LENGTH          *)
(*)          BY_ALIGNMENT, PROCEDURE (4)          *)
(*)
(*)          LOGICAL,          *)
(*)          ENUMERATION : BYTES_PRECISION = 1          *)
(*)          SIZE = BYTES_PRECISION * ARRAY_LENGTH          *)
(*)          BY_ALIGNMENT, PROCEDURE (4)          *)
(*)
(*)          POINTER : BYTES_PRECISION = 0          *)
(*)          SIZE = BYTES_PRECISION * ARRAY_LENGTH          *)
(*)          PNTR_ALIGNMENT, PROCEDURE (5)          *)
(*)
(*)          END CASE          *)
(*)          PROCEDURE (1) : DW_ALIGNMENT          *)
(*)          BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*)          REQUIRE FOR DOUBLE WORD ALIGNMENT          *)
(*)
(*)          PROCEDURE (2) : FW_ALIGNMENT          *)
(*)          BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*)          REQUIRE FOR FULL WORD ALIGNMENT          *)
(*)
(*)          PROCEDURE (3) : HW_ALIGNMENT          *)
(*)          BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*)          REQUIRE FOR HALF WORD ALIGNMENT          *)
(*)

```

CI PS560240032U
April 1990

```
(*  PROCEDURE (4) : BY_ALIGNMENT *)
(*  BUILD A LIST OF ATTRIBUTES ( NAME AND SIZE ) OF AN ENTITY *)
(*  REQUIRE FOR BYTES *)
(*  *)
(*  PROCEDURE (5) : PNTR_ALIGNMENT *)
(*  BUILD A LIST OF POINTER ATTRIBUTES OF AN ENTITY *)
(*  *)
(*  $COMMENTS: *)
(*  *)
(*  $CHANGE CONTROL: *)
(*  REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*  ORIGINATED: 25 AUGUST 1986, M. H. CHOI, DBMA *)
(*  *)
(*  END %INCLUDE PHDECBYT *****)
```

```
(* BEGIN %INCLUDE PHENTITY *****)
(*)
PROCEDURE PHENTITY ( VAR  ENTITY_KEY      : ENTKEY;
                     VAR  ARRAY_TABLE_POSITION : INTEGER;
                     VAR  ENUM_TABLE_POSITION : INTEGER;
                     VAR  CL_POSITION       : INTEGER;
                     VAR  OFFSET_LIST_COUNT : INTEGER;
                     VAR  OFFSET_LIST      : T_OFFSET_LIST;
                     VAR  ADB_SIZE         : INTEGER;
                     VAR  IRC              : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   PHYSICALIZE THE ENTITY DEFINITIONS OF THE SUBSCHEMA
(*)   ( DETERMINE ATTRIBUTE SIZE AND LOCATION )
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ENTITY_KEY     I    ENTITY KEY OF DEFINITION TO BE
(*)                   PHYSICALIZE
(*)   ARRAY_TABLE_POSIT 0    NUMBER OF ARRAYS
(*)   ENUM_TABLE_POSIT  0    NUMBER OF ENUMERATIONS
(*)   CL_TABLE_POSIT    0    NUMBER OF POINTERS
(*)   OFFSET_LIST_COUNT 0    NUMBER OF ATTRIBUTES IN THE LIST
(*)   OFFSET_LIST       0    LIST OF ATTRIBUTES OF AN ENTITY
(*)   NEW_SIZE          0    SIZE DIFFERENCE BETWEEN THE PREVIOUS
(*)                   TOTAL GLOBAL SIZE AND THE NEW TOTAL
(*)                   GLOBAL SIZE
(*)   IRC              0    RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)   LOOP THROUGH THE ATTRIBUTES THAT SPECIFIED THE ORDER
(*)   PHYSICALIZE THE ATTRIBUTES
(*)   END LOOP
(*)   LOOP THROUGH THE ATTRIBUTES THAT DID NOT SPECIFIED THE ORDER*)
(*)   PHYSICALIZE THE ATTRIBUTES
(*)   END LOOP
(*)
```

CI PS560240032U
April 1990

(* \$COMMENTS: *)
(* *)
(* \$CHANGE CONTROL: *)
(* ORIGINATED: 4 SEPTEMBER 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PHENTITY *****)

```

(* BEGIN %INCLUDE PHGLOBAL ***** )
(*)
PROCEDURE PHGLOBAL ( VAR LIST_OF_GLOBALS : LISTKEY;
                     VAR ARRAY_TABLE_POSITION : INTEGER;
                     VAR ENUM_TABLE_POSITION : INTEGER;
                     VAR CL_TABLE_POSITION : INTEGER;
                     VAR OFFSET_LIST_COUNT : INTEGER;
                     VAR OFFSET_LIST : T_OFFSET_LIST;
                     VAR IRC : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*) PHYSICALIZE THE GLOBAL FIELDS OF THE SCHEMA (*)
(*) ( DETERMINE ATTRIBUTE SIZE AND LOCATION ) (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) SUBSCHEMA_KEY I SUBSCHEMA KEY (*)
(*) ARRAY_TABLE_POSIT 0 NUMBER OF ARRAYS (*)
(*) ENUM_TABLE_POSIT 0 NUMBER OF ENUMERATIONS (*)
(*) CL_TABLE_POSIT 0 NUMBER OF POINTERS (*)
(*) OFFSET_LIST_COUNT 0 NUMBER OF ATTRIBUTES IN THE LIST (*)
(*) OFFSET_LIST 0 LIST OF ATTRIBUTES OF AN ENTITY (*)
(*) ACCORDING TO BOUNDARY ALIGNMENT (*)
(*) IRC 0 RETURN_CODE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*) IF THERE ARE ANY GLOBAL FIELDS THEN (*)
(*) LOOP THROUGH EACH GLOBAL FIELDS (*)
(*) DETERMINE THE FIELD POSITION NUMBER (*)
(*) END LOOP (*)
(*) LOOP THROUGH THE FIELDS WITH FIELD POSITION (*)
(*) PHYSICALIZE THE FIELD BY FIELD POSITION NUMBER (*)
(*) END LOOP (*)
(*) LOOP THROUGH THE FIELDS WITHOUT THE FIELD POSITION (*)
(*) PHYSICALIZE THE FIELD BY BOUNDARY ALIGNMENT (*)
(*) END LOOP (*)
(*)

```

CI PS560240032U
April 1990

(* \$COMMENTS: *)
(* *)
(* \$CHANGE CONTROL: *)
(* ORIGINATED: 25 NOVEMBER 1986, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PHGLOBAL *****)

```

(* BEGIN %INCLUDE PHGTFLD *****)
(*)
PROCEDURE PHGTFLD ( CONST FIELD_KEY      : ENTKEY;
                   CONST FIELD_NAME     : T_NAME;
                   VAR  S_ROOT          : T_STRUC_POINTER;
                   VAR  DW_ROOT         : T_DW_POINTER;
                   VAR  FW_ROOT         : T_FW_POINTER;
                   VAR  HW_ROOT         : T_HW_POINTER;
                   VAR  BY_ROOT         : T_BY_POINTER;
                   VAR  PNTR_ROOT       : T_PNTR_POINTER;
                   VAR  NO_OF_DIMENS    : INTEGER;
                   VAR  LOW_BOUND       : T_LOW_BOUND;
                   VAR  HIGH_BOUND      : T_HIGH_BOUND;
                   VAR  CL_POSITION     : INTEGER;
                   VAR  IRC             : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION:
(*) DETERMINE THE BOUNDARY ALIGNMENT OF DIFFERENT DATA TYPES
(*) IN THE FIELD
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*) NAME      I/O  DESCRIPTION
(*) =====
(*) FIELD_KEY  I    FIELD KEY TO DETERMINE THE DATA TYPE
(*)              AND THE BOUNDARY ALIGNMENT
(*) FIELD_NAME I    ATTRIBUTE NAME
(*) DW_ROOT    O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)              DOUBLE WORD ALIGNMENT
(*) FW_ROOT    O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)              FULL WORD ALIGNMENT
(*) HW_ROOT    O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)              HALF WORD ALIGNMENT
(*) BY_ROOT    O    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)              BYTES
(*) PNTR_ROOT  O    LIST OF POINTER ATTRIBUTES
(*) NO_OF_DIMENS O  NUMBER OF ARRAY DIMENSIONS
(*) LOW_BOUND  O    LIST OF LOWER BOUNDS IN ARRAY
(*)              ATTRIBUTE
(*) HIGH_BOUND O    LIST OF HIGH BOUNDS IN ARRAY
(*)              ATTRIBUTE
(*) IRC        O    RETURN_CODE
(*)
(*) $COMMONS:
(*)

```

```
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* DETERMINE THE FIELD BOUNDARY ALIGNMENT *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* ORIGINATED: 26 NOVEMBER 1986, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PHGTFLD *****)
```



```
(* BEGIN %INCLUDE PHGTST *****)
(*)
PROCEDURE PHGTST ( CONST ENTITY_KEY      : ENTKEY;
                   CONST STRUCTURE_NAME  : T_NAME;
                   VAR  STRUCTURE_ROOT    : T_STRUC_POINTER;
                   VAR  STRUCTURE_SIZE    : INTEGER;
                   VAR  STRUCTURE_ALIGN   : T_ALIGN;
                   VAR  CL_POSITION       : INTEGER;
                   VAR  IRC               : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   DETERMINE THE BOUNDARY ALIGNMENT OF STRUCTURE DATA TYPE
(*)   IN THE FIELD
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   ENTITY_KEY     I    ENTITY KEY TO DETERMINE THE DATA
(*)                   TYPE AND THE BOUNDARY ALIGNMENT
(*)   STRUCTURE_NAME I    NAME OF STRUCTURE
(*)   STRUCTURE_ROOT O    LIST OF STRUCTURE ATTRIBUTES
(*)   STRUCTURE_SIZE O    SIZE OF THE STRUCTURE
(*)   STRUCTURE_ALIGN O    STRUCTURE ALIGNMENT
(*)   IRC            O    RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)   DETERMINE THE STRUCTURE FIELD BOUNDARY ALIGNMENT
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   ORIGINATED: 15 JANUARY 1988, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PHGTST *****)
```

April 1990

```

(*) BEGIN %INCLUDE PHPOSITN *****
(*)
PROCEDURE PHPOSITN ( CONST LIST_OF_FIELDS : LISTKEY;
                     VAR  ORDER_INDEX   : INTEGER;
                     VAR  ORDER_REC     : T_GLOBAL_FIELD;
                     VAR  UNORDER_LIST  : LISTKEY;
                     VAR  IRC           : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) DETERMINE THE FIELD POSITION ORDER
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*) =====
(*) LIST_OF_FIELDS I FIELDS TO DETERMINE THE FIELD
(*) POSITION ORDER
(*) ORDER_INDEX 0 NUMBER OF ATTRIBUTES THAT SPECIFIED
(*) THE FIELD POSITION NUMBER
(*) ORDER_REC 0 LIST OF ATTRIBUTES THAT SPECIFIED
(*) THE FIELD POSITION NUMBER
(*) UNORDER_LIST 0 LIST OF ATTRIBUTES THAT DID NOT
(*) SPECIFIED THE ORDER
(*) IRC 0 RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*) LOOP THROUGH THE ATTRIBUTES
(*) IF ATTRIBUTE DID NOT SPECIFIED THE POSITION THEN
(*) ATTACH TO THE UNORDER LIST
(*) ELSE
(*) ATTACH TO THE ORDER LIST
(*) END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*) ORIGINATED: 14 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PHPOSITN *****

```

```
(* BEGIN %INCLUDE PHSRTFLD *****)
(*)
PROCEDURE PHSRTFLD ( VAR DW_ROOT      : T_DW_POINTER;
                    VAR FW_ROOT      : T_FW_POINTER;
                    VAR HW_ROOT      : T_HW_POINTER;
                    VAR BY_ROOT      : T_BY_POINTER;
                    VAR PNTR_ROOT    : T_PNTR_POINTER;
                    VAR OFFSET_LIST  : T_OFFSET_LIST;
                    VAR TOTAL_COUNT  : INTEGER;
                    VAR CL_POSITION  : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     DETERMINE THE LOCATION OF ATTRIBUTES OF AN ENTITY ACCORDING
(*)     TO BOUNDARY ALIGNMENT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     DW_ROOT       I    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   DOUBLE WORD ALIGNMENT
(*)     FW_ROOT       I    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   FULL WORD ALIGNMENT
(*)     HW_ROOT       I    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   HALF WORD ALIGNMENT
(*)     BY_ROOT       I    LIST OF ATTRIBUTES WHICH REQUIRE FOR
(*)                   BYTES
(*)     PNTR_ROOT     I    LIST OF POINTER ATTRIBUTES
(*)     OFFSET_LIST   O    LIST OF ATTRIBUTES OF AN ENTITY
(*)                   ACCORDING TO BOUNDARY ALIGNMENT
(*)     TOTAL_COUNT   O    NUMBER OF ENTRIES IN THE LIST
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH EACH LIST OF ALIGNMENTS TABLE ACCORDING TO
(*)     BOUNDARY ALIGNMENT
(*)         (1) LIST OF DOUBLE WORD ALIGNMENT
(*)         (2) LIST OF FULL WORD ALIGNMENT
(*)         (3) LIST OF HALF WORD ALIGNMENT
(*)         (4) LIST OF BYTE ALIGNMENT
```

```
(*      STORE ATTRIBUTE NAME AND SIZE INTO LIST OF OFFSET-LIST      *)
(*      TABLE                                                         *)
(*      IF ATTRIBUTE NAME AND SIZE ARE THE FIRST ONE TO STORE         *)
(*      OFFSET = 12                                                     *)
(*      ELSE                                                            *)
(*      OFFSET = PREVIOUS OFFSET + PREVIOUS SIZE                       *)
(*      INCREMENT NUMBER OF ENTRIES IN THE TABLE                     *)
(*      END LOOP                                                         *)
(*      *)                                                              *)
(*      $COMMENTS:                                                       *)
(*      *)                                                              *)
(*      $CHANGE CONTROL:                                                 *)
(*      REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)              *)
(*      ORIGINATED: 18 SEPTEMBER 1986, M. H. CHOI, DBMA               *)
(*      *)                                                              *)
(*      END %INCLUDE PHSRTFLD *****)
```

```
(* BEGIN %INCLUDE PHSRTORD *****)
(*)
PROCEDURE PHSRTORD ( CONST ORDER_INDEX      : INTEGER;
                    VAR  ORDER_REC          : T_GLOBAL_FIELD );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*)     SORT ATTRIBUTES BY THE FIELD POSITION NUMBER (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)     NAME          I/O  DESCRIPTION (*)
(*)     ====          ==  ===== (*)
(*)     ORDER_INDEX   I    NUMBER OF ATTRIBUTES THAT SPECIFIED (*)
(*)                   THE FIELD POSITION NUMBER (*)
(*)     ORDER_REC     O    LIST OF ATTRIBUTES THAT SPECIFIED (*)
(*)                   THE FIELD POSITION NUMBER (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)     LOOP THROUGH LIST OF ATTRIBUTES THAT SPECIFIED THE FIELD (*)
(*)     POSITION NUMBER AND PUT THEM IN ASCENDING ORDER. (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)     ORIGINATED: 18 NOVEMBER 1986, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE PHSRTORD *****)
```

```

(* BEGIN %INCLUDE PHSUBTYP *****)
(*)
PROCEDURE PHSUBTYP ( CONST SUBTYPE_KEY      : ENTKY;
                     VAR  ARRAY_TABLE_POSITION : INTEGER;
                     VAR  ENUM_TABLE_POSITION : INTEGER;
                     VAR  CL_POSITION         : INTEGER;
                     VAR  OFFSET_LIST_COUNT  : INTEGER;
                     VAR  OFFSET_LIST        : T_OFFSET_LIST;
                     VAR  IRC                 : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*)   PHYSICALIZE THE SUPER TYPES (*)
(*)   ( DETERMINE ATTRIBUTE SIZE AND LOCATION ) (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME          I/O  DESCRIPTION (*)
(*)   ====          ==  ===== (*)
(*)   SUBSCHEMA_KEY  I    SUBSCHEMA KEY OF THE ENTITY (*)
(*)                   DEFINITIONS TO BE PHYSICALIZE (*)
(*)   ARRAY_TABLE_POSIT 0    NUMBER OF ARRAYS (*)
(*)   ENUM_TABLE_POSIT  0    NUMBER OF ENUMERATIONS (*)
(*)   CL_TABLE_POSIT    0    NUMBER OF POINTERS (*)
(*)   OFFSET_LIST_COUNT 0    NUMBER OF ATTRIBUTES IN THE LIST (*)
(*)   OFFSET_LIST       0    LIST OF ATTRIBUTES OF AN ENTITY (*)
(*)   NEW_SIZE          0    SIZE DIFFERENCE BETWEEN THE PREVIOUS (*)
(*)                   TOTAL GLOBAL SIZE AND THE NEW TOTAL (*)
(*)                   GLOBAL SIZE (*)
(*)   IRC              0    RETURN_CODE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)   PHYSICALIZE THE ATTRIBUTES WITHIN THE SUPERTYPE (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)   ORIGINATED: 4 SEPTEMBER 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE PHSUBTYP *****

```

```
(* BEGIN %INCLUDE PHWOFPOS *****)
(*)
PROCEDURE PHWOFPOS ( VAR UNORDER_LIST : LISTKEY;
                     VAR ARRAY_TABLE_POSITION : INTEGER;
                     VAR ENUM_TABLE_POSITION : INTEGER;
                     VAR CL_POSITION : INTEGER;
                     VAR OFFSET_LIST_COUNT : INTEGER;
                     VAR OFFSET_LIST : T_OFFSET_LIST;
                     VAR IRC : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) PHYSICALIZE THE ATTRIBUTES THAT DID NOT SPECIFIED THE
(*) FIELD POSITION NUMBER ( DETERMINE ATTRIBUTE SIZE AND
(*) LOCATION )
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*) UNORDER_LIST 0 LIST OF ATTRIBUTES WITHOUT THE FIELD
(*) POSITION NUMBER
(*)
(*) ARRAY_TABLE_POS 0 NUMBER OF ARRAYS
(*)
(*) ENUM_TABLE_POSR 0 NUMBER OF ENUMERATIONS
(*)
(*) CL_POSITION 0 NUMBER OF POINTERS
(*)
(*) OFFSET_LIST_COUNT 0 NUMBER OF ATTRIBUTES IN THE LIST
(*)
(*) OFFSET_LIST 0 LIST OF ATTRIBUTES OF AN ENTITY
(*)
(*) IRC 0 RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*) LOOP THROUGH THE ATTRIBUTES THAT DID NOT SPECIFIED THE
(*) FIELD POSITION NUMBER
(*) PHYSICALIZE THE ATTRIBUTE
(*) END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*) ORIGINATED: 14 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PHWOFPOS *****)
```

```
(* BEGIN %INCLUDE PHYSICAL *****)
(*)
PROCEDURE PHYSICAL ( VAR  SUBSCHEMA_KEY  : ENTKY;
                     VAR  IRC             : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     PHYSICALIZE THE ENTITY DEFINITIONS OF THE SUBSCHEMA
(*)     ( DETERMINE ATTRIBUTE SIZE AND LOCATION )
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     =====
(*)     SUBSCHEMA_KEY  I    SUBSCHEMA KEY OF THE ENTITY
(*)                     DEFINITIONS TO BE PHYSICALIZE
(*)     IRC            0    RETURN_CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)
(*) $PROCESSING DESCRIPTION:
(*)     PHYSICALIZE THE GLOBAL FIELDS
(*)     LOOP THROUGH LIST OF ENTITIES
(*)         CREATE A LIST OF SUPERTYPES INCLUSIVELY BY SUPERTYPE
(*)         KIND FOR SPECIFIC ENTITY
(*)     LOOP THROUGH LIST OF SUPERTYPES
(*)         PHYSICALIZE THE SUPERTYPES
(*)     END LOOP
(*)     PHYSICALIZE THE ATTRIBUTES
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: 9 SEPTEMBER 1987, M. H. CHOI, DBMA
(*)         ADDED A SUPERTYPE
(*)     ORIGINATED: 14 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PHYSICAL *****)
```



```

(* BEGIN %INCLUDE PSGTSM *****)
(*)
PROCEDURE PSGTSM ( CONST ENTITY_KEY      : ENIKEY;
                   VAR  RUN_TIME         : P_RUN_TIME;
                   VAR  RUN_TIME_SIZE    : INTEGER;
                   VAR  IRC               : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) BUILD RUN-TIME SUBSCHEMA FROM SCHEMA MODEL FOR
(*) THE PHYSICAL SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*) NAME          I/O  DESCRIPTION
(*) =====
(*) ENTITY_KEY    I    KEY FROM SCHEMA MODEL WHICH THE
(*)                TRANSLATION WILL BE PERFORMED.
(*) IRC           O    RETURN CODE
(*)                = 0  SUCCESS
(*)                > 0  CRITICAL ERROR:
(*) RUN_TIME      O    RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*)                ENTITY DEFINITION, ALONG WITH ANY
(*)                ENUMERATION VALUES, CONSTITUENT LIST,
(*)                AND ARRAY INFORMATION, IN A COMPACTED
(*)                FORM.
(*) RUN_TIME_SIZE O    THE NUMBER OF BYTES ACTUALLY REQUIRED
(*)                FOR THE COMPACTED RUN-TIME SUBSCHEMA.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) RUN-TIME SUBSCHEMA
(*) CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*) TRANSLATE SCHEMA MODEL ENTRY INTO ENTITY ATTRIBUTES
(*) AND ENUMERATION VALUES AND ARRAY INFORMATION
(*) IF THERE WERE ANY ENUMERATION ATTRIBUTES THEN
(*) CALCULATE THE STARTING POSITION OF THE ENUMERATION
(*) INDEX TABLE AND STORE INTO RUN-TIME SUBSCHEMA
(*) DETERMINE THE ACTUAL SIZE OF THE ENUMERATION INDEX TABLE
(*) COPY ENUMERATION INDEX TABLE INFORMATION INTO RUN-TIME
(*) SUBSCHEMA

```

```
(*      ENDIF *)
(*      IF THERE WERE ANY ENUMERATION ATTRIBUTES THEN *)
(*          CALCULATE THE STARTING POSITION OF THE ENUMERATION *)
(*          VALUE TABLE AND STORE INTO RUN-TIME SUBSCHEMA *)
(*          DETERMINE THE ACTUAL SIZE OF THE ENUMERATION VALUE TABLE *)
(*          COPY THE ENUMERATION VALUES INTO THE RUN-TIME SUBSCHEMA *)
(*      ENDIF *)
(*      IF THERE WERE ANY ARRAY ATTRIBUTES THEN *)
(*          CALCULATE THE STARTING POSITION OF THE ARRAY INDEX TABLE *)
(*          AND STORE INTO RUN-TIME SUBSCHEMA *)
(*          DETERMINE THE ACTUAL SIZE OF THE ARRAY INDEX TABLE *)
(*          COPY ARRAY TABLE INDEX INFORMATION INTO RUN-TIME SUBSCHEMA *)
(*      ENDIF *)
(*      IF THERE WERE ANY ARRAY ATTRIBUTES THEN *)
(*          CALCULATE THE STARTING POSITION OF THE ARRAY LIST TABLE *)
(*          AND STORE INTO RUN-TIME SUBSCHEMA *)
(*          DETERMINE THE ACTUAL SIZE OF THE ARRAY LIST TABLE *)
(*          COPY ARRAY LIST INFORMATION INTO RUN-TIME SUBSCHEMA *)
(*      ENDIF *)
(*      CALCULATE THE SIZE OF THE RUN-TIME SUBSCHEMA *)
(*      *)
(*      $COMMENTS: *)
(*      *)
(*      $CHANGE CONTROL: *)
(*          ORIGINATED: 25 JANUARY 1988, M. H. CHOI, DBMA *)
(*      *)
(*      END %INCLUDE PSGTSM ***** *)
```

```

(* BEGIN %INCLUDE PSMASKND *****)
(*)
PROCEDURE PSMASKND ( VAR ENTITY          : P_SCHEMA );
SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   INSERT THE MODEL ACCESS SOFTWARE(MAS) ATTRIBUTES ( KIND,
(*)   LENGTH, SYSUSE ) INTO A RUN-TIME SUBSCHEMA FOR THE
(*)   PHYSICAL SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ENTITY        0   RUN-TIME SUBSCHEMA ENTITY DEFINITION.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   INSERT KIND, LENGTH, SYSUSE ATTRIBUTES INTO A RUN-TIME
(*)   SUBSCHEMA ENTITY DEFINITION.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)   ORIGINATED: 25 JANUARY 1988, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PSMASKND *****)

```

CI PS560240032U
April 1990

```
(* BEGIN %INCLUDE PSORDER *****)
(*)
PROCEDURE PSORDER ( CONST RUNTIME      : P_RUN_TIME;
                    VAR  PS_ORDER      : T_PS_ORDER );
    SUBPROGRAM;

(*)
(*) $FUNCTION: (*)
(*)     DETERMINE THE PHYSICAL SCHEMA ORDER OF ENTITY DEFINITION (*)
(*)     BY ITS OFFSET FOR THE PHYSICAL SCHEMA REPORT. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)     NAME          I/O  DESCRIPTION (*)
(*)     ====          ==  ===== (*)
(*)     RUNTIME       I    CONTAINS THE ENTITY DEFINITION (*)
(*)     PS_ORDER      0    LIST OF ATTRIBUTES IN PHYSICAL (*)
(*)                     ORDER (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)     LOOP THROUGH THE NUMBER OF ATTRIBUTES IN THE DEFINITION (*)
(*)     INSERT THE OFFSET IN THE PHYSICAL SCHEMA ORDER TABLE (*)
(*)     END LOOP (*)
(*)     LOOP THROUGH THE NUMBER OF ATTRIBUTES IN THE DEFINITION (*)
(*)     IF CURRENT OFFSET GREATER THAN NEXT OFFSET THEN (*)
(*)     SWITCH CURRENT OFFSET WITH NEXT OFFSET (*)
(*)     END IF (*)
(*)     END LOOP (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)     ORIGINATED: 26 JANUARY 1988, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE PSORDER *****)
```

```
(* BEGIN %INCLUDE PSRABNDS *****)
(*)
PROCEDURE PSRABNDS ( VAR   PSRDATA           : TEXT;
                     CONST NO_OF_DIMEN       : INTEGER;
                     CONST STARTING_ARRAY_POSITION : INTEGER;
                     VAR   POINTER           : T_VARIANT_POINTER;
                     VAR   ENUM_INDEX        : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION: (*)
(*)     WRITE LOW-BOUND AND UPPER-BOUND FOR THE ARRAY ATTRIBUTE (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)     NAME           I/O  DESCRIPTION (*)
(*)     ====           ==  ===== (*)
(*)     PSRDATA        0    PHYSICAL SCHEMA REPORT SEQUENTIAL FILE(*)
(*)     NO_OF_DIMENS    I    NUMBER OF ARRAY DIMENSIONS (*)
(*)     STARTING_ARRAY_PO I  STARTING POSITION IN THE ARRAY TABLE (*)
(*)     POINTER         I    POINTER TO ARRAY INFORMATION TABLE (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)     LOOP THROUGH THE NUMBER OF DIMENSIONS (*)
(*)     WRITE LOW-BOUND AND UPPER-BOUND (*)
(*)     END LOOP (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)     ORIGINATED: 24 MARCH 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE PSRABNDS *****)
```

```
(* BEGIN %INCLUDE PSRADB ******)
(*)
PROCEDURE PSRADB ( VAR   PSRDATA           : TEXT;
                   CONST RUNTIME          : P_RUN_TIME;
                   CONST ENTRY            : INTEGER;
                   CONST PS_ORDER         : T_PS_ORDER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   WRITE THE BASIC RECORD OF AN ENTITY TO A PHYSICAL SCHEMA
(*)   REPORT FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O DESCRIPTION
(*)   ====           === =====
(*)   PSRDATA         0  PHYSICAL SCHEMA REPORT SEQUENTIAL FILE*)
(*)   RUNTIME         I  CONTAINS THE ENTITY DEFINITION
(*)   ENTRY           I  ENTRY ORDER IN THE DEFINITION
(*)   PS_ORDER        I  LIST OF ATTRIBUTES IN PHYSICAL ORDER
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   WRITE THE BASIC ATTRIBUTES ( INTEGER, REAL, STRING, LOGICAL)*)
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PSRADB ******)
```

```

(* BEGIN %INCLUDE PSRARRAY ***** )
(*)
PROCEDURE PSRARRAY ( VAR   PSRDATA           : TEXT;
                     CONST RUNTIME          : P_RUN_TIME;
                     CONST ENTRY            : INTEGER;
                     VAR   CL_HEADING_FLAG  : BOOLEAN;
                     VAR   ENUM_HEADING_FLAG : BOOLEAN;
                     CONST PS_ORDER         : T_PS_ORDER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   WRITE THE ARRAY ATTRIBUTE OF AN ENTITY TO A SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME           I/O DESCRIPTION
(*)   ====           == =====
(*)   PSRDATA        O   PHYSICAL SCHEMA REPORT TEXT FILE
(*)   RUNTIME         I   CONTAINS THE ENTITY DEFINITION
(*)   ENTRY           I   ENTRY ORDER IN THE DEFINITION
(*)   CL_HEADING_FLAG I   FLAG TO DETERMINE WHETHER THE HEADING
(*)                       HAS BEEN WRITTEN
(*)   PS_ORDER        I   LIST OF ATTRIBUTES IN PHYSICAL ORDER
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   OBTAIN THE NUMBER OF DIMENSIONS
(*)   OBTAIN THE STARTING POSITION OF ARRAY TABLE
(*)   CASE DATA TYPE OF
(*)     IN-ADB : WRITE BASIC DEFINITION
(*)             PSRABNDS ( EXTERNAL SUBPROGRAM TO WRITE LOW-BOUND*
(*)                     AND UPPER-BOUND )
(*)     IN-CL  : WRITE BASIC DEFINITION
(*)             PSRCL ( EXTERNAL SUBPROGRAM FOR CONSTITUENT LIST)*
(*)             PSRABNDS ( EXTERNAL SUBPROGRAM TO WRITE LOW-BOUND*
(*)                     AND UPPER-BOUND )
(*)   END CASE
(*)
(*) $COMMENTS:
(*)

```

CI PS560240032U
April 1990

```
(* $CHANGE CONTROL: *)
(* ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PSRARRAY *****)
```



```

(* BEGIN %INCLUDE PSRCL ***** *)
(*)
PROCEDURE PSRCL ( VAR   PSRDATA           : TEXT;
                  CONST RUNTIME          : P_RUN_TIME;
                  CONST ENTRY            : INTEGER;
                  VAR   CL_HEADING_FLAG   : BOOLEAN;
                  CONST NO_OF_DIMEN       : INTEGER;
                  CONST STARTING_POSITION : INTEGER;
                  VAR   ARRAY_POINTER     : T_VARIANT_POINTER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   WRITE THE CONSTITUENT REFERENCES OF AN ENTITY TO A
(*)   SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O DESCRIPTION
(*)   ====           === =====
(*)   PSRDATA         0   PHYSICAL SCHEMA REPORT TEXT FILE
(*)   RUNTIME          I   CONTAINS THE ENTITY DEFINITION
(*)   ENTRY            I   ENTRY ORDER IN THE DEFINITION
(*)   CL_HEADING_FLAG  I   FLAG TO DETERMINE WHETHER THE HEADING
(*)                       HAS BEEN WRITTEN
(*)   NO_OF_DIMEN      I   NUMBER OF ARRAY DIMENSIONS
(*)   STARTING_ARRAY_PO I   STARTING POSITION IN THE ARRAY TABLE
(*)   ARRAY_POINTER     I   POINTER TO ARRAY TABLE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   IF NOT ARRAY ATTRIBUTE THEN
(*)     WRITE BASIC DEFINITION
(*)   END IF
(*)   OBTAIN THE NUMBER OF ELIGIBLE KINDS
(*)   OBTAIN STARTING POSITION OF CONSTITUENT LIST TABLE
(*)   LOOP THROUGH THE NUMBER OF ELIGIBLE KINDS
(*)     WRITE ELIGIBLE KIND IN THE CONSTITUENT LIST TABLE
(*)   END LOOP
(*)

```

I

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PSRCL *****)
```

I

```

(* BEGIN %INCLUDE PSRENUM *****
*)
PROCEDURE PSRENUM ( VAR   PSRDATA           : TEXT;
                   CONST RUNTIME           : P_RUN_TIME;
                   CONST ENTRY             : INTEGER;
                   CONST PS_ORDER          : T_PS_ORDER;
                   VAR   ENUM_HEADING_FLAG : BOOLEAN;
                   CONST ENUM_TABLE_INDEX  : INTEGER;
                   CONST NO_OF_VALUES      : INTEGER );
    SUBPROGRAM;
(*
*) $FUNCTION:
(*   WRITE THE ENUMERATION ATTRIBUTE OF AN ENTITY TO A
(*   SEQUENTIAL FILE
(*
*) $DESCRIPTION OF ARGUMENTS:
(*   NAME           I/O DESCRIPTION
(*   ====           === =====
(*   PSRDATA        0   PHYSICAL SCHEMA REPORT SEQUENTIAL FILE*)
(*   RUNTIME        I   CONTAINS THE ENTITY DEFINITION
(*   ENTRY          I   ENTRY ORDER IN THE DEFINITION
(*   PS_ORDER       I   LIST OF ATTRIBUTES IN PHYSICAL ORDER
(*   ENUM_HEADING_FLAG I FLAG TO DETERMINE WHETHER THE HEADING
(*                       HAS BEEN WRITTEN
(*
*) $COMMONS:
(*
*) $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
*) $EXECUTION PROCEDURE:
(*   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*
*) $PROCESSING DESCRIPTION:
(*   WRITE BASIC DEFINITION
(*   OBTAIN THE NUMBER OF ENUMERATION VALUES
(*   OBTAIN THE STARTING POSITION OF ENUMERATION VALUE TABLE
(*   LOOP THROUGH THE NUMBER OF ENUMERATION VALUES
(*       WRITE THE ENUMERATION VALUE FROM THE TABLE
(*   END LOOP
(*
*) $COMMENTS:
(*
*) $CHANGE CONTROL:
(*   ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA
(*
*) END %INCLUDE PSRENUM *****

```

```
(* BEGIN %INCLUDE PSREPORT *****)
(*)
PROCEDURE PSREPORT ( VAR  SUBSCHEMA_KEY : ENTKEY;
                     VAR  IRC           : RET_REC );
    SUBPROGRAM;
(*)
(*) $FUNCTION: (*)
(*) FILE PHYSICAL SCHEMA REPEORT TO SEQUENTIAL FILE (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) SUBSCHEMA_KEY I (*)
(*) IRC 0 (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*) CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*) ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE PSREPORT *****)
```

```

(* BEGIN %INCLUDE PSRHEAD *****)
(*)
PROCEDURE PSRHEAD ( VAR   PSRDATA           : TEXT;
                   CONST RUNTIME          : P_RUN_TIME;
                   CONST PAGE_NO         : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE PHYSICAL SCHEMA REPORT HEADING
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME           I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     PSRDATA        O   PHYSICAL SCHEMA REPORT SEQUENTIAL FILE*)
(*)     RUNTIME        I   CONTAINS THE ENTITY DEFINITION
(*)     PAGE_NO        I   NUMBER OF PAGE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     WRITE THE HEADING
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PSRHEAD *****)

```

```
(* BEGIN %INCLUDE PSRINDEX *****)
(*)
PROCEDURE PSRINDEX ( VAR   PSRDATA           : TEXT;
                     CONST PAGE_NO          : INTEGER;
                     CONST ENTITY_INDEX      : T_ENTITY_INDEX;
                     CONST LIST_OF_ENTITIES  : LISTKEY );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     WRITE THE TABLE OF CONTENTS FOR THE PHYSICAL SCHEMA REPORT
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME      I/O  DESCRIPTION
(*)     ====      ==  =====
(*)     PSRDATA    O   PHYSICAL SCHEMA REPORT TEXT FILE
(*)     PAGE_NO    I   PAGE NUMBER
(*)     ENTITY_INDEX I   ENTITY NAME AND THE KIND NUMBER
(*)     LIST_OF_ENTITIES I LIST OF ENTITY KEYS TO SORT LATER
(*)                      BY THE ENTITY NAME
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     WRITE THE TABLE OF CONTENTS IN ORDER OF ENTITY KIND
(*)     WRITE THE TABLE OF CONTENTS IN ORDER OF ENTITY NAME
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 09 MARCH 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PSRINDEX *****)
```

```

(*) BEGIN %INCLUDE PSRNAME *****
(*)
PROCEDURE PSRNAME ( VAR PSRDATA          : TEXT;
                   CONST RUNTIME         : P_RUN_TIME;
                   CONST ENTRY           : INTEGER;
                   VAR  GLOBAL_FLAG      : BOOLEAN;
                   VAR  INHERITED_FLAG   : BOOLEAN;
                   VAR  LOCAL_FLAG       : BOOLEAN );
    SUBPROGRAM;
(*)
(*) $FUNCTION:
(*)     INDICATE WHETHER THE ATTRIBUTE IS GLOBAL, INHERITED, OR
(*)     LOCAL FOR THE PHYSICAL SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     PSRDATA       O    PHYSICAL SCHEMA REPORT SEQUENTIAL FILE*)
(*)     RUNTIME       I    CONTAINS THE ENTITY DEFINITION
(*)     ENTRY         I    ENTRY ORDER IN THE DEFINITION
(*)     PS_ORDER      I    LIST OF ATTRIBUTES IN PHYSICAL ORDER
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     WRITE THE BASIC ATTRIBUTES ( INTEGER, REAL, STRING, LOGICAL)*)
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 26 JANUARY 1988, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PSRNAME *****

```

```

(* BEGIN %INCLUDE PSRSTC *****
(*)
PROCEDURE PSRSTC ( VAR   PSRDATA           : TEXT;
                   CONST RUNTIME           : P_RUN_TIME;
                   CONST PS_ORDER          : T_PS_ORDER;
                   VAR   ENTRY             : INTEGER;
                   VAR   CL_HEADING_FLAG   : BOOLEAN;
                   VAR   ENUM_HEADING_FLAG : BOOLEAN;
                   VAR   LIST_OF_STRUCTURE: T_ENUMERATION;
                   VAR   S_COUNT           : INTEGER );
      SUBPROGRAM;
(*)
(*) $FUNCTION:
(*)   WRITE THE STRUCTURE ATTRIBUTE OF AN ENTITY TO A PHYSICAL
(*)   SCHEMA REPORT FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O DESCRIPTION
(*)   =====
(*)   PSRDATA        0   PHYSICAL SCHEMA REPORT SEQUENTIAL FILE*)
(*)   RUNTIME         I   CONTAINS THE ENTITY DEFINITION
(*)   ENTRY           I   ENTRY ORDER IN THE DEFINITION
(*)   PS_ORDER        I   LIST OF ATTRIBUTES IN PHYSICAL ORDER
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   WRITE THE STRUCTURE ATTRIBUTE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   ORIGINATED: 15 JANUARY 1988, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE PSRSTC *****

```



```

(*) BEGIN %INCLUDE PSTRGF *****
(*)
PROCEDURE PSTRGF ( VAR ENTITY      : P_SCHEMA;
                   VAR ENUM       : T_ENUM_COMPACTOR;
                   VAR ENUM_INDEX : T_ENUM_INX_COMPACTOR;
                   VAR ARRAY_LIST : T_ARRAY_LIST_COMPACTOR;
                   VAR ARRAY_INDEX: T_ARRAY_INX_COMPACTOR;
                   VAR CL_INDEX   : T_CL_INX_COMPACTOR;
                   VAR CL_LIST    : T_CL_KINDS_COMPACTOR;
                   VAR IRC        : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   TRANSLATE THE GLOBAL FIELDS INTO A RUN-TIME SUBSCHEMA
(*)   FOR THE PHYSICAL SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O  DESCRIPTION
(*)   ====      ==  =====
(*)   ENTITY     0   RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*)               ENTITY DEFINITION
(*)   ENUM        0   ENUMERATION VALUES
(*)   ENUM_INDEX  0   ENUMERATION INDEX TABLE
(*)   ARRAY_LIST  0   LOWER BOUND AND ARRAY SIZE
(*)   ARRAY_INDEX 0   ARRAY INDEX TABLE
(*)   CL_INDEX    0   CONSTITUENT INDEX TABLE
(*)   CL_LIST     0   CONSTITUENT KINDS
(*)   IRC         0   RETURN CODE
(*)               = 0  SUCCESS
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   RUN-TIME SUBSCHEMA
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   INITIALIZE TABLE INDEXES
(*)   MAKE A LIST OF GLOBAL KINDS
(*)   LOOP THROUGH A LIST OF GLOBAL KINDS
(*)   TRANSLATE GLOBAL FIELD INTO A RUN-TIME SUBSCHEMA
(*)

```

CI PS560240032U
April 1990

(* \$COMMENTS: *)
(* *)
(* \$CHANGE CONTROL: *)
(* ORIGINATED: 25 JANUARY 1988, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE PSTRGF *****)

```
(* BEGIN %INCLUDE PSTSRM *****)
(*)
PROCEDURE PSTSRM ( CONST ENTITY_KEY      : LISTKEY;
                   VAR  NAME_TYPE        : CHAR;
                   VAR  ENTITY            : P_SCHEMA;
                   VAR  ENUM              : T_ENUM_COMPACTOR;
                   VAR  ENUM_INDEX        : T_ENUM_INX_COMPACTOR;
                   VAR  ARRAY_LIST        : T_ARRAY_LIST_COMPACTOR;
                   VAR  ARRAY_INDEX       : T_ARRAY_INX_COMPACTOR;
                   VAR  CL_INDEX          : T_CL_INX_COMPACTOR;
                   VAR  CL_LIST           : T_CL_KINDS_COMPACTOR;
                   VAR  IRC               : RET_REC );
SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) TRANSLATE A SCHEMA MODEL ENTRY INTO A RUN-TIME SUBSCHEMA
(*) ENTITY, ENUMERATION TABLE AND ARRAY INFO TABLE FOR
(*) THE PHYSICAL SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*) NAME          I/O DESCRIPTION
(*)
(*) =====
(*) ARRAY_INDEX    0  RUN-TIME SUBSCHEMA ARRAY TABLE INDEX
(*)                INFORMATION.
(*) ARRAY_LIST     0  RUN-TIME SUBSCHEMA ARRAY TABLE
(*)                AND COMPACTION INFORMATION.
(*) ENUM           0  RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                AND COMPACTION INFORMATION.
(*) ENUM_INDEX     0  RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                INDEX INFORMATION.
(*) ENTITY         0  RUN-TIME SUBSCHEMA ENTITY DEFINITION.
(*) ENTITY_KEY     I  KEY FROM SCHEMA MODEL WHICH THE
(*)                TRANSLATION WILL BE PERFORMED.
(*) IRC           0  RETURN CODE
(*)                = 0 SUCCESS
(*)                > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) NAME/VALUE INTERFACE
(*) CALLED FROM THE NAME/VALUE INTERFACE
(*)
```

April 1990

```

(*) $PROCESSING DESCRIPTION: (*)
(*)   OBTAIN ENTITY NAME AND KIND FROM SCHEMA MODEL (*)
(*)   STORE ENTITY NAME AND KIND INTO RUN-TIME SUBSCHEMA (*)
(*)   LOOP THROUGH SCHEMA MODEL ENTRIES (*)
(*)   OBTAIN ATTRIBUTE ENTRY FROM SCHEMA MODEL (*)
(*)   CASE DATA TYPE OF (*)
(*)     INTEGER, REAL, STRING, LOGICAL (*)
(*)       : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) (*)
(*)     POINTER :   CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) (*)
(*)     ARRAY   :           ARRAY_ATTRIBUTE, PROCEDURE (3) (*)
(*)     DEFINED_TYPE :   DEFINED_TYPE_ATTRIBUTE, PROCEDURE (4) (*)
(*)     OTHERWISE :   ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' (*)
(*)   ENDCASE (*)
(*)   ENDLOOP (*)

(*) PROCEDURE (1) : APPLICATION_DATA_BLOCK_ATTRIBUTE (*)
(*)   STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY (*)
(*)
(*) PROCEDURE (2) : CONSTITUENT_LIST_ATTRIBUTE (*)
(*)   OBTAIN CONSTITUENT LIST POSITION FROM SCHEMA MODEL (*)
(*)   STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY (*)
(*)
(*) PROCEDURE (3) : ARRAY_ATTRIBUTE (*)
(*)   DETERMINE THE NUMBER OF ARRAY DIMENSIONS (*)
(*)   STORE ARRAY INFORMATION INTO RUN-TIME SUBSCHEMA (*)
(*)   STORE TABLE INDEX POSITION FOR ARRAY LIST TABLE AND THE (*)
(*)     NUMBER OF DIMENSIONS INTO ARRAY INDEX TABLE (*)
(*)   CALCULATE TOTAL SIZE OF THE ARRAY AND STORE INTO ARRAY (*)
(*)     INDEX TABLE (*)
(*)   FOR THE NUMBER OF ARRAY DIMENSIONS (*)
(*)     CALCULATE THE SIZE OF EACH ARRAY (*)
(*)     STORE SIZE AND LOW-BOUND INTO ARRAY LIST TABLE (*)
(*)   END LOOP (*)
(*)
(*) PROCEDURE (4) : DEFINED_TYPE_ATTRIBUTE (*)
(*)   OBTAIN DATA TYPE FOR DEFINED TYPE ATTRIBUTE IN SCHEMA MODEL (*)
(*)   CASE DATA_TYPE OF (*)
(*)     INTEGER, REAL, STRING, LOGICAL (*)
(*)       : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) (*)
(*)     ENUMERATION :   ENUMERATION_ATTRIBUTE, PROCEDURE (5) (*)
(*)     POINTER :   CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) (*)
(*)     ARRAY   :           ARRAY_ATTRIBUTE, PROCEDURE (3) (*)
(*)     OTHERWISE :   ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' (*)
(*)   ENDCASE (*)
(*)

```

```
(*  PROCEDURE (5) : ENUMERATION_ATTRIBUTE *)
(*  STORE ATTRIBUTE DEFINITION FOR ENUMERATION TYPE *)
(*  OBTAIN NUMBER OF ENUMERATION VALUES FROM SCHEMA MODEL *)
(*  STORE NUMBER OF ENUMERATION VALUE IN ENUMERATION INDEX TABLE*)
(*  STORE ENUMERATION VALUE TABLE INDEX POSITION IN ENUMERATION *)
(*  INDEX TABLE *)
(*  LOOP THROUGH ENUMERATION VALUES *)
(*  OBTAIN ENUMERATION VALUE FROM SCHEMA MODEL *)
(*  STORE ENUMERATION VALUE IN ENUMERATION VALUE TABLE *)
(*  END LOOP *)
(*  *)
(*  $COMMENTS: *)
(*  *)
(*  $CHANGE CONTROL: *)
(*  REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*  ORIGINATED: 25 JANUARY 1988, M. H. CHOI, DBMA *)
(*  *)
(* END %INCLUDE PSTRSM *****)
```

```
(* BEGIN %INCLUDE PSTRST *****)
```

```
(*
PROCEDURE PSTRST ( CONST SUBTYPE_KEY      : ENTKEY;
                   VAR  ENTITY             : P_SCHEMA;
                   VAR  ENUM               : T_ENUM_COMPACTOR;
                   VAR  ENUM_INDEX         : T_ENUM_INX_COMPACTOR;
                   VAR  ARRAY_LIST         : T_ARRAY_LIST_COMPACTOR;
                   VAR  ARRAY_INDEX       : T_ARRAY_INX_COMPACTOR;
                   VAR  CL_INDEX           : T_CL_INX_COMPACTOR;
                   VAR  CL_LIST            : T_CL_KINDS_COMPACTOR;
                   VAR  IRC                : RET_REC );
SUBPROGRAM;
(*
(* $FUNCTION:
(*   TRANSLATE SUPER TYPE INTO A RUN-TIME SUBSCHEMA FOR
(*   THE PHYSICAL SCHEMA REPORT.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   SUBTYPE_KEY   I    KEY FROM SCHEMA MODEL WHICH THE
(*                   TRANSLATION WILL BE PERFORMED.
(*                   > 0  CRITICAL ERROR:
(*   ENTITY        O    RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*                   ENTITY DEFINITION
(*   ENUM          O    ENUMERATION VALUES
(*   ENUM_INDEX    O    ENUMERATION INDEX TABLE
(*   ARRAY_LIST    O    LOWER BOUND AND ARRAY SIZE
(*   ARRAY_INDEX   O    ARRAY INDEX TABLE
(*   CL_INDEX      O    CONSTITUENT INDEX TABLE
(*   CL_LIST       O    CONSTITUENT KINDS
(*   IRC           O    RETURN CODE
(*                   = 0  SUCCESS
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   RUN-TIME SUBSCHEMA
(*   CALLED FROM THE NAME/VALUE INTERFACE
(*)
```

CI PS560240032U
April 1990

```
(* $PROCESSING DESCRIPTION: *)
(*   MAKE A LIST OF SUPER TYPE   *)
(*   LOOP THROUGH A LIST OF SUPER TYPE   *)
(*   TRANSLATE SUPER TYPE INTO A RUN-TIME SUBSCHEMA   *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(*   ORIGINATED: 25 JANUARY 1988, M. H. CHOI, DBMA   *)
(* *)
(* END %INCLUDE PSTRTS ***** *)
```

```
(* BEGIN %INCLUDE PSTRTC *****)
(*)
PROCEDURE PSTRTC ( CONST ENTITY_KEY      : LISTKEY;
                   CONST NAME_TYPE       : CHAR;
                   VAR  S_OFFSET         : INTEGER;
                   VAR  ENTITY           : P_SCHEMA;
                   VAR  ENUM              : T_ENUM_COMPACTOR;
                   VAR  ENUM_INDEX        : T_ENUM_INX_COMPACTOR;
                   VAR  ARRAY_LIST        : T_ARRAY_LIST_COMPACTOR;
                   VAR  ARRAY_INDEX       : T_ARRAY_INX_COMPACTOR;
                   VAR  CL_INDEX          : T_CL_INX_COMPACTOR;
                   VAR  CL_LIST           : T_CL_KINDS_COMPACTOR;
                   VAR  IRC                : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   TRANSLATE A STRUCTURE ATTRIBUTE INTO A RUN-TIME SUBSCHEMA
(*)   ENTITY, ENUMERATION TABLE AND ARRAY INFO TABLE FOR
(*)   THE PHYSICAL SCHEMA REPORT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ARRAY_INDEX    0    RUN-TIME SUBSCHEMA ARRAY TABLE INDEX
(*)                   INFORMATION.
(*)   ARRAY_LIST      0    RUN-TIME SUBSCHEMA ARRAY TABLE
(*)                   AND COMPACTION INFORMATION.
(*)   ENUM            0    RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                   AND COMPACTION INFORMATION.
(*)   ENUM_INDEX      0    RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                   INDEX INFORMATION.
(*)   ENTITY          0    RUN-TIME SUBSCHEMA ENTITY DEFINITION.
(*)   ENTITY_KEY      I    KEY FROM SCHEMA MODEL WHICH THE
(*)                   TRANSLATION WILL BE PERFORMED.
(*)   IRC            0    RETURN CODE
(*)                   = 0 SUCCESS
(*)                   > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
```



```

(*) $PROCESSING DESCRIPTION: (*)
(*)   OBTAIN ENTITY NAME AND KIND FROM SCHEMA MODEL (*)
(*)   STORE ENTITY NAME AND KIND INTO RUN-TIME SUBSCHEMA (*)
(*)   LOOP THROUGH SCHEMA MODEL ENTRIES (*)
(*)   OBTAIN ATTRIBUTE ENTRY FROM SCHEMA MODEL (*)
(*)   CASE DATA TYPE OF (*)
(*)     INTEGER, REAL, STRING, LOGICAL (*)
(*)       : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) (*)
(*)     POINTER : CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) (*)
(*)     ARRAY   : ARRAY_ATTRIBUTE, PROCEDURE (3) (*)
(*)     DEFINED_TYPE : DEFINED_TYPE_ATTRIBUTE, PROCEDURE (4) (*)
(*)     OTHERWISE : ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' (*)
(*)   ENDCASE (*)
(*) ENDLOOP (*)

(*) PROCEDURE (1) : APPLICATION_DATA_BLOCK_ATTRIBUTE (*)
(*)   STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY (*)
(*)
(*) PROCEDURE (2) : CONSTITUENT_LIST_ATTRIBUTE (*)
(*)   OBTAIN CONSTITUENT LIST POSITION FROM SCHEMA MODEL (*)
(*)   STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY (*)
(*)
(*) PROCEDURE (3) : ARRAY_ATTRIBUTE (*)
(*)   DETERMINE THE NUMBER OF ARRAY DIMENSIONS (*)
(*)   STORE ARRAY INFORMATION INTO RUN-TIME SUBSCHEMA (*)
(*)   STORE TABLE INDEX POSITION FOR ARRAY LIST TABLE AND THE (*)
(*)     NUMBER OF DIMENSIONS INTO ARRAY INDEX TABLE (*)
(*)   CALCULATE TOTAL SIZE OF THE ARRAY AND STORE INTO ARRAY (*)
(*)   INDEX TABLE (*)
(*)   FOR THE NUMBER OF ARRAY DIMENSIONS (*)
(*)     CALCULATE THE SIZE OF EACH ARRAY (*)
(*)     STORE SIZE AND LOW-BOUND INTO ARRAY LIST TABLE (*)
(*)   END LOOP (*)
(*)
(*) PROCEDURE (4) : DEFINED_TYPE_ATTRIBUTE (*)
(*)   OBTAIN DATA TYPE FOR DEFINED TYPE ATTRIBUTE IN SCHEMA MODEL (*)
(*)   CASE DATA_TYPE OF (*)
(*)     INTEGER, REAL, STRING, LOGICAL (*)
(*)       : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) (*)
(*)     ENUMERATION : ENUMERATION_ATTRIBUTE, PROCEDURE (5) (*)
(*)     POINTER : CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) (*)
(*)     ARRAY   : ARRAY_ATTRIBUTE, PROCEDURE (3) (*)
(*)     OTHERWISE : ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' (*)
(*)   ENDCASE (*)
(*)
(*) PROCEDURE (5) : ENUMERATION_ATTRIBUTE (*)
(*)   STORE ATTRIBUTE DEFINITION FOR ENUMERATION TYPE (*)

```

```
(*      OBTAIN NUMBER OF ENUMERATION VALUES FROM SCHEMA MODEL      *)
(*      STORE NUMBER OF ENUMERATION VALUE IN ENUMERATION INDEX TABLE*)
(*      STORE ENUMERATION VALUE TABLE INDEX POSITION IN ENUMERATION *)
(*      INDEX TABLE                                                  *)
(*      LOOP THROUGH ENUMERATION VALUES                             *)
(*      OBTAIN ENUMERATION VALUE FROM SCHEMA MODEL                   *)
(*      STORE ENUMERATION VALUE IN ENUMERATION VALUE TABLE          *)
(*      END LOOP                                                       *)
(*      *)                                                             *)
(*      $COMMENTS:                                                     *)
(*      *)                                                             *)
(*      $CHANGE CONTROL:                                              *)
(*      REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)            *)
(*      ORIGINATED: 25 JANUARY 1988, M. H. CHOI, DBMA                *)
(*      *)                                                             *)
(*      END %INCLUDE PSTRTC *****)
```

```
(* %INCLUDE REARRAY *)
(**)
PROCEDURE REARRAY(VAR MESS      : MESSAGE;
                  VAR LBND      : CHAR8;
                  VAR HBND      : CHAR8;
                  VAR FTYPE     : CHAR12;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE REVIEW ARRAY PANEL.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   LBND      I   THE LOWER BOUND OF THE ARRAY
(*   HBND      I   THE UPPER BOUND OF THE ARRAY
(*   FTYPE     I   THE ARRAY TYPE
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*       NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW ARRAY PANEL (REARRAY) BY MAKING ISPLNK
(*   CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
```

CI PS560240032U
April 1990

(* \$COMMENTS:
(* NONE
(*
(* \$CHANGE CONTROL:
(*

*)
)
)
)
)
)

```
(* %INCLUDE RECLASS *)
(**)
PROCEDURE RECLASS(VAR MESS      : MESSAGE;
                  VAR NAME      : T_NAME;
                  VAR KNUM      : CHAR8;
                  VAR COMMENT   : CHAR150;
                  VAR CLAS      : T_ARRAYTV;
                  VAR ARRAY_SIZE : INTEGER;
                  VAR MEMBER     : T_NAME;
                  VAR NEXT_OP    : OPERATIONS;
                  VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE REVIEW CLASS PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      I   THE CLASS NAME
(*   KNUM      I   THE CLASS KIND NUMBER
(*   CLAS      I   THE ARRAY OF MEMBERS
(*   ARRAY_SIZE I   THE SIZE OF THE ARRAY OF MEMBERS
(*   MEMBER     O   THE MEMBER SELECTED
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
```

CI PS560240032U
April 1990

(* \$PROCESSING DESCRIPTION: *)
(* DISPLAY THE REVIEW CLASS PANEL (RECLASS) BY MAKING ISPLNK *)
(* CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(* TYPE. *)
(* *)
(* \$COMMENTS: *)
(* NONE *)
(* *)
(* \$CHANGE CONTROL: *)
(* *)

```
(* %INCLUDE REDEFTYP *)
(**)
PROCEDURE REDEFTYP(VAR MESS      : MESSAGE;
                   VAR NAME      : IDCHAR;
                   VAR FTYPE     : CHAR12;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE DEFINED TYPE REVIEW PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      O   THE NAME OF THE DEFINED TYPE ENTERED
(*   FTYPE     O   TYPES INTEGER,STRING,REAL...ETC.
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW DEFINED TYPE PANEL (REDEFTYP) BY MAKING
(*   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE REENTITY *)
(**)
PROCEDURE REENTITY(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR KNUM      : CHAR8;
                   VAR COMMENT   : CHAR150;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE REVIEW ENTITY MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME          I/O  DESCRIPTION
(*      ====          ==  =====
(*      MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      NAME          I    THE ENTITY NAME
(*      KNUM          I    THE ENTITY KIND NUMBER
(*      NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                       OPERATION
(*      RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                       IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE REVIEW ENTITY PANEL (REENTITY) BY MAKING ISPLNK
(*      CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*      TYPE.
(*
```


CI PS560240032U
April 1990

(* \$COMMENTS:
(* NONE
(*
(* \$CHANGE CONTROL:
(*

*)
)
)
)
)
)

```
(* %INCLUDE REENUM *)
(**)
PROCEDURE REENUM(VAR MESS      : MESSAGE;
                  VAR MEMBERS  : T_ARRAYID;
                  VAR SIZE     : INTEGER;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE REVIEW ENUMERATION MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MEMBERS   I    THE ARRAY OF MEMBERS TO DISPLAY
(*   SIZE      I    THE SIZE OF THE ARRAY OF NUMBERS
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW ENUMERATION MENU (REENUM) BY MAKING
(*   ISPLNK CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE REFIELD *)
(**)
```

```
PROCEDURE REFIELD(VAR MESS      : MESSAGE;
                  VAR NAME      : T_NAME;
                  VAR POS       : CHAR8;
                  (* VAR PURP    : CHAR8; *)
                  VAR REQD      : CHAR8;
                  (* VAR DEPD    : CHAR12; *)
                  VAR FTYPE     : CHAR12;
                  VAR FLDT      : CHAR9;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);
```

SUBPROGRAM;

```
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE :
(*               DISPLAYS THE REVIEW FIELD PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME          I    THE NAME OF THE ARRAY
(*   PURP          I    THE PURPOSE OF THE ARRAY
(*   REQD          I    THE REQUIREDNESS OF THE ARRAY
(*   DEPD          I    THE DEPENDENCE/INDEPENDENCE OF THE ARRAY
(*   FTYPE         I    THE TYPE OF ELEMENT STORED IN THE ARRAY
(*   FLDT          I    THE TYPE OF ARRAY (GLOBAL, STRUCTURE,
(*                           ENTITY)
(*   NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                           OPERATION
(*   RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                           IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*)
```

```
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* DDNAMES USED WITH STANDARD FILES: *)
(* NONE *)
(* $EXECUTION PROCEDURE: *)
(* SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* $PROCESSING DESCRIPTION: *)
(* DISPLAY THE REVIEW FIELD PANEL (REFIELD) BY MAKING ISPLNK *)
(* CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(* TYPE. *)
(* $COMMENTS: *)
(* NONE *)
(* $CHANGE CONTROL: *)
(*)
```

(* %INCLUDE REFIELD1 *)

(**)

```
PROCEDURE REFIELD1(VAR MESS      : MESSAGE;
                   VAR FIELD_TYPE : T_FIELDTYPE;
                   VAR NAME       : T_NAME;
                   VAR KNUM       : CHAR8;
                   VAR MEMBERS    : T_ARRAYID;
                   VAR SIZE       : INTEGER;
                   VAR FNAME      : T_NAME;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*)      THIS PROCEDURE : (*)
(*)      DISPLAYS THE REVIEW FIELD A MENU OR THE (*)
(*)      REVIEW FIELD B MENU (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)      NAME          I/O  DESCRIPTION (*)
(*)      ====          ==  ===== (*)
(*)      MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL (*)
(*)      FIELD_TYPE    I    THE TYPE OF FIELD TO BE REVIEWED (*)
(*)      NAME          I    THE ENTITY NAME (*)
(*)      KNUM          I    THE ENTITY KIND NUMBER (*)
(*)      MEMBERS       I    THE ARRAY OF MEMBERS TO SELECT FROM (*)
(*)      SIZE          I    THE SIZE OF THE ARRAY OF MEMBERS (*)
(*)      FNAME         O    THE MEMBER SELECTED (*)
(*)      NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT (*)
(*)                      OPERATION (*)
(*)      RR            O    INDICATES IF AN ERROR HAS OCCURRED AND, (*)
(*)                      IF ONE HAS, WHAT ROUTINE IT OCCURRED IN (*)
(*)
(*) $COMMONS: (*)
(*)      NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*)      LANGUAGE: IBM PASCAL (*)
(*)      HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)      DDNAMES USED WITH STANDARD FILES: (*)
(*)      NONE (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE (*)
(*)
```

CI PS560240032U
April 1990

```
(* $PROCESSING DESCRIPTION: *)
(*   DISPLAY THE REVIEW ENTITY PANEL (REFIELD1) BY MAKING ISPLNK *)
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*   TYPE. *)
(* *)
(* $COMMENTS: *)
(*   NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE REFIELD2 *)
(**)
```

```
PROCEDURE REFIELD2(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR POS       : CHAR8;
                   (* VAR PURP    : CHAR8; *)
                   VAR REQD      : CHAR8;
                   (* VAR DEPD    : CHAR12; *)
                   VAR COM       : CHAR50;
                   VAR FTYPE     : CHAR12;
                   VAR FLDT      : CHAR9;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);
```

SUBPROGRAM;

```
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS PROCEDURE : *)
(* DISPLAYS THE REVIEW FIELD PANEL *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* MESS I THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(* NAME I THE NAME OF THE ARRAY *)
(* POS I THE POSITION OF THE FIELD IN THE ADB *)
(* PURP I THE PURPOSE OF THE ARRAY *)
(* REQD I THE REQUIREDNESS OF THE ARRAY *)
(* DEPD I THE DEPENDENCE/INDEPENDENCE OF THE ARRAY *)
(* FTYPE I THE TYPE OF ELEMENT STORED IN THE ARRAY *)
(* FLDT I THE TYPE OF ARRAY (GLOBAL, STRUCTURE, *)
(* ENTITY) *)
(* NEXT_OP 0 ENUMERATED TYPE INDICATING THE NEXT *)
(* OPERATION *)
(* RR 0 INDICATES IF AN ERROR HAS OCCURRED AND, *)
(* IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
```

```

(*)      DDNAMES USED WITH STANDARD FILES:                      *)
(*)      NONE                                                    *)
(*)                                                                 *)
(*) $EXECUTION PROCEDURE:                                       *)
(*)      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE                *)
(*)                                                                 *)
(*) $PROCESSING DESCRIPTION:                                     *)
(*)      DISPLAY THE REVIEW FIELD PANEL (REFIELD2) BY MAKING ISPLNK *)
(*)      CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*)      TYPE.                                                    *)
(*)                                                                 *)
(*) $COMMENTS:                                                  *)
(*)      NONE                                                    *)
(*)                                                                 *)
(*) $CHANGE CONTROL:                                           *)
(*)                                                                 *)
(*)      REVISED: 09/28/87          C. H. MOHME                DBMA *)
(*)      INCORPORATED THE SUPERTYPE DATA TYPE.                  *)
(*)                                                                 *)
(*)      REVISED: 08/13/87          C. H. MOHME                DBMA *)
(*)      CHANGED PANEL OPTION NUMBERS; CHANGED FIELD ADB DATA; ADDED *)
(*)      LIST AND SET. NOTE: THE LIST AND SET DATA TYPES ARE IMPL- *)
(*)      MENTED IN THE SOFTWARE AS AN ARRAY. A FIELD WAS ADDED TO THE *)
(*)      ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL ARRAY, *)
(*)      LIST, OR SET.                                           *)
(*)                                                                 *)
(*)      REVISED: 07/02/87          C. H. MOHME                DBMA *)
(*)      CHANGED CURSOR POSITIONING.                               *)
(*)                                                                 *)
(*)      ORIGINATED: 07/07/86        C. H. MOHME                DBMA *)
(*)                                                                 *)
(*)-----*)
(*)                                                                 *)
(*)END-----*)
(*) END %INCLUDE REFIELD2 *)

```



```
(* %INCLUDE REFSUP *)
(**)
PROCEDURE REFSUP(VAR IRC          : RET_REC;
                 VAR TRANS_STACK : TRANSPTR;
                 VAR TOKEN_VALUE : T_TOKEN_VALUE);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     BATCH INTERFACE ROUTINE THAT ATTEMPTS TO RESOLVE A
(*     REFERENCE TO A SUPERTYPE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC           0    INTERNAL RETURN CODE
(*     TRANS_STACK   I/O  TRANSACTION STACK
(*     TOKEN_VALUE   I/O  TOKEN VALUE FROM BATCH INPUT
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE BATCH INPUT
(*
(* $PROCESSING DESCRIPTION:
(*
(*     INITIALIZE VARIABLES
(*     DETERMINE IF THE SUPERTYPE HAS ALREADY BEEN MODELED.
(*     IF SUPERTYPE MODELED, PUSH ITS KEY ONTO THE TRANSACTION STACK.
(*     IF SUPERTYPE NOT MODELED, PUSH UNRESOLVED TRANSACTION ONTO THE
(*     TRANSACTION STACK.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*     ORIGINATED: 09/29/87          C. H. MOHME          DBMA
(*
(*-----*)
(*END-----*)
(* END %INCLUDE REFSUP *)
```

```

(*) %INCLUDE REINTGR *)
(**)
PROCEDURE REINTGR(VAR MESS      : MESSAGE;
                  VAR PREC      : CHAR8;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*)   THIS FUNCTION:*)
(*)       DISPLAYS THE REVIEW INTEGER MENU*)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS:*)
(*)   NAME      I/O  DESCRIPTION*)
(*)   ====      ==  =====*)
(*)   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL*)
(*)   PREC       O   THE PRECISION OF THE INTEGER ENTERED*)
(*)   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT*)
(*)               OPERATION*)
(*)   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,*)
(*)               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN*)
(*)*)
(*) $COMMONS:*)
(*)   NONE*)
(*)*)
(*) $ENVIRONMENT:*)
(*)   LANGUAGE: IBM PASCAL*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381*)
(*)   DDNAMES USED WITH STANDARD FILES:*)
(*)   NONE*)
(*)*)
(*) $EXECUTION PROCEDURE:*)
(*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE*)
(*)*)
(*) $PROCESSING DESCRIPTION:*)
(*)   DISPLAY THE REVIEW INTEGER PANEL (REINTGR) BY MAKING ISPLNK*)
(*)   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED*)
(*)   TYPE.)*
(*)*)
(*) $COMMENTS:*)
(*)   NONE*)
(*)*)
(*) $CHANGE CONTROL:*)
(*)*)

```

```
(* %INCLUDE RELIST *)
(**)
PROCEDURE RELIST(VAR MESS      : MESSAGE;
                  VAR MIN      : CHAR8;
                  VAR MAX      : CHAR8;
                  VAR FTYPE    : CHAR12;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE REVIEW LIST PANEL.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MIN      I   THE MINIMUM NUMBER OF OCCURRENCES IN THE
(*               LIST
(*   MAX      I   THE MAXIMUM NUMBER OF OCCURRENCES IN THE
(*               LIST
(*   FTYPE    I   THE LIST TYPE
(*   NEXT_OP  O   ENUMERATED TYPE INDICATING THE NEXT
(*               OPERATION
(*   RR      O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW LIST PANEL (RELIST) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
```

CI PS560240032U
April 1990

(*	\$COMMENTS:	
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)
		*)

```
(* %INCLUDE REPNTNTR *)
(**)
PROCEDURE REPNTNTR(VAR MESS      : MESSAGE;
                   VAR MEMBERS   : T_ARRAYTV;
                   VAR ARRAY_SIZE : INTEGER;
                   VAR MEMBER     : T_NAME;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE REVIEW POINTER MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MEMBERS   I   THE ARRAY OF MEMBERS TO SELECT FROM
(*   ARRAY_SIZE I   THE SIZE OF THE ARRAY OF MEMBERS
(*   MEMBER     O   THE MEMBER SELECTED
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW POINTER PANEL (REPNTNTR) BY MAKING ISPLNK
(*   CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
```

CI PS560240032U
April 1990

(*	\$COMMENTS:	
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)
		*)

```
(* %INCLUDE RREAL *)
(**)
PROCEDURE RREAL(VAR MESS      : MESSAGE;
                 VAR SIZE     : CHAR8;
                 VAR NEXT_OP  : OPERATIONS;
                 VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE REVIEW REAL PANEL.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   SIZE      O   THE SIZE OF THE REAL ENTERED
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW REAL PANEL (RREAL) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
(* $COMMENTS:
(*   NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE RESET *)
(**)
PROCEDURE RESET(VAR MESS      : MESSAGE;
                VAR MIN       : CHAR8;
                VAR MAX       : CHAR8;
                VAR FTYPE     : CHAR12;
                VAR NEXT_OP   : OPERATIONS;
                VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE REVIEW SET PANEL.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MIN      I    THE MINIMUM NUMBER OF OCCURRENCES IN THE
(*               LIST
(*   MAX      I    THE MAXIMUM NUMBER OF OCCURRENCES IN THE
(*               LIST
(*   FTYPE     I    THE SET TYPE
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*               OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW SET PANEL (RESET) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE.
(*
```


CI PS560240032U
April 1990

(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```

(*) %INCLUDE RESTRING *)
(**)
PROCEDURE RESTRING(VAR MESS      : MESSAGE;
                   VAR SLEN      : CHAR8;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*)   THIS FUNCTION:*)
(*)           DISPLAYS THE REVIEW STRING PANEL*)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS:*)
(*)   NAME      I/O  DESCRIPTION*)
(*)   ====      ==  =====*)
(*)   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL*)
(*)   SIZE       0   THE SIZE OF THE STRING ENTERED*)
(*)   NEXT_OP    0   ENUMERATED TYPE INDICATING THE NEXT*)
(*)               OPERATION*)
(*)   RR         0   INDICATES IF AN ERROR HAS OCCURRED AND,*)
(*)               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN*)
(*)*)
(*) $COMMONS:*)
(*)   NONE*)
(*)*)
(*) $ENVIRONMENT:*)
(*)   LANGUAGE: IBM PASCAL*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381*)
(*)   DDNAMES USED WITH STANDARD FILES:*)
(*)   NONE*)
(*)*)
(*) $EXECUTION PROCEDURE:*)
(*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE*)
(*)*)
(*) $PROCESSING DESCRIPTION:*)
(*)   DISPLAY THE REVIEW STRING PANEL (RESTRING) BY MAKING ISPLNK*)
(*)   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED*)
(*)   TYPE.)*
(*)*)
(*) $COMMENTS:*)
(*)   NONE*)
(*)*)
(*) $CHANGE CONTROL:*)
(*)*)

```

```

(*) %INCLUDE RESTRUC *)
(**)
PROCEDURE RESTRUC(VAR MESS      : MESSAGE;
                  VAR MEMBERS   : T_ARRAYID;
                  VAR SIZE      : INTEGER;
                  VAR MEMBER     : T_NAME;
                  VAR NEXT_OP    : OPERATIONS;
                  VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*) THIS FUNCTION:*)
(*) DISPLAYS THE REVIEW STRUCTURE PANEL*)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS:*)
(*) NAME I/O DESCRIPTION*)
(*)==== === =====*)
(*) MESS I THE ERROR MESSAGE DISPLAYED ON THE PANEL*)
(*) MEMBERS I THE ARRAY OF MEMBERS TO SELECT FROM*)
(*) SIZE I THE SIZE OF THE ARRAY OF MEMBERS*)
(*) MEMBER O THE MEMBER SELECTED*)
(*) NEXT_OP O ENUMERATED TYPE INDICATING THE NEXT*)
(*) OPERATION*)
(*) RR O INDICATES IF AN ERROR HAS OCCURRED AND,*)
(*) IF ONE HAS, WHAT ROUTINE IT OCCURRED IN*)
(*)*)
(*) $COMMONS:*)
(*) NONE*)
(*)*)
(*) $ENVIRONMENT:*)
(*) LANGUAGE: IBM PASCAL*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381*)
(*) DDNAMES USED WITH STANDARD FILES:*)
(*) NONE*)
(*)*)
(*) $EXECUTION PROCEDURE:*)
(*) SCHEMA EXECUTIVE MENU INTERFACE ROUTINE*)
(*)*)
(*) $PROCESSING DESCRIPTION:*)
(*) DISPLAY THE REVIEW STRUCTURE PANEL (RESTRUC) BY MAKING*)
(*) ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN*)
(*) ENUMERATED TYPE. *)
(*)*)

```

CI PS560240032U
April 1990

(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```
(* %INCLUDE RESUBSCM *)
(**)
PROCEDURE RESUBSCM(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR COMMENT    : CHAR150;
                   VAR MEMBERS    : T_ARRAYTV;
                   VAR SIZE       : INTEGER;
                   VAR MEMBER     : T_NAME;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE REVIEW SUBSCHEMA PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME       I   THE SUBSCHEMA NAME
(*   MEMBERS    I   THE ARRAY OF MEMBERS TO SELECT FROM
(*   SIZE       I   THE SIZE OF THE ARRAY OF MEMBERS
(*   MEMBER     O   THE MEMBER SELECTED
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE REVIEW SUBSCHEMA PANEL (RESUBSCM) BY MAKING
(*   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE.
```

CI PS560240032U
April 1990

(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```
(* %INCLUDE RESUPTYP *)
(**)
PROCEDURE RESUPTYP(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE REVIEW SUPERTYPE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME          I/O  DESCRIPTION
(*      ====          ==  =====
(*      MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      NAME          I    THE SUPERTYPE NAME
(*      NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                      OPERATION
(*      RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                      IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE REVIEW SUPERTYPE PANEL (RESUPTYP) BY MAKING
(*      ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*      ENUMERATED TYPE.
(*
(* $COMMENTS:
(*      NONE
(*
(* $CHANGE CONTROL:
(*
```

```
(* BEGIN %INCLUDE RSCPAI *****)
(*)
PROCEDURE RSCPAI ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                  CONST INPUT_VALUE   : T_ARRAY_INDEX;
                  CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*) COPY THE ARRAY INDEX TABLE INFORMATION INTO THE RUN-TIME
(*) SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*) ====
(*) INPUT_VALUE I INPUT VALUE OF ARBITRARY SIZE
(*) OUTPUT_VALUE O OUTPUT VALUE OF ARBITRARY SIZE
(*) SIZE_OF_VALUE I SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*) CALL MACHINE DEPENDENT ROUTINE TO COPY ARRAY TABLE INFO
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*) ORIGINATED: 01 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPAI *****)
```



```

(*) BEGIN %INCLUDE RSCPAT *****)
(*)
PROCEDURE RSCPAT ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                  CONST INPUT_VALUE   : T_ARRAY_LIST;
                  CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE SIZE AND THE LOWER BOUND OF THE ARRAY INTO THE
(*)     RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ==  =====
(*)     INPUT_VALUE     I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE    O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE   I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY THE SIZE AND THE
(*)     LOWER BOUND OF THE ARRAY
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 01 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPAT *****)

```

```
(* BEGIN %INCLUDE RSCPCI *****)
(*)
PROCEDURE RSCPCI ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                   CONST INPUT_VALUE  : T_CL_INDEX;
                   CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;
(*)
(*) $FUNCTION: (*)
(*)     COPY THE POINTER INDEX TABLE INFORMATION INTO THE (*)
(*)     RUN-TIME SUBSCHEMA. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)     NAME          I/O  DESCRIPTION (*)
(*)     ====          ==  ===== (*)
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE (*)
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE (*)
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)     RUN-TIME SUBSCHEMA (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY CL INDEX INFO (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)     ORIGINATED: 29 JANUARY 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE RSCPCI *****)
```

```

(* BEGIN %INCLUDE RSCPCT *****
(*)
PROCEDURE RSCPCT ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                   CONST INPUT_VALUE  : T_CL_KINDS;
                   CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE KINDS OF POINTERS INTO THE RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY KINDS OF POINTER
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 29 JANUARY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPCT *****

```

```

(* BEGIN %INCLUDE RSCPEI *****)
(*)
PROCEDURE RSCPEI ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                   CONST INPUT_VALUE  : T_ENUM_INDEX;
                   CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE ENUMERATION INDEX TABLE INFORMATION INTO THE
(*)     RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY ENUMERATION INFO
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 01 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPEI *****)

```

```
(* BEGIN %INCLUDE RSCPET *****)
(*)
PROCEDURE RSCPET ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                   CONST INPUT_VALUE   : T_ENUMERATION;
                   CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE ENUMERATION VALUES INTO THE RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY ENUMERATION TABLE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED
(*)             GLOBAL DECLARATIONS INTO 'NVITYP'.
(*)     ORIGINATED: 15 OCTOBER 1985, G. A. WHITE, FRMI
(*)
(*) END %INCLUDE RSCPET *****)
```

```
(* BEGIN %INCLUDE RSFILE *****)
(*)
PROCEDURE RSFILE ( VAR  SUBSCHEMA_KEY  : ENTKEY;
                   VAR  IRC             : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     FILE RUN-TIME SUBSCHEMA INTO SEQUENTIAL FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     SUBSCHEMA_KEY  I    SUBSCHEMA KEY OF THE ENTITY DEFINITIONS
(*)     IRC            0    RETURN CODE
(*)                      = 0  SUCCESS
(*)                      > 0  CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)     CALLED FROM THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     OPEN DATAFILE
(*)     OPEN INXFILE
(*)     WRITE ARRAY_ENTITY (1100) IF THERE IS ARRAY OF POINTER
(*)       IN THE ENTITY DEFINITIONS
(*)     LOOP THROUGH LIST OF SUBSCHEMA
(*)       GET RUN_TIME SUBSCHEMA ( RSGTSM )
(*)       WRITE KIND, RECORD NO, OFFSET, RUNTIME_SIZE INTO INXFILE
(*)       FOR INDEX := 1 TO RUNTIME_SIZE
(*)         WRITE RUN_TIME SUBSCHEMA INTO DATAFILE
(*)     END LOOP
(*)     CLOSE DATAFILE
(*)     CLOSE INXFILE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 27 JANUARY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSFILE *****)
```

```

(*) BEGIN %INCLUDE RSGTSM *****
(*)
PROCEDURE RSGTSM ( CONST ENTITY_KEY      : ENIKEY;
                   VAR  RUN_TIME         : T_RUN_TIME;
                   VAR  RUN_TIME_SIZE    : INTEGER;
                   VAR  IRC               : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     BUILD RUN-TIME SUBSCHEMA FROM SCHEMA MODEL
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME      I/O  DESCRIPTION
(*)     ====      ==  =====
(*)     ENTITY_KEY  I   KEY FROM SCHEMA MODEL WHICH THE
(*)                   TRANSLATION WILL BE PERFORMED.
(*)     IRC         0   RETURN CODE
(*)                   = 0  SUCCESS
(*)                   > 0  CRITICAL ERROR:
(*)     RUN_TIME    0   RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*)                   ENTITY DEFINITION, ALONG WITH ANY
(*)                   ENUMERATION VALUES, CONSTITUENT LIST,
(*)                   AND ARRAY INFORMATION, IN A COMPACTED
(*)                   FORM.
(*)     RUN_TIME_SIZE 0   THE NUMBER OF BYTES ACTUALLY REQUIRED
(*)                   FOR THE COMPACTED RUN-TIME SUBSCHEMA.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)     CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     TRANSLATE SCHEMA MODEL ENTRY INTO ENTITY ATTRIBUTES
(*)     AND ENUMERATION VALUES AND ARRAY INFORMATION
(*)     IF THERE WERE ANY ENUMERATION ATTRIBUTES THEN
(*)     CALCULATE THE STARTING POSITION OF THE ENUMERATION
(*)     INDEX TABLE AND STORE INTO RUN-TIME SUBSCHEMA
(*)     DETERMINE THE ACTUAL SIZE OF THE ENUMERATION INDEX TABLE
(*)     COPY ENUMERATION INDEX TABLE INFORMATION INTO RUN-TIME
(*)                                     SUBSCHEMA
(*)     ENDIF

```

```
(* IF THERE WERE ANY ENUMERATION ATTRIBUTES THEN *)
(* CALCULATE THE STARTING POSITION OF THE ENUMERATION *)
(* VALUE TABLE AND STORE INTO RUN-TIME SUBSCHEMA *)
(* DETERMINE THE ACTUAL SIZE OF THE ENUMERATION VALUE TABLE *)
(* COPY THE ENUMERATION VALUES INTO THE RUN-TIME SUBSCHEMA *)
(* ENDIF *)
(* IF THERE WERE ANY ARRAY ATTRIBUTES THEN *)
(* CALCULATE THE STARTING POSITION OF THE ARRAY INDEX TABLE *)
(* AND STORE INTO RUN-TIME SUBSCHEMA *)
(* DETERMINE THE ACTUAL SIZE OF THE ARRAY INDEX TABLE *)
(* COPY ARRAY TABLE INDEX INFORMATION INTO RUN-TIME SUBSCHEMA *)
(* ENDIF *)
(* IF THERE WERE ANY ARRAY ATTRIBUTES THEN *)
(* CALCULATE THE STARTING POSITION OF THE ARRAY LIST TABLE *)
(* AND STORE INTO RUN-TIME SUBSCHEMA *)
(* DETERMINE THE ACTUAL SIZE OF THE ARRAY LIST TABLE *)
(* COPY ARRAY LIST INFORMATION INTO RUN-TIME SUBSCHEMA *)
(* ENDIF *)
(* CALCULATE THE SIZE OF THE RUN-TIME SUBSCHEMA *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* REVISED: 15 SEPTEMBER 1987, M. H. CHOI, DBMA *)
(* ADDED SUPER TYPE IN THE ENTITY *)
(* REVISED: 12 AUGUST 1987, M. H. CHOI, DBMA *)
(* ADDED MINIMUM OCCURRENCES TO ATTRIBUTE *)
(* ORIGINATED: 11 AUGUST 1986, M. H. CHOI, FRMI *)
(* *)
(* END %INCLUDE RSGTSM *****)
```



```

(* BEGIN %INCLUDE RSMASKND *****)
(*)
PROCEDURE RSMASKND ( VAR ENTITY          : T_SCHEMA );
SUBPROGRAM;

(*)
(*) $FUNCTION: (*)
(*)   INSERT THE MODEL ACCESS SOFTWARE(MAS) ATTRIBUTES ( KIND, (*)
(*)   LENGTH, SYSUSE ) INTO A RUN-TIME SUBSCHEMA. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME          I/O  DESCRIPTION (*)
(*)   ====          ==  ===== (*)
(*)   ENTITY        O   RUN-TIME SUBSCHEMA ENTITY DEFINITION. (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)   NAME/VALUE INTERFACE (*)
(*)   CALLED FROM THE NAME/VALUE INTERFACE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)   INSERT KIND, LENGTH, SYSUSE ATTRIBUTES INTO A RUN-TIME (*)
(*)   SUBSCHEMA ENTITY DEFINITION. (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)   REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) (*)
(*)   ORIGINATED: 15 SEPTEMBER 1987, M. H. CHOI, DBMA (*)
(*)
(*) END %INCLUDE RSMASKND *****)

```

```

(*) BEGIN %INCLUDE RSORDER *****
(*)
PROCEDURE RSORDER ( CONST RUNTIME      : T_RUN_TIME;
                    VAR  PS_ORDER      : T_PS_ORDER );
    SUBPROGRAM;
(*)
(*) $FUNCTION:
(*)     DETERMINE THE PHYSICAL SCHEMA ORDER OF ENTITY DEFINITION
(*)     BY ITS OFFSET.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     RUNTIME        I    CONTAINS THE ENTITY DEFINITION
(*)     PS_ORDER       0    LIST OF ATTRIBUTES IN PHYSICAL
(*)                       ORDER
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH THE NUMBER OF ATTRIBUTES IN THE DEFINITION
(*)     INSERT THE OFFSET IN THE PHYSICAL SCHEMA ORDER TABLE
(*)     END LOOP
(*)     LOOP THROUGH THE NUMBER OF ATTRIBUTES IN THE DEFINITION
(*)     IF CURRENT OFFSET GREATER THAN NEXT OFFSET THEN
(*)     SWITCH CURRENT OFFSET WITH NEXT OFFSET
(*)     END IF
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 27 JANUARY 1988, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSORDER *****

```

```
(* BEGIN %INCLUDE RSTRGF *****~******)
(*)
PROCEDURE RSTRGF ( VAR ENTITY      : T_SCHEMA;
                   VAR ENUM       : T_ENUM_COMPACTOR;
                   VAR ENUM_INDEX : T_ENUM_INX_COMPACTOR;
                   VAR ARRAY_LIST : T_ARRAY_LIST_COMPACTOR;
                   VAR ARRAY_INDEX : T_ARRAY_INX_COMPACTOR;
                   VAR CL_INDEX   : T_CL_INX_COMPACTOR;
                   VAR CL_LIST    : T_CL_KINDS_COMPACTOR;
                   VAR IRC        : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     TRANSLATE THE GLOBAL FIELDS INTO A RUN-TIME SUBSCHEMA
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME      I/O DESCRIPTION
(*)     ====      == =====
(*)     ENTITY    0  RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*)                ENTITY DEFINITION
(*)     ENUM      0  ENUMERATION VALUES
(*)     ENUM_INDEX 0  ENUMERATION INDEX TABLE
(*)     ARRAY_LIST 0  LOWER BOUND AND ARRAY SIZE
(*)     ARRAY_INDEX 0  ARRAY INDEX TABLE
(*)     CL_INDEX   0  CONSTITUENT INDEX TABLE
(*)     CL_LIST    0  CONSTITUENT KINDS
(*)     IRC        0  RETURN CODE
(*)                = 0  SUCCESS
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)     CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     INITIALIZE TABLE INDEXES
(*)     MAKE A LIST OF GLOBAL KINDS
(*)     LOOP THROUGH A LIST OF GLOBAL KINDS
(*)     TRANSLATE GLOBAL FIELD INTO A RUN-TIME SUBSCHEMA
(*)
(*) $COMMENTS:
(*)
```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 20 NOVEMBER 1986, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE RSTRGF *****)

```
(* BEGIN %INCLUDE RSTRSM *****)
(*)
PROCEDURE RSTRSM ( CONST ENTITY_KEY      : LISTKEY;
                   VAR  ENTITY          : T_SCHEMA;
                   VAR  ENUM             : T_ENUM_COMPACTOR;
                   VAR  ENUM_INDEX       : T_ENUM_INX_COMPACTOR;
                   VAR  ARRAY_LIST       : T_ARRAY_LIST_COMPACTOR;
                   VAR  ARRAY_INDEX      : T_ARRAY_INX_COMPACTOR;
                   VAR  CL_INDEX         : T_CL_INX_COMPACTOR;
                   VAR  CL_LIST          : T_CL_KINDS_COMPACTOR;
                   VAR  IRC              : RET_REC );
SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) TRANSLATE A SCHEMA MODEL ENTRY INTO A RUN-TIME SUBSCHEMA
(*) ENTITY, ENUMERATION TABLE AND ARRAY INFO TABLE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*) NAME          I/O DESCRIPTION
(*) =====
(*) ARRAY_INDEX    0  RUN-TIME SUBSCHEMA ARRAY TABLE INDEX
(*)                INFORMATION.
(*) ARRAY_LIST     0  RUN-TIME SUBSCHEMA ARRAY TABLE
(*)                AND COMPACTION INFORMATION.
(*) ENUM           0  RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                AND COMPACTION INFORMATION.
(*) ENUM_INDEX     0  RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                INDEX INFORMATION.
(*) ENTITY         0  RUN-TIME SUBSCHEMA ENTITY DEFINITION.
(*) ENTITY_KEY     I  KEY FROM SCHEMA MODEL WHICH THE
(*)                TRANSLATION WILL BE PERFORMED.
(*) IRC           0  RETURN CODE
(*)                = 0 SUCCESS
(*)                > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) NAME/VALUE INTERFACE
(*) CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*) OBTAIN ENTITY NAME AND KIND FROM SCHEMA MODEL
(*)
```

```
(*      STORE ENTITY NAME AND KIND INTO RUN-TIME SUBSCHEMA      *)
(*      LOOP THROUGH SCHEMA MODEL ENTRIES                      *)
(*      OBTAIN ATTRIBUTE ENTRY FROM SCHEMA MODEL                *)
(*      CASE DATA TYPE OF                                      *)
(*          INTEGER, REAL, STRING, LOGICAL                      *)
(*              : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) *)
(*      POINTER :   CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) *)
(*      ARRAY   :   ARRAY_ATTRIBUTE, PROCEDURE (3) *)
(*      DEFINED_TYPE :   DEFINED_TYPE_ATTRIBUTE, PROCEDURE (4) *)
(*      OTHERWISE :   ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' *)
(*      ENDCASE                                                *)
(*      ENDLOOP                                                *)

(*      PROCEDURE (1) : APPLICATION_DATA_BLOCK_ATTRIBUTE      *)
(*      STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY *)
(*      *)
(*      PROCEDURE (2) : CONSTITUENT_LIST_ATTRIBUTE            *)
(*      OBTAIN CONSTITUENT LIST POSITION FROM SCHEMA MODEL      *)
(*      STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY *)
(*      *)
(*      PROCEDURE (3) : ARRAY_ATTRIBUTE                        *)
(*      DETERMINE THE NUMBER OF ARRAY DIMENSIONS              *)
(*      STORE ARRAY INFORMATION INTO RUN-TIME SUBSCHEMA        *)
(*      STORE TABLE INDEX POSITION FOR ARRAY LIST TABLE AND THE *)
(*      NUMBER OF DIMENSIONS INTO ARRAY INDEX TABLE          *)
(*      CALCULATE TOTAL SIZE OF THE ARRAY AND STORE INTO ARRAY *)
(*      INDEX TABLE                                          *)
(*      FOR THE NUMBER OF ARRAY DIMENSIONS                    *)
(*      CALCULATE THE SIZE OF EACH ARRAY                      *)
(*      STORE SIZE AND LOW-BOUND INTO ARRAY LIST TABLE      *)
(*      END LOOP                                              *)
(*      *)
(*      PROCEDURE (4) : DEFINED_TYPE_ATTRIBUTE                *)
(*      OBTAIN DATA TYPE FOR DEFINED TYPE ATTRIBUTE IN SCHEMA MODEL *)
(*      CASE DATA_TYPE OF                                    *)
(*          INTEGER, REAL, STRING, LOGICAL                    *)
(*              : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) *)
(*          ENUMERATION :   ENUMERATION_ATTRIBUTE, PROCEDURE (5) *)
(*          POINTER :   CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) *)
(*          ARRAY   :   ARRAY_ATTRIBUTE, PROCEDURE (3) *)
(*          OTHERWISE :   ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' *)
(*      ENDCASE                                                *)
(*      *)
```

CI PS560240032U
April 1990

```
(*  PROCEDURE (5) : ENUMERATION_ATTRIBUTE *)
(*  STORE ATTRIBUTE DEFINITION FOR ENUMERATION TYPE *)
(*  OBTAIN NUMBER OF ENUMERATION VALUES FROM SCHEMA MODEL *)
(*  STORE NUMBER OF ENUMERATION VALUE IN ENUMERATION INDEX TABLE*)
(*  STORE ENUMERATION VALUE TABLE INDEX POSITION IN ENUMERATION *)
(*  INDEX TABLE *)
(*  LOOP THROUGH ENUMERATION VALUES *)
(*  OBTAIN ENUMERATION VALUE FROM SCHEMA MODEL *)
(*  STORE ENUMERATION VALUE IN ENUMERATION VALUE TABLE *)
(*  END LOOP *)
(*  $COMMENTS: *)
(*  $CHANGE CONTROL: *)
(*  REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*  ORIGINATED: 06 AUGUST 1986, M. H. CHOI, FRMI *)
(*  END %INCLUDE RSTRSM *****)
```

```

(*) BEGIN %INCLUDE RSTRST *****
(*)
PROCEDURE RSTRST ( CONST SUBTYPE_KEY      : ENTKEY;
                   VAR  ENTITY            : T_SCHEMA;
                   VAR  ENUM              : T_ENUM_COMPACTOR;
                   VAR  ENUM_INDEX        : T_ENUM_INX_COMPACTOR;
                   VAR  ARRAY_LIST        : T_ARRAY_LIST_COMPACTOR;
                   VAR  ARRAY_INDEX       : T_ARRAY_INX_COMPACTOR;
                   VAR  CL_INDEX          : T_CL_INX_COMPACTOR;
                   VAR  CL_LIST           : T_CL_KINDS_COMPACTOR;
                   VAR  IRC               : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   TRANSLATE SUPER TYPE INTO A RUN-TIME SUBSCHEMA
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O  DESCRIPTION
(*)   ====      ==  =====
(*)   SUBTYPE_KEY  I   KEY FROM SCHEMA MODEL WHICH THE
(*)                   TRANSLATION WILL BE PERFORMED.
(*)                   > 0  CRITICAL ERROR:
(*)   ENTITY      0   RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*)                   ENTITY DEFINITION
(*)   ENUM         0   ENUMERATION VALUES
(*)   ENUM_INDEX   0   ENUMERATION INDEX TABLE
(*)   ARRAY_LIST   0   LOWER BOUND AND ARRAY SIZE
(*)   ARRAY_INDEX  0   ARRAY INDEX TABLE
(*)   CL_INDEX     0   CONSTITUENT INDEX TABLE
(*)   CL_LIST      0   CONSTITUENT KINDS
(*)   IRC         0   RETURN CODE
(*)                   = 0  SUCCESS
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   RUN-TIME SUBSCHEMA
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   MAKE A LIST OF SUPER TYPE
(*)   LOOP THROUGH A LIST OF SUPER TYPE
(*)   TRANSLATE SUPER TYPE INTO A RUN-TIME SUBSCHEMA
(*)

```


CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* ORIGINATED: 9 SEPTEMBER 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE RSTRST *****)
```

```

(* BEGIN %INCLUDE RSTRSTC *****)
(*)
PROCEDURE RSTRSTC ( CONST ENTITY_KEY      : LISTKEY;
                    VAR  S_OFFSET        : INTEGER;
                    VAR  ENTITY          : T_SCHEMA;
                    VAR  ENUM            : T_ENUM_COMPACTOR;
                    VAR  ENUM_INDEX      : T_ENUM_INX_COMPACTOR;
                    VAR  ARRAY_LIST      : T_ARRAY_LIST_COMPACTOR;
                    VAR  ARRAY_INDEX     : T_ARRAY_INX_COMPACTOR;
                    VAR  CL_INDEX        : T_CL_INX_COMPACTOR;
                    VAR  CL_LIST         : T_CL_KINDS_COMPACTOR;
                    VAR  IRC             : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   TRANSLATE A STRUCTURE ATTRIBUTE INTO A RUN-TIME SUBSCHEMA
(*)   ENTITY, ENUMERATION TABLE AND ARRAY INFO TABLE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ARRAY_INDEX   0    RUN-TIME SUBSCHEMA ARRAY TABLE INDEX
(*)                   INFORMATION.
(*)   ARRAY_LIST    0    RUN-TIME SUBSCHEMA ARRAY TABLE
(*)                   AND COMPACTION INFORMATION.
(*)   ENUM          0    RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                   AND COMPACTION INFORMATION.
(*)   ENUM_INDEX    0    RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                   INDEX INFORMATION.
(*)   ENTITY        0    RUN-TIME SUBSCHEMA ENTITY DEFINITION.
(*)   ENTITY_KEY    I    KEY FROM SCHEMA MODEL WHICH THE
(*)                   TRANSLATION WILL BE PERFORMED.
(*)   IRC          0    RETURN CODE
(*)                   = 0 SUCCESS
(*)                   > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)

```

```

(*) $PROCESSING DESCRIPTION: (*)
(*)   OBTAIN ENTITY NAME AND KIND FROM SCHEMA MODEL (*)
(*)   STORE ENTITY NAME AND KIND INTO RUN-TIME SUBSCHEMA (*)
(*)   LOOP THROUGH SCHEMA MODEL ENTRIES (*)
(*)   OBTAIN ATTRIBUTE ENTRY FROM SCHEMA MODEL (*)
(*)   CASE DATA TYPE OF (*)
(*)     INTEGER, REAL, STRING, LOGICAL (*)
(*)       : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) (*)
(*)     POINTER : CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) (*)
(*)     ARRAY   : ARRAY_ATTRIBUTE, PROCEDURE (3) (*)
(*)     DEFINED_TYPE : DEFINED_TYPE_ATTRIBUTE, PROCEDURE (4) (*)
(*)     OTHERWISE : ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' (*)
(*)   ENDCASE (*)
(*)   ENDLLOOP (*)

(*) PROCEDURE (1) : APPLICATION_DATA_BLOCK_ATTRIBUTE (*)
(*)   STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY (*)
(*)
(*) PROCEDURE (2) : CONSTITUENT_LIST_ATTRIBUTE (*)
(*)   OBTAIN CONSTITUENT LIST POSITION FROM SCHEMA MODEL (*)
(*)   STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY (*)
(*)
(*) PROCEDURE (3) : ARRAY_ATTRIBUTE (*)
(*)   DETERMINE THE NUMBER OF ARRAY DIMENSIONS (*)
(*)   STORE ARRAY INFORMATION INTO RUN-TIME SUBSCHEMA (*)
(*)   STORE TABLE INDEX POSITION FOR ARRAY LIST TABLE AND THE (*)
(*)     NUMBER OF DIMENSIONS INTO ARRAY INDEX TABLE (*)
(*)   CALCULATE TOTAL SIZE OF THE ARRAY AND STORE INTO ARRAY (*)
(*)   INDEX TABLE (*)
(*)   FOR THE NUMBER OF ARRAY DIMENSIONS (*)
(*)     CALCULATE THE SIZE OF EACH ARRAY (*)
(*)   STORE SIZE AND LOW-BOUND INTO ARRAY LIST TABLE (*)
(*)   END LOOP (*)
(*)
(*) PROCEDURE (4) : DEFINED_TYPE_ATTRIBUTE (*)
(*)   OBTAIN DATA TYPE FOR DEFINED TYPE ATTRIBUTE IN SCHEMA MODEL (*)
(*)   CASE DATA_TYPE OF (*)
(*)     INTEGER, REAL, STRING, LOGICAL (*)
(*)       : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) (*)
(*)     ENUMERATION : ENUMERATION_ATTRIBUTE, PROCEDURE (5) (*)
(*)     POINTER : CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) (*)
(*)     ARRAY   : ARRAY_ATTRIBUTE, PROCEDURE (3) (*)
(*)     OTHERWISE : ERROR MESSAGE = 'UNKNOWN ATTRIBUTE TYPE' (*)
(*)   ENDCASE (*)
(*)

```

```
(* PROCEDURE (5) : ENUMERATION_ATTRIBUTE *)
(* STORE ATTRIBUTE DEFINITION FOR ENUMERATION TYPE *)
(* OBTAIN NUMBER OF ENUMERATION VALUES FROM SCHEMA MODEL *)
(* STORE NUMBER OF ENUMERATION VALUE IN ENUMERATION INDEX TABLE*)
(* STORE ENUMERATION VALUE TABLE INDEX POSITION IN ENUMERATION *)
(* INDEX TABLE *)
(* LOOP THROUGH ENUMERATION VALUES *)
(* OBTAIN ENUMERATION VALUE FROM SCHEMA MODEL *)
(* STORE ENUMERATION VALUE IN ENUMERATION VALUE TABLE *)
(* END LOOP *)
(* $COMMENTS: *)
(* $CHANGE CONTROL: *)
(* REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(* ORIGINATED: 15 JANUARY 1988, M. H. CHOI, DBMA *)
(* END %INCLUDE RSTRSTC *****)
```

```
(* BEGIN %INCLUDE RS1100 *****)
(*)
PROCEDURE RS1100 ( CONST SUBSCHEMA_KEY      : ENTKEY;
                   VAR  RUN_TIME            : T_RUN_TIME;
                   VAR  RUN_TIME_SIZE       : INTEGER;
                   VAR  IRC                 : RET_REC );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   STORE ARRAY_ENTITY(1100) IN THE DATA DICTIONARY AND
(*)   THE RUN-TIME SUBSCHEMA
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   RUN_TIME      0    RUN-TIME SUBSCHEMA WHICH CONTAINS
(*)                   THE ENTITY DEFINITION, ALONG WITH
(*)                   ANY ENUMERATION VALUES, CONSTITUENT
(*)                   LIST, AND ARRAY INFORMATION, IN A
(*)                   COMPACTED FORM.
(*)   RUN_TIME_SIZE 0    THE NUMBER OF BYTES ACTUALLY REQUIRED*)
(*)                   FOR THE COMPACTED RUN-TIME SUBSCHEMA.*)
(*)   IRC           0    RETURN CODE
(*)                   = 0 SUCCESS
(*)                   > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   OBTAIN ENTITY NAME AND KIND FROM SCHEMA MODEL
(*)   STORE ENTITY NAME AND KIND INTO RUN-TIME SUBSCHEMA
(*)   STORE GLOBAL ATTRIBUTES
(*)   STORE CL_ENTITIES ATTRIBUTE
(*)
(*) $COMMENTS:
(*)
```

CI PS560240032U
April 1990

```
(* $CHANGE CONTROL: *)
(* REVISD: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(* ORIGINATED: 20 AUGUST 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE RS1100 *****)
```

```
(* %INCLUDE SCALFSRT *)
(**)
  PROCEDURE SCALFSRT(CONST CURRENT : ENTBLOCK;
                    CONST NEXT   : ENTBLOCK;
                    VAR  FLIP    : BOOLEAN;
                    VAR  RRC     : EXT_RET_CODE;
                    VAR  PROC    : ROUTINE);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE IS THE ORDER FUNCTION CALLED BY MALSRT
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O DESCRIPTION
(*   ====          ==
(*   CURRENT       I   THE ADB OF THE CURRENT ENTITY
(*   NEXT          I   THE ADB OF THE NEXT ENTITY
(*   FLIP          O   INDICATES IF THE ENTITIES SHOULD BE
(*                   FLIPPED
(*   RRC           O   THE ROUTINE'S RETURN CODE
(*                   = 0 OK
(*                   <> 0 ERROR
(*   XRC           O   EXTERNAL RETURN CODE FROM MAS
(*                   = 0 OK
(*                   >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY THE MAS ROUTINE MALSRT. THE
(*   TWO ENTITIES ARE COMPARED AND IF THEY ARE OUT OF ALPHA-
(*   BETICAL ORDER THE FLIP FLAG IS SET TO TRUE OTHERWISE THE
(*   FLAG REMAINS FALSE. IF THE FLAG IS TRUE THE ENTITIES ARE
(*   SWAPPED OTHERWISE THEY ARE NOT.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```

(** %INCLUDE SCARYCR *)
(**)
PROCEDURE SCARYCR(VAR IRC : RET_REC;
                  VAR TRANS_STACK : TRANSPTR;
                  VAR CONTEXT : T_CONTEXT);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)     THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE *)
(*)     ARRAY ENTITY AND PUSHES THE DATA ON THE TRANSACTION *)
(*)     STACK. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)     NAME          I/O  DESCRIPTION *)
(*)     ====          ==  ===== *)
(*)     IRC            0    THE RETURN CODE *)
(*)     TRANS_STACK    I/O  THE TRANSACTION STACK *)
(*)     CONTEXT        I    THE CONTEXT IN WHICH THE ARRAY IS BEING *)
(*)                      CREATED *)
(*) *)
(*) $COMMONS: *)
(*)     REF *)
(*)     INSIDE        I/O  INDICATES IF THE EXIT OR RETURN OPTION *)
(*)                      IS CHOSEN WITHIN ANOTHER ROUTINE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)     LANGUAGE: IBM PASCAL *)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)     THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRARRAY *)
(*)     TO DISPLAY THE PANEL. THE DATA ENTERED ON THE PANEL IS *)
(*)     VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION *)
(*)     STACK OR THE APPROPRIATE ACTION IS TAKEN. IF ANY OF THE *)
(*)     DATA ENTERED IS INVALID THEN THE PANEL IS REDISPLAYED *)
(*)     WITH AN ERROR MESSAGE. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)     REVISED: JANUARY 1988      C. H. MOHME      DBMA *)

```


CI PS560240032U

April 1990

```
(*      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE      *)
(*)
(*)      REVISED: 10/09/87      C. H. MOHME      DBMA      *)
(*)      ADDED PARAMETER TO SCDEFRCR CALL.      *)
(*)
(*)      REVISED: 08/13/87      C. H. MOHME      DBMA      *)
(*)      ADDED LIST AND SET.  NOTE:  THE LIST AND SET DATA TYPES ARE  *)
(*)      IMPLEMENTED IN THE SOFTWARE AS AN ARRAY.  A FIELD WAS ADDED  *)
(*)      TO THE ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL *)
(*)      ARRAY, LIST, OR SET.      *)
(*)
(*)      ORIGINATED: 04/22/86      L. J. BEHAN      FRMI      *)
(*)
(*)-----*)
(*)
(*)END-----*)
(* END %INCLUDE SCARYCR *)
```

```
(* %INCLUDE SCARYUP *)
(**)
PROCEDURE SCARYUP(VAR IRC : RET_REC;
                  VAR TRANS_STACK : TRANSPTR;
                  VAR ARRAY_KEY : ENTKEY;
                  VAR NUMBER_OF_USERS : LISTKEY;
                  VAR DELETE_LIST : LISTKEY;
                  VAR NEW_KEYS_LIST : LISTKEY;
                  VAR CONTEXT      : T_CONTEXT);
SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE GATHERS THE DATA TO UPDATE AN ARRAY. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 RETURN CODE *)
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK *)
(* ARRAY_KEY I/O KEY OF THE ARRAY TO BE UPDATED *)
(* NUMBER_OF_USERS I/O THE NUMBER OF USERS OF THE ARRAY *)
(* DELETE_LIST I/O LIST OF ENTITIES TO BE DELETED *)
(* NEW_KEYS_LIST I/O LIST OF NEWLY CREATED ENTITIES *)
(* CONTEXT I THE CONTEXT IN WHICH THE ARRAY IS BEING *)
(* UPDATED *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* THE CURRENT ARRAY DATA IS DISPLAYED ON THE UPARRAY PANEL. *)
(* THE DATA CAN THEN BE UPDATED BY THE USER. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE SCBASIN *)
(**)
PROCEDURE SCBASIN(VAR IRC      : RET_REC;
                  VAR INCLD    : TEXT;
                  VAR ENTITY_KEY : ENTKEY;
                  VAR ADB      : ENTITY_ADB);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE WRITES THE ENTITY KIND CONSTANTS TO THE
(* PASCAL INCLUDE FILE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ====
(* IRC I/O RETURN CODE
(* INCLD I THE FILE NAME
(* ENTITY_KEY I THE ENTITY KEY
(* ADB O THE ENTITY ADB
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* GET THE ADB OF THE PRIMITIVE ENTITY
(* CASE PRIMITIVE ENTITY KIND OF
(* INTEGER :
(* CASE PRECISION IN DECIMAL DIGITS OF
(* 1, 2 : WRITE TO THE FILE PACKED 0..255;
(* 3, 4 : WRITE TO THE FILE PACKED 0..65535;
(* 5, 6, 7,
(* 8, 9 : WRITE TO THE FILE INTEGER;
(* END;
(* REAL :
(* CASE PRECISION IN DECIMAL DIGITS OF
(* 1, 2, 3, 4,
(* 5, 6, 7 : WRITE TO THE FILE SHORTREAL;
(* 8, 9, 10,
```

```

(*)      11, 12, 13, (*)
(*)      14, 15, 16 : WRITE TO THE FILE REAL; (*)
(*)      END; (*)
(*)      STRING : (*)
(*)      WRITE TO THE FILE PACKED ARRAY(.1..<STRING LENGTH>.) (*)
(*)      OF CHAR; (*)
(*)      DEFINED TYPE : (*)
(*)      WRITE TO THE FILE T_<DEFINED TYPE NAME>; (*)
(*)      LOGICAL : (*)
(*)      WRITE TO THE FILE BOOLEAN; (*)
(*)      ENUMERATION : (*)
(*)      WRITE TO THE FILE ( (*)
(*)      SET THE ENUMERATION'S CONSTITUENT LIST TO READ FORWARD (*)
(*)      COUNT THE NUMBER OF CONSTITUENTS (*)
(*)      FOR INDEX EQUALS ONE TO THE NUMBER OF CONSTITUENTS DO (*)
(*)      GET THE KEY TO THE NEXT CONSTITUENT IN THE LIST (*)
(*)      GET THE CONSTITUENT'S ADB (*)
(*)      WRITE TO THE FILE <ENUMERITEM NAME> (*)
(*)      IF THE COUNT IS EQUAL TO THE NUMBER OF CONSTITUENTS (*)
(*)      WRITE TO THE FILE ); (*)
(*)      ELSE (*)
(*)      WRITE TO THE FILE , (*)
(*)      END; (*)
(*)      ARRAY : (*)
(*)      GET THE ADB OF THE ARRAY ENTITY (*)
(*)      WRITE TO THE FILE ARRAY(.<LOW BOUND>..<HIGH BOUND>.) OF (*)
(*)      SET THE ARRAY ENTITY'S CONSTITUENT LIST TO READ FORWARD (*)
(*)      GET THE KEY TO THE ARRAY ENTITY'S FIRST CONSTITUENT (*)
(*)      GET THE CONSTITUENT'S ADB (*)
(*)      CALL THIS ROUTINE TO WRITE OUT THE ARRAY'S TYPE (*)
(*)      INTEGER (*)
(*)      REAL (*)
(*)      STRING (*)
(*)      LOGICAL (*)
(*)      ARRAY (*)
(*)      DEFINED TYPE (*)
(*)      POINTER (*)
(*)      END; (*)
(*)      STRUCTURE : (*)
(*)      WRITE TO THE FILE RECORD (*)
(*)      SET THE STRUCTURE'S CONSTITUENT LIST TO BE READ FORWARD (*)
(*)      CONTINUE TO READ CONSTITUENTS UNTIL THE END OF LIST (*)
(*)      GET THE KEY TO THE NEXT CONSTITUENT ENTITY ON THE LIST (*)
(*)      GET THE CONSTITUENT'S ADB (*)
(*)      WRITE TO THE FILE <FIELD NAME> : (*)
(*)      GET THE KEY TO THE CONSTITUENT'S FIRST CONSTITUENT (*)
(*)      GET THIS CONSTITUENT'S ADB (*)

```

CI PS560240032U
April 1990

```
(*          CALL THIS ROUTINE TO WRITE OUT THE PRIMITIVE TYPE      *)
(*          INTEGER                                                  *)
(*          REAL                                                      *)
(*          STRING                                                    *)
(*          LOGICAL                                                   *)
(*          ARRAY                                                     *)
(*          DEFINED TYPE                                              *)
(*          POINTER (IS NOT ALLOWED WITHIN A STRUCTURE)              *)
(*          IF THE END OF LIST IS FOUND THEN                          *)
(*          WRITE TO THE FILE END;                                    *)
(*          END;                                                      *)
(*          END;                                                      *)
(*          *)                                                        *)
(*          $COMMENTS:                                                *)
(*          *)                                                        *)
(*          $CHANGE CONTROL:                                          *)
(*          *)                                                        *)
```

```
(* %INCLUDE SCCHRCK *)
(**)
PROCEDURE SCCHRCK(VAR IRC      : RET_REC;
                  VAR NAME     : T_NAME;
                  VAR INVALID  : BOOLEAN);
SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) THIS CHECKS THAT THE CHARACTERS IN AN ENTITY NAME ARE VALID (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) IRC I/O RETURN CODE (*)
(*) NAME I NAME TO CHECK FOR VALID CHARACTERS (*)
(*) INVALID 0 INDICATES IF THE NAME CONTAINS AN IN- (*)
(*) VALID CHARACTER OR A SPACE INBETWEEN (*)
(*)
(*) $COMMONS: (*)
(*) NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*) THIS ROUTINE CHECKS THAT THE NAME CONTAINS VALID CHARAC- (*)
(*) TERS. VALID CHARACTERS ARE THE LETTERS A THRU Z, THE (*)
(*) DIGITS 0 THRU 9, BLANKS AND UNDERSCORES ARE ALLOWED. ANY (*)
(*) OTHER CHARACTERS ARE INVALID. THE NAME IS ALSO CHECKED (*)
(*) FOR SPACES INBETWEEN CHARACTERS. THIS TOO IS INVALID. (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)
```

```

(*) %INCLUDE SCCLSCR1 *)
(**)
PROCEDURE SCCLSCR1(VAR IRC : RET_REC;
                   VAR TRANS_STACK : TRANSPTR);
SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) THIS ROUTINE GATHERS THE NAME AND KIND NUMBER TO BE *)
(*) ASSIGNED TO THE CLASS ENTITY TO BE CREATED. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) IRC 0 RETURN CODE *)
(*) TRANS_STACK I/O POINTS TO THE TRANSACTION STACK *)
(*) *)
(*) $COMMONS: *)
(*) REF *)
(*) INSIDE I/O INDICATES IF THE EXIT OPTION OR RETURN *)
(*) HAS BEEN CHOSEN WITHIN ANOTHER CREATE *)
(*) PROCEDURE *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (CRCLASS1) *)
(*) WHICH PUTS UP THE MENU TO GATHER THE NAME AND KIND NUMBER *)
(*) FOR THE CLASS ENTITY. THE DATA IS VERIFIED FOR UNIQUENESS. *)
(*) IF THE INFORMATION ENTERED IS INDEED UNIQUE THEN IT IS *)
(*) PUSHED ONTO THE TRANSACTION STACK. THEN (SCCLSCR2) IS *)
(*) CALLED TO GATHER THE CONSTITUENTS. WHEN ALL THE REQUIRED *)
(*) DATA HAS BEEN ENTERED TO CREATE A CLASS ENTITY THE *)
(*) TRANSACTION PROCESSOR ROUTINE (SCTRSR) IS CALLED TO *)
(*) PROCESS THE TRANSACTIONS. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```

```

(*) %INCLUDE SCGLSCR2 *)
(**)
  PROCEDURE SCGLSCR2(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS ROUTINE GATHERS THE CONSTITUENTS OF THE CLASS ENTITY. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   IRC            0   RETURN CODE *)
(*)   TRANS_STACK    I/O  POINTS TO THE TRANSACTION STACK *)
(*)
(*) $COMMONS: *)
(*)   REF *)
(*)   INSIDE        I/O  INDICATES IF THE EXIT OPTION OR RETURN *)
(*)                   HAS BEEN CHOSEN WITHIN ANOTHER CREATE *)
(*)                   PROCEDURE *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (CRCLASS2) *)
(*)   WHICH PUTS UP THE MENU TO GATHER THE CONSTITUENTS. THE *)
(*)   CONSTITUENTS ARE PUSHED ON THE STACK AFTER VERIFYING THAT *)
(*)   THEY DO EXIST. *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)

```



```
(* %INCLUDE SCCLSUP *)
(**)
  PROCEDURE SCCLSUP(VAR IRC      : RET_REC;
                   VAR CLASS_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE THE CLASS ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   CLASS_KEY  I   KEY OF THE ENTITY TO BE UPDATED
(*   IRC        O   RETURN CODE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINES WHICH
(*   DISPLAY THE DATA DESCRIBING THE CLASS ENTITY.  THE UPDATE
(*   CLASS OPTIONS INCLUDE : CHANGE THE NAME AND/OR KIND NUMBER,
(*   UPDATE THE CONSTITUENT LIST BY
(*   ADDING OR REMOVING ELEMENTS,
(*   REVIEW A CONSTITUENT,
(*   DELETE THE CLASS ENTITY,
(*   SAVE THE CHANGES,
(*   RETURN AND EXIT.
(*   THE CHANGES MADE TO THE CLASS ENTITY ARE KEPT ONLY IF THE
(*   OPTION SAVE THE CHANGES IS SELECTED.  OTHERWISE ANY CHANGES
(*   MADE ARE IGNORED.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCCOMPAR *)
(**)
  PROCEDURE SCCOMPAR(VAR FIRST_NAME : T_NAME;
                    VAR SECOND_NAME : T_NAME;
                    VAR DIFFERENT : BOOLEAN);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE COMPARES TWO NAMES FOR THE LENGTH SPECIFIED *)
(*   BY 'UNIQUENESS_LENGTH' IN SCECON. THE VARIABLE 'DIFFERENT' *)
(*   IS SET TO TRUE IF THE TWO NAMES DIFFER. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   FIRST_NAME     I    THE FIRST NAME *)
(*   SECOND_NAME    I    THE SECOND NAME *)
(*   DIFFERENT      0    SET TO TRUE IF THE TWO NAMES DIFFER *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   INITIALIZE VARIABLES *)
(*   COMPARE FIRST_NAME TO SECOND_NAME CHARACTER-BY-CHARACTER *)
(*   UNTIL LENGTH IS SURPASSED OR A DIFFERENCE IS DETECTED. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE SCCONIN *)
(**)
  PROCEDURE SCCONIN(VAR IRC          : RET_REC;
                   VAR INCLD        : TEXT;
                   VAR ENTITY_LIST  : LISTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE WRITES THE ENTITY KIND CONSTANTS TO THE
(*   PASCAL INCLUDE FILE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           I/O  RETURN CODE
(*   INCLD         I    THE FILE NAME
(*   ENTITY_LIST   I    A LIST OF ENTITIES
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   WRITE THE ENTITY CONSTANT HEADING TO THE FILE
(*   WRITE 'CONST' TO THE FILE
(*   SET THE LIST OF ENTITIES TO BE READ FORWARD
(*   CONTINUE TO READ THE ENTITIES UNTIL THE END OF THE LIST
(*   GET THE KEY TO THE NEXT ENTITY IN THE LIST
(*   GET THIS ENTITY'S ADB
(*   IF THE ENTITY KIND IS AN ENTITY THEN
(*     WRITE TO THE FIELD K_<ADB.ENT.NAME> = <ADB.ENT.KIND> ;
(*   END;
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE SGC CREATE *)
(**)
PROCEDURE SGC CREATE(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR);
SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) THIS ROUTINE DETERMINES THE NEXT MENU TO DISPLAY FROM THE *)
(*) CREATE OPTION CHOSEN. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) IRC 0 RETURN CODE *)
(*) TRANS_STACK I/O POINTS TO THE TRANSACTION STACK *)
(*) *)
(*) $COMMONS: *)
(*) DEF *)
(*) INSIDE I/O INDICATES IF THE EXIT OPTION HAS *)
(*) BEEN CHOSEN WITHIN ANOTHER CREATE *)
(*) PROCEDURE *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) CALLS THE CREATE MENU INTERFACE ROUTINE (MCREATE) AND *)
(*) PROCESSES THE DATA RECEIVED FROM THE MENU EITHER BY *)
(*) CALLING THE APPROPRIATE ROUTINE OR EXITING THE PROCEDURE. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*)

```

```

(*) %INCLUDE SCDEFRCR *)
(**)
PROCEDURE SCDEFRCR(VAR IRC          : RET_REC;
                   VAR T_TYP        : TRANS_TYPE;
                   VAR TRANS_STACK   : TRANSPTR;
                   VAR CONTEXT       : T_CONTEXT);

SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*) THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(*) DEFINED TYPE ENTITY AND PUSHES THE DATA ON THE TRANS-
(*) ACTION STACK.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*)
(*) IRC O THE RETURN CODE
(*) T_TYP I THE TYPE OF TRANSACTION (EITHER CREATING
(*) A STAND-ALONE DEFINED TYPE OR A DEFINED
(*) TYPE WITHIN AN ENTITY).
(*) TRANS_STACK I/O THE TRANSACTION STACK
(*) CONTEXT I THE CONTEXT IN WHICH THE DEFINED TYPE
(*) IS BEING CREATED
(*)
(*) $COMMONS:
(*) REF
(*) INSIDE I/O INDICATES IF THE EXIT OR RETURN OPTION
(*) IS CHOSEN WITHIN ANOTHER ROUTINE
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*) THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRDEFTYP
(*) TO DISPLAY THE PANEL. THE DATA ENTERED ON THE PANEL IS
(*) VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(*) STACK OR THE APPROPRIATE ACTION IS TAKEN. IF ANY OF THE
(*) DATA ENTERED IS INVALID THEN THE PANEL IS REDISPLAYED
(*) WITH AN ERROR MESSAGE.
(*)
(*) $COMMENTS:
(*)

```

April 1990

```
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* REVISED: JANUARY 1988 C. H. MOHME DBMA *)
(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE *)
(* *)
(* REVISED: 10/09/87 C. H. MOHME DBMA *)
(* ADDED PARAMETER TO ALLOW FOR THE CREATION OF A DEFINED TYPE *)
(* FROM THE MAIN CREATE MENU. *)
(* *)
(* REVISED: 08/13/87 C. H. MOHME DBMA *)
(* ADDED LIST AND SET. NOTE: THE LIST AND SET DATA TYPES ARE *)
(* IMPLEMENTED IN THE SOFTWARE AS AN ARRAY. A FIELD WAS ADDED *)
(* TO THE ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL *)
(* ARRAY, LIST, OR SET. *)
(* *)
(* ORIGINATED: 04/22/86 L. J. BEHAN FRMI *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE SCDEFRCR *)
```

```
(* %INCLUDE SCDEFUP *)
(**)
PROCEDURE SCDEFUP(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR DEFINED_TYPE_KEY : ENTKEY;
                  VAR NUMBER_OF_USERS : INTEGER;
                  VAR DELETE_LIST    : LISTKEY;
                  VAR NEW_KEYS_LIST  : LISTKEY;
                  VAR CONTEXT        : T_CONTEXT);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE THE DEFINED TYPE
(*   ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   DEFINED_TYPE_KEY I/O  KEY OF THE ENTITY TO BE UPDATED
(*   DELETE_LIST      I/O  LIST OF ENTITIES TO BE DELETED
(*   NEW_KEYS_LIST    I/O  LIST OF ENTITIES TO BE DELETED IF
(*                       THE CHANGES MADE ARE REJECTED
(*   NUMBER_OF_USERS  I   INDICATES THE NUMBER OF USERS
(*   TRANS_STACK      I/O  POINTS TO THE TRANSACTION STACK
(*   IRC              O   RETURN CODE
(*   CONTEXT          I   THE CONTEXT IN WHICH THE DEFINED TYPE
(*                       IS BEING UPDATED
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (UPDEFTYP)
(*   TO DISPLAY THE INFORMATION ABOUT THE DEFINED TYPE ENTITY.
(*   THE USER CAN CHANGE THE NAME OR TYPE OF THE ENTITY OR
(*   SELECT ONE OF THE FOLLOWING OPTIONS ON THE PANEL :
(*       REVIEW THE CURRENT TYPE,
(*       UPDATE THE CURRENT TYPE,
```

```
(*          SAVE THE CHANGES MADE,          *)
(*          RETURN OR EXIT.                  *)
(*  IF SAVE THE CHANGES IS SELECTED AND THE NUMBER OF USERS OF  *)
(*  THE DEFINED TYPE IS ONE THEN THE OLD DEFINED TYPE KEY IS     *)
(*  PLACED ON THE DELETE LIST AND A NEW DEFINED TYPE ENTITY IS  *)
(*  CREATED.  THE NEW DEFINED TYPE KEY IS THEN PLACED ON THE    *)
(*  NEW KEYS LIST.  IF SAVE THE CHANGES IS SELECTED AND THE    *)
(*  NUMBER OF USERS OF THE DEFINED TYPE IS GREATER THAN ONE     *)
(*  THEN THE DEFINED TYPE ENTITY IS UPDATED.                   *)
(*  *)
(*  $COMMENTS:                                                  *)
(*  *)
(*  $CHANGE CONTROL:                                           *)
(*  *)
```



```
(* %INCLUDE SCENMUP *)
(**)
PROCEDURE SCENMUP(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR ENUM_KEY     : ENTKEY;
                  VAR DELETE_LIST  : LISTKEY;
                  VAR NEW_KEYS_LIST : LISTKEY);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE GATHERS THE DATA TO UPDATE THE ENUMERATION
(* ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK
(* ENUM_KEY I/O KEY OF THE ENTITY TO BE UPDATED
(* DELETE_LIST I/O LIST OF ENTITIES TO BE DELETED
(* NEW_KEYS_LIST I/O LIST OF NEWLY CREATED ENTITIES
(* IRC O RETURN CODE
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (UPENUM)
(* TO DISPLAY DATA ABOUT THE ENUMERATION ENTITY.
(* THE UPDATE OPTIONS INCLUDE THE FOLLOWING :
(* ADD AN ITEM TO THE ENUMERATION,
(* REMOVE AN ITEM FROM THE ENUMERATION,
(* SAVE THE CHANGES MADE,
(* RETURN AND EXIT.
(* THE CHANGES MADE TO THE ENUMERATION ARE KEPT ONLY IF THE
(* OPTION SAVE THE CHANGES IS SELECTED, OTHERWISE ANY CHANGES
(* THAT WERE MADE ARE IGNORED.
(*
(*)
```

CI PS560240032U
April 1990

(*	\$COMMENTS:	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```
(* %INCLUDE SCENTCR *)
(**)
  PROCEDURE SCENTCR(VAR IRC : RET_REC;
                   VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA NECESSARY TO MODEL THE
(*   ENTITY ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC            0   RETURN CODE
(*   TRANS_STACK    I/O  POINTS TO THE TRANSACTION STACK
(*
(* $COMMONS:
(*   REF
(*   INSIDE          I/O  INDICATES IF THE EXIT OR RETURN OPTION
(*                        IS CHOSEN WITHIN ANOTHER ROUTINE.
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (CRENTITY)
(*   WHICH DISPLAYS THE CREATE ENTITY PANEL.  THE NAME AND
(*   KIND NUMBER IS CHECKED FOR UNIQUENESS.  IF THEY ARE UNIQUE
(*   THEN THE DATA IS PUSHED ONTO THE TRANSACTION STACK AND
(*   THE ROUTINE (SCFLDCR) IS CALLED TO ENTER THE ENTITY'S
(*   FIELDS.  AFTER ALL OF THE FIELDS HAVE BEEN ENTERED THE
(*   TRANSACTION PROCESSING ROUTINE IS CALLED TO MODEL THE
(*   ENTITY.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE SCENTIN *)
(**)
  PROCEDURE SCENTIN(VAR IRC          : RET_REC;
                   VAR INCLD        : TEXT;
                   VAR ENTITY_LIST  : LISTKEY;
                   VAR ADB_DEFN     : LISTKEY;
                   VAR CL_DEFN      : LISTKEY);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE WRITES THE ENTITY TYPE DECLARATIONS TO THE
(*   PASCAL INCLUDE FILE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           I/O  RETURN CODE
(*   INCLD         I    THE FILE NAME
(*   ENTITY_LIST   I    A LIST OF ENTITIES
(*   ADB_DEFN      O    A LIST OF ENTITIES ADB DEFINITIONS WERE
(*                   GENERATED FOR
(*   CL_DEFN       O    A LIST OF ENTITIES CONSTITUENT DEFINI-
(*                   TIONS WERE GENERATED FOR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   WRITE TO THE FILE THE ENTITY DECLARATIONS HEADING
(*   WRITE TYPE TO THE FILE
(*   SET THE LIST OF ENTITIES TO BE READ FORWARD
(*   CONTINUE TO READ ENTITIES UNTIL THE END OF LIST IS FOUND
(*   GET THE KEY TO THE NEXT ENTITY ON THE LIST
(*   GET THE ENTITY'S ADB
(*   CALL THE PROCEDURE TO SORT THE FIELDS
(*   COUNT THE NUMBER OF ITEMS IN THE LIST OF ADB FIELDS
(*   IF THE NUMBER OF ITEMS ON THE LIST IS GREATER THAN 0 THEN
(*   WRITE TO THE FILE E_<ENTITY NAME> = RECORD
```

April 1990

```

(*)      CALL THE FIELD INCLUDE PROCEDURE                                *)
(*)      WRITE TO THE FILE END;                                           *)
(*)      ADD THE ENTITY KEY TO THE LIST OF ENTITY ADB DEFINITIONS*)
(*)      COUNT THE NUMBER OF ITEMS ON THE CONSTITUENT FIELD LIST *)
(*)      IF THE NUMBER OF ITEMS ON THE LIST IS GREATER THAN 0 THEN *)
(*)          WRITE TO THE FILE C_<ENTITY NAME> = RECORD *)
(*)          CALL THE FIELD INCLUDE PROCEDURE *)
(*)          WRITE TO THE FILE END; *)
(*)          WRITE TO THE FILE P_T_<ENTITY NAME> = RECORD *)
(*)          CALL THE FIELD INCLUDE PROCEDURE *)
(*)          WRITE TO THE FILE END; *)
(*)          WRITE TO THE FILE CONST *)
(*)          WRITE TO THE FILE P_<ENTITY NAME> = P_T_<ENTITY NAME>(*)
(*)          FOR INDEX EQUALS ONE TO NUMBER OF CONSTITUENT FIELDS DO *)
(*)              WRITE TO THE FILE INDEX *)
(*)              IF INDEX EQUALS THE NUMBER OF CONSTITUENT FIELDS THEN *)
(*)                  WRITE TO THE FILE ); *)
(*)              ELSE *)
(*)                  WRITE TO THE FILE , *)
(*)          ADD THE KEY TO THE LIST OF GENERATED CONSTITUENT *)
(*)                                  DEFINITIONS *)
(*)      END; *)
(*) *)
(*)      $COMMENTS: *)
(*) *)
(*)      $CHANGE CONTROL: *)
(*) *)
(*)      REVISED: 09/28/87          C. H. MOHME          DBMA *)
(*)      INCORPORATED THE SUPERTYPE DATA TYPE. *)
(*) *)
(*)      REVISED: 08/04/87          C. H. MOHME          DBMA *)
(*)      CHANGED NAMING CONVENTION FROM C_T_ TO S_ FOR THE TYPE DE- *)
(*)      CLARATION FOR A STRUCTURED CONSTANT FOR CONSTITUENT LIST *)
(*)      POSITIONS (IBM SPECIFIC). *)
(*) *)
(*)      REVISED: 07/30/87          C. H. MOHME          DBMA *)
(*)      CHANGED NAMING CONVENTION FROM P_ TO C_ FOR THE CONSTANT FOR *)
(*)      CONSTITUENT LIST POSITION AND C_ TO P_ FOR ENTITY TYPE *)
(*)      DECLARATIONS FOR THE CONSTITUENT REFERENCED BY THE CONSTI- *)
(*)      TUENT KEY. *)
(*) *)
(*)      REVISED: 07/30/87          C. H. MOHME          DBMA *)
(*)      CHANGED NAMING CONVENTION FROM P_T_ TO C_T_ FOR THE ENTITY *)
(*)      TYPE DECLARATION FOR THE CONSTITUENT REFERENCED BY THE CON- *)
(*)      STITUENT LIST POSITION. *)
(*) *)
(*)      REVISED: 07/22/87          C. H. MOHME          DBMA *)

```

CI PS560240032U
April 1990

```
(*      CHANGED NAMING CONVENTION FROM C_ TO K_ FOR ENTITY KIND      *)
(*      CONSTANTS AND FROM K_ TO C_ FOR ENTITY TYPE DECLARATIONS      *)
(*      FOR THE CONSTITUENTS REFERENCED BY THE CONSTITUENT KEY.      *)
(*      *)      *)
(*      ORIGINATED: 10/23/86      L. J. BEHAN      DBMA      *)
(*      *)      *)
(*-----*)
(*      *)      *)
(*END-----*)
(* END %INCLUDE SCENTIN *)
```

```

(** %INCLUDE SCENTUP *)
(**)
PROCEDURE SCENTUP(VAR IRC          : RET_REC;
                  VAR TRANS_STACK : TRANSPTR;
                  VAR ENT_KEY     : ENTKEY);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE UPDATES THE ENTITY ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    RETURN CODE
(*   TRANS_STACK   I/O  POINTS TO THE TRANSACTION STACK
(*   ENT_KEY       I/O  KEY OF THE ENTITY TO BE UPDATED
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINES TO DISPLAY
(*   THE DATA ABOUT THE ENTITY.  THE UPDATE ENTITY OPTIONS IN-
(*   CLUDE THE FOLLOWING :
(*       CHANGE THE ENTITY NAME OR KIND NUMBER,
(*       UPDATE THE ENTITY'S CONSTITUENTS (FIELDS) BY
(*       ADDING FIELDS, REMOVING FIELDS OR UPDATING FIELDS,
(*       REVIEW A FIELD,
(*       DELETE THE ENTITY,
(*       SAVE THE CHANGES MADE,
(*       RETURN OR EXIT.
(*   IF SAVE THE CHANGES WAS SELECTED THE ENTITY IS UPDATED
(*   AND THE ENTITIES ON THE DELETE LIST WILL BE DELETED IF
(*   POSSIBLE.  IF RETURN OR EXIT IS SELECTED THE ENTITIES ON
(*   THE NEW KEYS LIST WILL BE DELETED IF POSSIBLE.
(*
(* $COMMENTS:
(*

```

```
(* $CHANGE CONTROL: *)
(*)
(*) REVISD: 09/28/87 C. H. MOHME DBMA *)
(*) INCORPORATED THE SUPERTYPE DATA TYPE. *)
(*)
(*) REVISD: 08/13/87 C. H. MOHME DBMA *)
(*) CORRECTED EXIT OPTION SO THAT WHEN EXIT IS SELECTED, THE *)
(*) UPDATE MENU IS REDISPLAYED. *)
(*)
(*) REVISD: 07/28/87 C. H. MOHME DBMA *)
(*) MODIFIED TO PREVENT ATTRIBUTES FROM BEING ERRONEOUSLY REMOVED *)
(*) FROM AN ENTITY WHILE UPDATING. *)
(*)
(*) ORIGINATED: 07/01/86 L. J. BEHAN FRMI *)
(*)
(*)-----*)
(*)-----*)
(*END-----*)
(* END %INCLUDE SCENTUP *)
```



```

(*) %INCLUDE SCENUCR *)
(**)
PROCEDURE SCENUCR(VAR IRC : RET_REC;
                  VAR TRANS_STACK : TRANSPTR);
SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE *)
(*) ENUMERITEM ENTITY AND PUSHES THE DATA ON THE TRANSACTION *)
(*) STACK. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) IRC 0 THE RETURN CODE *)
(*) TRANS_STACK I/O THE TRANSACTION STACK *)
(*) *)
(*) $COMMONS: *)
(*) REF *)
(*) INSIDE I/O INDICATES IF THE EXIT OR RETURN OPTION *)
(*) IS CHOSEN WITHIN ANOTHER ROUTINE *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRENUM *)
(*) TO DISPLAY THE PANEL. THE DATA ENTERED ON THE PANEL IS *)
(*) VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION *)
(*) STACK OR THE APPROPRIATE ACTION IS TAKEN. IF ANY OF THE *)
(*) DATA ENTERED IS INVALID THEN THE PANEL IS REDISPLAYED WITH *)
(*) AN ERROR MESSAGE. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```

```

(*) %INCLUDE SCEXIST *)
(**)
  PROCEDURE SCEXIST(VAR IRC          : RET_REC;
                   VAR KIND_NUMBER : INTEGER;
                   VAR PRESENT      : BOOLEAN;
                   VAR KEY_OF_ENT   : ENTKEY);

    SUBPROGRAM;

(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE VERIFIES THE EXISTENCE OF AN ENTITY AND
(*)   RETURNS THE ENTITY KEY IF IT DOES IN FACT EXIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KIND_NUMBER    I    THE USER DEFINED KIND NUMBER OF THE
(*)                   ENTITY TO BE FOUND
(*)   PRESENT        O    INDICATES IF THE ENTITY EXISTS
(*)   KEY_OF_ENT     O    THE KEY OF THE ENTITY FOUND
(*)   IRC            O    RETURN CODE
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THIS ROUTINE IS CALLED BY A MAS EXECUTE PROCEDURE. THE
(*)   SCHEMA MODEL IS SEARCHED FOR A CLASS OR ENTITY WITH A
(*)   USER DEFINED KIND NUMBER IDENTICAL TO THE ONE PASSED IN.
(*)   IF A MATCH IS FOUND THEN THE KEY TO THE ENTITY IS RETURNED
(*)   TO THE CALLING PROCEDURE AND A FLAG IS SET TO INDICATE THAT
(*)   A MATCH WAS INDEED FOUND. IF A MATCH IS NOT FOUND THEN
(*)   THE FLAG IS SET TO FALSE.
(*)
(*) $COMMENTS:
(*)   NONE
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE SCFLDAD *)
(**)
PROCEDURE SCFLDAD(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR LIST_OF_CNSTS : LISTKEY;
                  VAR FIELDTYP     : T_FIELDTYPE;
                  VAR FIELD_KEY    : ENTKY;
                  VAR DELETE_LIST  : LISTKEY);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO ADD A FIELD TO AN ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC            0   RETURN CODE
(*   TRANS_STACK    I/O  POINTS TO THE TRANSACTION STACK
(*   LIST_OF_CNSTS  I/O  LIST OF CONSTITUENTS OF THE ENTITY
(*   FIELDTYP       I   INDICATES THE TYPE OF FIELD
(*   FIELD_KEY      0   KEY TO THE FIELD CREATED
(*   DELETE_LIST    I   LIST OF KEYS TO BE DELETED
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE ALLOWS A USER TO ADD A FIELD TO AN ENTITY
(*   DURING AN UPDATE.  THE ADD FIELD (ADDFIELD) ROUTINE IS
(*   CALLED AND THE DATA IS GATHERED.  THEN THE TRANSACTION
(*   PROCESSING ROUTINE (SCTRSFR) IS CALLED TO MODEL THE FIELD
(*   ENTITY.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: JANUARY 1988      C. H. MOHME      DBMA
(*)
```

```
(*      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE      *)
(*)
(*)      REVISED: 09/28/87      C. H. MOHME      DBMA      *)
(*)      INCORPORATED THE SUPERTYPE DATA TYPE.      *)
(*)
(*)      REVISED: 08/13/87      C. H. MOHME      DBMA      *)
(*)      CHANGED FIELD ADB DATA; ADDED LIST AND SET.  NOTE:  THE LIST *)
(*)      AND SET DATA TYPES ARE IMPLEMENTED IN THE SOFTWARE AS AN *)
(*)      ARRAY.  A FIELD WAS ADDED TO THE ARRAY ADB TO SPECIFY WHETHER *)
(*)      THE ARRAY IS A CONCEPTUAL ARRAY, LIST, OR SET.      *)
(*)
(*)      ORIGINATED: 07/21/86      L. J. BEHAN      DBMA      *)
(*)
(*)-----*)
(*)
(*)END-----*)
(* END %INCLUDE SCFLDAD *)
```

```

(** %INCLUDE SCFLDCR *)
(**)
PROCEDURE SCFLDCR(VAR IRC : RET_REC;
                  VAR FIELDTYP : T_FIELDTYPE;
                  VAR TRANS_STACK : TRANSPTR);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE GATHERS THE DATA NECESSARY TO MODEL THE
(* FIELD ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* === == =====
(* IRC 0 RETURN CODE
(* FIELDTYP I INDICATES THE TYPE OF FIELD
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK
(*
(* $COMMONS:
(* REF
(* INSIDE I/O INDICATES IF THE EXIT OR RETURN OPTION
(* IS CHOSEN WITHIN ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (CRFIELD)
(* TO DISPLAY THE CREATE FIELD PANEL. THE DATA ENTERED IS
(* VERIFIED AND IF IT IS VALID IT IS PUSHED ON TO THE TRANS-
(* ACTION STACK AND THE APPROPRIATE FIELD TYPE ROUTINE IS
(* CALLED. IF THE DATA IS INVALID THE PANEL IS REDISPLAYED
(* WITH AN ERROR MESSAGE. THIS PANEL CONTINUES TO BE DIS-
(* PLAYED UNTIL EXIT OR NO MORE FIELDS IS CHOSEN.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* REVISED: JANUARY 1988 C. H. MOHME DBMA
(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE

```

CI PS560240032U
April 1990

```
(*  REVISED: 09/29/87          C. H. MOHME          DBMA      *)
(*  INCORPORATED THE SUPERTYPE DATA TYPE.          *)
(*  *)                                              *)
(*  REVISED: 08/13/87          C. H. MOHME          DBMA      *)
(*  CHANGED FIELD ADB DATA; ADDED LIST AND SET.  NOTE:  THE LIST *)
(*  AND SET DATA TYPES ARE IMPLEMENTED IN THE SOFTWARE AS AN      *)
(*  ARRAY.  A FIELD WAS ADDED TO THE ARRAY ADB TO SPECIFY WHETHER  *)
(*  THE ARRAY IS A CONCEPTUAL ARRAY, LIST, OR SET.              *)
(*  *)                                              *)
(*  ORIGINATED: 02/05/86        L. J. BEHAN          FRMI      *)
(*  *)                                              *)
(*-----*)
(*                                              *)
(*END-----*)
(* END %INCLUDE SCFLDCR *)
```

```
(* %INCLUDE SCFLDIN *)
(**)
PROCEDURE SCFLDIN(VAR   IRC           : RET_REC;
                  VAR   INCLD        : TEXT;
                  VAR   FIELD_LIST   : LISTKEY;
                  CONST FIELD_DEF    : T_FIELD_DEF);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE WRITES THE FIELD TYPE DECLARATION TO THE *)
(* PASCAL INCLUDE FILE. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME          I/O  DESCRIPTION *)
(* ====          ==  ===== *)
(* IRC           I/O  RETURN CODE *)
(* INCLD         I    THE FILE NAME *)
(* FIELD_LIST    I    A LIST OF FIELDS *)
(* FIELD_DEF     I    INDICATES THE FIELD DEFINITION TYPE *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* SET THE LIST OF FIELDS TO BE READ FORWARD *)
(* CONTINUE TO READ FIELDS UNTIL THE END OF LIST IS FOUND *)
(* GET THE KEY TO THE NEXT FIELD *)
(* GET THE FIELD'S ADB *)
(* WRITE TO THE FILE <FIELD NAME> *)
(* IF THE TYPE OF DEFINITION IS ADB THEN *)
(* SET THE FIELD'S CONSTITUENT LIST TO BE READ FORWARD *)
(* GET THE KEY TO THE FIRST ENTITY ON THE CONSTITUENT LIST *)
(* GET THE CONSTITUENT'S ADB *)
(* CALL THE ROUTINE TO WRITE OUT THE PRIMITIVE TYPES THAT *)
(* WILL APPEAR IN THE ADB OF THE ENTITY *)
(* INTEGER *)
(* REAL *)
(* LOGICAL *)
```

CI PS560240032U
April 1990

```
(*          STRING                                *)
(*          DEFINED TYPE                          *)
(*          ARRAY                                *)
(*      ELSE                                    *)
(*          IF THE TYPE OF DEFINITION IS THE CONSTITUENT LIST *)
(*                                          POSITION THEN *)
(*          WRITE TO THE FILE T_CL_POSITION;      *)
(*      ELSE                                    *)
(*          IF THE DEFINITION TYPE IS CONSTITUENT KEY THEN *)
(*              WRITE TO THE FILE ENTKEY;          *)
(*      END;                                    *)
(*                                          *)
(* $COMMENTS:                                    *)
(*                                          *)
(* $CHANGE CONTROL:                              *)
(*                                          *)
```



```
(* %Include SCFLDLST *)
(**)
  Procedure SCFLDLST( Var  Entity_Kind_Exists : Boolean;
                    Const Ent_Kind          : Integer;
                    Var  Field_List         : Listkey   );
    Subprogram;
(**)
(*-----*)
(*
(* $Function:
(*   This routine creates a list of fields that are constituents
(*   of a specified entity type (GLOBAL, SUPERTYPE, STRUCTURE,
(*   or ENTITY).
(*
(* $Description of arguments:
(*
(*   Name          I/O  Description
(*   ====          ==  =====
(*
(*   Entity_Kind_Exists  0  Indicates whether an instance of the
(*                           specified entity type exists in the
(*                           model.  If no instances exist, then
(*                           a list of fields will NOT be created.
(*
(*   Ent_Kind           I  The specified entity kind (GLOBAL,
(*                           SUPERTYPE, STRUCTURE, or ENTITY) for
(*                           which a list of fields is to be made.
(*
(*   Field_List         0  The list of fields that are constitu-
(*                           ents of the specified entity type.
(*
(* $Commons:
(*   None
(*
(* $Environment:
(*   Language: IBM Pascal
(*   Hardware System: IBM 360/370/4341/4381
(*
(* $Execution Procedure:
(*   Internal procedure for the Schema Manager
(*
(* $Processing Description:
(*   Set ENTITY_KIND_EXISTS to FALSE.
(*   Make a list of the specified entity kind.
(*   If an instance of the specified entity kind exists in the
(*   model then
(*   Make a list of constituent fields.
```

CI PS560240032U
April 1990

```
(*          Delete list of the specified kind.          *)
(*          Set ENTITY_KIND_EXISTS to TRUE.              *)
(*                                                     *)
(* $Comments:                                           *)
(*                                                     *)
(* $Change Control:                                     *)
(*                                                     *)
(*   Originated: 01/06/88      C. H. Mohme      DBMA    *)
(*                                                     *)
(*-----*)
(*-----*)
(*End-----*)
(* End %Include SCFLDLST *)
```

```
(* %INCLUDE SCFLDSRT *)
(**)
  PROCEDURE SCFLDSRT(CONST CURRENT : ENTBLOCK;
                    CONST NEXT   : ENTBLOCK;
                    VAR  FLIP    : BOOLEAN;
                    VAR  RRC     : EXT_RET_CODE;
                    VAR  PROC    : ROUTINE);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE FINDS THE KEY AND RETURNS IT TO THE CALLING
(*   PROCEDURE GIVEN A NAME OR USER DEFINED KIND NUMBER AS WELL
(*   AS THE ENTITY KIND.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   ENTITY_KEY    I    KEY TO THE ENTITY
(*   ADB           0    DATA STORED IN THE ADB
(*   DATA         I/O  THE USER DEFINED DATA STRUCTURE USED
(*                       TO PASS DATA INTO THIS PROCEDURE AND
(*                       TO GET THE DESIRED OUTPUT FROM THE
(*                       PROCEDURE
(*   RRC           0    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY THE MAS EXECUTE ROUTINES,
(*   MAKXEQ AND MAEXEQ. THE LIST IS SEARCHED FOR IDENTICAL
(*   DATA AND IF IT IS FOUND THE ENTITY'S KEY IS RETURNED TO
(*   THE CALLING PROCEDURE AS WELL A FLAG INDICATING THAT THE
(*   KEY RETURNED IS THE CORRECT ONE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE SCFLDST *)
(**)
PROCEDURE SCFLDST(VAR   IRC           : RET_REC;
                  VAR   FIELD_LIST    : LISTKEY;
                  VAR   FIELDS_IN_ADB : LISTKEY;
                  VAR   FIELDS_IN_CL  : LISTKEY);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE SORTS THE FIELDS INTO TWO GROUPS AND THEN
(*     ORDERS THEM ACCORDING TO OFFSET OR POSITION DEPENDING ON
(*     WHETHER OR NOT THEY ARE IN THE LIST OF ADB FIELDS OR THE
(*     LIST OF CONSTITUENT FIELDS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME           I/O  DESCRIPTION
(*     ====           ==  =====
(*     IRC            I/O  RETURN CODE
(*     FIELD_LIST     I    A LIST OF FIELDS
(*     FIELDS_IN_ADB  0    A LIST OF FIELDS IN THE ADB
(*     FIELDS_IN_CL   0    A LIST OF FIELDS IN THE CONSTITUENT LIST
(*
(* $COMMONS:
(*     NONE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*     THIS ROUTINE READS EACH ENTITY FROM THE LIST AND WRITES
(*     OUT THE ENTITY NAME AND CONSTANT TO THE INCLUDE FILE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCFLDUP *)
(**)
PROCEDURE SCFLDUP(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR LIST_OF_FIELDS : LISTKEY;
                  VAR FIELDTYP     : T_FIELDTYPE;
                  VAR FIELD_KEY     : ENTKEY;
                  VAR DELETE_LIST   : LISTKEY;
                  VAR NEW_KEYS_LIST : LISTKEY);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE THE FIELD ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           O    RETURN CODE
(*   TRANS_STACK   I/O  POINTS TO THE TRANSACTION STACK
(*   LIST_OF_FIELDS I/O  LIST OF THE ENTITY'S FIELDS FROM WHICH
(*                       A FIELD IS TO BE UPDATED
(*   FIELDTYP      I    TYPE OF FIELD
(*   FIELD_KEY     I/O  KEY OF THE FIELD TO BE UPDATED
(*   DELETE_LIST   I/O  LIST OF ENTITIES TO BE DELETED
(*   NEW_KEYS_LIST I/O  LIST OF NEWLY CREATED ENTITIES
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE DATA ABOUT THE FIELD ENTITY IS DISPLAYED BY CALLING
(*   THE MENU INTERFACE ROUTINE (UPFIELD). THE USER CAN CHANGE
(*   ANY OF THE DATA ON THE PANEL OR SELECT ONE OF THE FOLLOWING
(*   OPTIONS :
(*       REVIEW THE CURRENT TYPE,
(*       UPDATE THE CURRENT TYPE,
(*       SAVE THE CHANGES MADE,
(*       RETURN OR EXIT.
```

```
(*      IF SAVE THE CHANGES IS CHOSEN AND THE USER HAS CHANGED ANY *)
(*      OF THE FIELD DATA THE FIELD KEY IS PLACED ON THE DELETE   *)
(*      LIST AND A NEW FIELD ENTITY IS MODELED.  THE KEY TO THE NEW *)
(*      FIELD ENTITY IS PLACE ON THE NEW KEYS LIST.                 *)
(*                                                                  *)
(* $COMMENTS:                                                       *)
(*                                                                  *)
(* $CHANGE CONTROL:                                                 *)
(*                                                                  *)
(*      REVISED: JANUARY 1988      C. H. MOHME                      DBMA *)
(*      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE           *)
(*                                                                  *)
(*      REVISED: 09/28/87          C. H. MOHME                      DBMA *)
(*      INCORPORATED THE SUPERTYPE DATA TYPE.                     *)
(*                                                                  *)
(*      REVISED: 08/13/87          C. H. MOHME                      DBMA *)
(*      CHANGED FIELD ADB DATA; ADDED LIST AND SET.  NOTE:  THE LIST *)
(*      AND SET DATA TYPES ARE IMPLEMENTED IN THE SOFTWARE AS AN  *)
(*      ARRAY.  A FIELD WAS ADDED TO THE ARRAY ADB TO SPECIFY WHETHER *)
(*      THE ARRAY IS A CONCEPTUAL ARRAY, LIST, OR SET.           *)
(*                                                                  *)
(*      ORIGINATED: 08/05/86        L. J. BEHAN                     DBMA *)
(*                                                                  *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE SCFLDUP *)
```

```
(* %INCLUDE SCFNDKEY *)
(**)
  PROCEDURE SCFNDKEY(CONST ENTITY_KEY : ENTKEY;
                    VAR  ADB : ENTBLOCK;
                    VAR  DATA : BLKDATA;
                    VAR  RRC : EXT_RET_CODE);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE FINDS THE KEY AND RETURNS IT TO THE CALLING
(*   PROCEDURE GIVEN A NAME OR USER DEFINED KIND NUMBER AS WELL
(*   AS THE ENTITY KIND.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   ENTITY_KEY    I    KEY TO THE ENTITY
(*   ADB           0    DATA STORED IN THE ADB
(*   DATA         I/O  THE USER DEFINED DATA STRUCTURE USED
(*                   TO PASS DATA INTO THIS PROCEDURE AND
(*                   TO GET THE DESIRED OUTPUT FROM THE
(*                   PROCEDURE
(*   RRC           0    EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY THE MAS EXECUTE ROUTINES,
(*   MAKXEQ AND MAEXEQ. THE LIST IS SEARCHED FOR IDENTICAL
(*   DATA AND IF IT IS FOUND THE ENTITY'S KEY IS RETURNED TO
(*   THE CALLING PROCEDURE AS WELL A FLAG INDICATING THAT THE
(*   KEY RETURNED IS THE CORRECT ONE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE SCGENRPT *)
(**)
  PROCEDURE SCGENRPT(VAR IRC          : RET_REC;
                    VAR REPORT_TYPE : OPERATIONS;
                    VAR MSG          : MESSAGE);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE DETERMINES THE SUBSCHEMA FOR WHICH A REPORT *)
(*   IS TO BE GENERATED AND CALLS THE APPROPRIATE ROUTINE TO *)
(*   PRODUCE THE REPORT. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   IRC           I/O  RETURN CODE *)
(*   REPORT_TYPE   I    TYPE OF REPORT TO BE GENERATED *)
(*   MSG           I/O  PANEL MESSAGE *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   DISPLAY A PANEL CONTAINING A LIST OF ALL OF THE SUBSCHEMAS *)
(*   WITHIN THE SCHEMA MODEL. *)
(*   IF RETURN OR EXIT WAS NOT CHOSEN ON THE PANEL THEN *)
(*   IF A MULTIPLE SELECT WAS MADE ON THE PANEL THEN *)
(*   DISPLAY AN ERROR MESSAGE *)
(*   ELSE *)
(*   GET THE KEY OF THE SUBSCHEMA SELECTED *)
(*   IF THE SUBSCHEMA HAS NOT BEEN PHYSICALIZED THEN *)
(*   PHYSICALIZE THE SUBSCHEMA *)
(*   CALL THE APPROPRIATE ROUTINE TO PRODUCE A REPORT *)
(*   ELSE *)
(*   IF RETURN WAS SELECTED THEN *)
(*   RETURN TO THE REPORT MENU *)
(*   ELSE *)
(*   IF EXIT WAS SELECTED THEN
```


CI PS560240032U
April 1990

```
(*          RETURN TO THE MAIN MENU                      *)
(*          ELSE                                           *)
(*          DISPLAY AN ERROR MESSAGE FOR AN INVALID OPTION *)
(*          END;                                           *)
(*          $COMMENTS:                                     *)
(*          $CHANGE CONTROL:                               *)
(*)
```

```
(* %INCLUDE SCHDVR *)
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   THIS IS THE MAINLINE PROGRAM WHICH DRIVES THE SCHEMA
(*)   EXECUTIVE PACKAGE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NONE
(*)
(*) $COMMONS:
(*)   DEF
(*)   CURRENT_LIST   I/O   POINTS TO THE LIST KEY CURRENTLY
(*)                   IN USE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)   DDNAMES USED WITH STANDARD FILES:
(*)   NONE
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERFACE PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THIS ROUTINE INITIALIZES THE NETWORK, THEN DETERMINES
(*)   IF THE PACKAGE IS TO BE USED INTERACTIVELY OR BY BATCH
(*)   AND CONTINUES PROCESSING ACCORDINGLY.
(*)
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
```

```

(** %INCLUDE SCINCLD *)
(**)
  PROCEDURE SCINCLD(VAR SUBSCHEMA_KEY : ENTKEY;
                   VAR IRC           : RET_REC);
    SUBPROGRAM;
(**)
  (*)-----*)
  (*)
  (*) $FUNCTION:
  (*)   THIS ROUTINE GENERATES THE PASCAL INCLUDE FILES AND WRITES
  (*)   THESE DEFINITIONS TO A FILE
  (*)
  (*) $DESCRIPTION OF ARGUMENTS:
  (*)   NAME      I/O  DESCRIPTION
  (*)   ====      ==  =====
  (*)   IRC       I/O  RETURN CODE
  (*)   MSG       I/O  PANEL MESSAGE
  (*)
  (*) $COMMONS:
  (*)   NONE
  (*)
  (*) $ENVIRONMENT:
  (*)   LANGUAGE: IBM PASCAL
  (*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
  (*)
  (*) $EXECUTION PROCEDURE:
  (*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
  (*)
  (*) $PROCESSING DESCRIPTION:
  (*)   INITIALIZE VARIABLES
  (*)   GET THE ADB OF THE SUBSCHEMA AND DETERMINE IF IT HAS BEEN
  (*)   PHYSICALIZED.
  (*)   IF THE SUBSCHEMA HAS NOT BEEN PHYSICALIZED, THEN PHYSICALIZE
  (*)   IT.
  (*)   WRITE THE HEADING TO THE FILE
  (*)   WRITE OUT TO THE FILE THE SUBSCHEMA COMMENT, IF ONE EXISTS
  (*)   MAKE A LIST OF ALL OF THE ENTITIES WITHIN THE SUBSCHEMA
  (*)   DELETE ANY DUPLICATE ENTRIES OFF OF THE LIST
  (*)   SORT THE LIST OF ENTITIES ALPHABETICALLY
  (*)   CALL THE ROUTINE TO WRITE THE ENTITY KIND CONSTANTS TO THE
  (*)   FILE
  (*)   CALL THE ROUTINE TO WRITE THE DEFINED TYPES TO THE FILE
  (*)   CALL THE ROUTINE TO WRITE THE ENTITY DECLARATIONS TO THE FILE
  (*)   CALL THE ROUTINE TO WRITE THE MAS DECLARATIONS TO THE FILE
  (*)   CALL THE ROUTINE TO WRITE THE CONSTITUENT DECLARATIONS TO THE
  (*)   FILE
  (*)   CHECK FOR ERROR CONDITIONS

```

CI PS560240032U
April 1990

```
(*      DELETE ALL OF THE LISTS CREATED IN THIS ROUTINE      *)
(*)
(*) $COMMENTS:                                                  *)
(*)                                                            *)
(*) $CHANGE CONTROL:                                           *)
(*)                                                            *)
(*)                                                            *)
```

```
(* %INCLUDE SCINTCR *)
(**)
  PROCEDURE SCINTCR(VAR IRC : RET_REC;
                   VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(*   INTEGER ENTITY AND PUSHES THE DATA ON THE TRANSACTION
(*   STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC            0    THE RETURN CODE
(*   TRANS_STACK    I/O  THE TRANSACTION STACK
(*
(* $COMMONS:
(*   REF
(*   INSIDE          I/O  INDICATES IF THE EXIT OPTION OR
(*                        THE RETURN OPTION IS CHOSEN WITHIN
(*                        ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRINTEGR
(*   TO DISPLAY THE PANEL. THE DATA ENTERED ON THE PANEL IS
(*   VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(*   STACK. OTHERWISE, IF THE DATA ENTERED IS INVALID THE
(*   PANEL IS REDISPLAYED.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCINTUP *)
(**)
PROCEDURE SCINTUP(VAR IRC : RET_REC;
                  VAR INTEGER_KEY : ENTKEY;
                  VAR NUMBER_OF_USERS : INTEGER;
                  VAR DELETE_LIST : LISTKEY;
                  VAR NEW_KEYS_LIST : LISTKEY);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE UPDATES THE INTEGER ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* INTEGER_KEY I/O KEY OF THE ENTITY TO BE UPDATED
(* NUMBER_OF_USERS I NUMBERS OF USERS OF THE INTEGER ENTITY
(* DELETE_LIST I/O LIST OF ENTITIES TO BE DELETED
(* NEW_KEYS_LIST I/O LIST OF NEWLY CREATED ENTITIES
(* IRC O RETURN CODE
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE DISPLAYS THE PRECISION OF THE INTEGER ENTITY
(* AND ALLOWS THE USER TO CHANGE THE INFORMATION IF HE/SHE SO
(* DESIRES. IF A CHANGE IS MADE AND THE NUMBER OF USERS OF
(* THE OLD INTEGER ENTITY IS ONE OR LESS THEN ITS KEY IS
(* PLACED ON THE DELETE LIST. AFTER THE NEW INTEGER ENTITY
(* IS CREATED ITS KEY IS PLACED ON THE NEW KEYS LIST.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE SCKEFIND *)
(**)
  PROCEDURE SCKEFIND(VAR IRC : RET_REC;
                     CONST ENT_KIND : INTEGER;
                     VAR ENT_KEY : ENTKEY;
                     VAR CREATE : BOOLEAN;
                     VAR CONTEXT : T_CONTEXT);

    SUBPROGRAM;

(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*) THIS ROUTINE GETS THE KEY TO THE ENTITY TO BE UPDATED OR*)
(*) REVIEWED. *)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === *)
(*) ENT_KIND I KIND OF THE ENTITY *)
(*) ENT_KEY O KEY OF THE ENTITY *)
(*) CREATE O INDICATES WHETHER CREATE OR REVIEW *)
(*) IRC O RETURN CODE *)
(*) CONTEXT I THE CONTEXT IN WHICH THE KEY IS TO BE *)
(*) CREATED *)
(*)*)
(*) $COMMONS: *)
(*) NONE *)
(*)*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)*)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)*)
(*) $PROCESSING DESCRIPTION: *)
(*) THIS ROUTINE MAKES A LIST OF ENTITIES OF THE KIND *)
(*) SPECIFIED AND DISPLAYS THIS LIST OF ENTITIES BY CALLING *)
(*) THE MENU INTERFACE ROUTINE (DISPLIST). IF AN ENTITY IS *)
(*) CHOSEN THEN MAKXEQ IS CALLED TO GET THE KEY TO THE ENTITY *)
(*) IF IT EXISTS. IF AN INVALID PICK IS MADE A MESSAGE IS *)
(*) DISPLAYED ON THE DISPLIST PANEL. *)
(*)*)
(*) $COMMENTS: *)
(*)*)
(*) $CHANGE CONTROL: *)
(*)*)

```

```
(* %INCLUDE SCKEYIN *)
(**)
  PROCEDURE SCKEYIN(VAR IRC          : RET_REC;
                    VAR INCLD        : TEXT;
                    VAR ENTITY_LIST  : LISTKEY;
                    VAR CL_DEFN      : LISTKEY);

    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE WRITES THE KEYBLOCK TYPE DECLARATION TO THE *)
(*   PASCAL INCLUDE FILE. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   IRC           I/O  RETURN CODE *)
(*   INCLD          I   THE FILE NAME *)
(*   ENTITY_LIST    I   A LIST OF ENTITIES *)
(*   CL_DEFN        I   A LIST OF ENTITIES WHO HAVE CONSTITUENT *)
(*                   LIST DEFINITIONS *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   WRITE THE KEYBLOCK HEADING TO THE FILE *)
(*   WRITE TYPE TO THE FILE *)
(*   WRITE OUT TO THE FILE KEYBLOCK = RECORD *)
(*   WRITE OUT TO THE FILE CASE INTEGER OF *)
(*   SET THE LIST OF ENTITIES TO BE READ IN THE FORWARD *)
(*   CONTINUE TO READ ENTITIES UNTIL THE END OF LIST IS FOUND *)
(*   GET THE KEY TO THE NEXT ENTITY ON THE LIST *)
(*   GET THE ENTITY'S ADB *)
(*   WRITE TO THE FILE C_<ENTITY NAME> : *)
(*   IF THE ENTITY IS IN THE LIST OF GENERATED CONSTITUENT *)
(*                               DEFINITIONS THEN *)
(*   WRITE TO THE FILE (<ENTITY NAME> : C_<ENTITY NAME>); *)
(*   ELSE *)
```


CI PS560240032U
April 1990

```
(*          WRITE TO THE FILE ( );          *)
(*          END;                             *)
(*                                           *)
(* $COMMENTS:                               *)
(*                                           *)
(* $CHANGE CONTROL:                         *)
(*                                           *)
```

```
(* %INCLUDE SCLISTCR *)
(**)
PROCEDURE SCLISTCR(VAR IRC : RET_REC;
                   VAR TRANS_STACK : TRANSPTR;
                   VAR CONTEXT      : T_CONTEXT);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(* LIST ENTITY AND PUSHES THE DATA ON THE TRANSACTION
(* STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(* NAME          I/O  DESCRIPTION
(* =====
(* IRC            0    THE RETURN CODE
(* TRANS_STACK    I/O  THE TRANSACTION STACK
(* CONTEXT        I    THE CONTEXT IN WHICH THE LIST IS BEING
(*                     CREATED
(*
(* $COMMONS:
(* REF
(* INSIDE          I/O  INDICATES IF THE EXIT OR RETURN OPTION
(*                     IS CHOSEN WITHIN ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRLIST
(* TO DISPLAY THE PANEL. THE DATA ENTERED ON THE PANEL IS
(* VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(* STACK OR THE APPROPRIATE ACTION IS TAKEN. IF ANY OF THE
(* DATA ENTERED IS INVALID THEN THE PANEL IS REDISPLAYED
(* WITH AN ERROR MESSAGE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCLISTUP *)
(**)
  PROCEDURE SCLISTUP(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR;
                    VAR LIST_KEY : ENTKEY;
                    VAR NUMBER_OF_USERS : LISTKEY;
                    VAR DELETE_LIST : LISTKEY;
                    VAR NEW_KEYS_LIST : LISTKEY;
                    VAR CONTEXT      : T_CONTEXT);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    RETURN CODE
(*   TRANS_STACK   I/O  POINTS TO THE TRANSACTION STACK
(*   LIST_KEY      I/O  KEY OF THE LIST TO BE UPDATED
(*   NUMBER_OF_USERS I/O  THE NUMBER OF USERS OF THE LIST
(*   DELETE_LIST   I/O  LIST OF ENTITIES TO BE DELETED
(*   NEW_KEYS_LIST I/O  LIST OF NEWLY CREATED ENTITIES
(*   CONTEXT       I    THE CONTEXT IN WHICH THE LIST IS BEING
(*                     UPDATED
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE CURRENT LIST DATA IS DISPLAYED ON THE UPLIST PANEL.
(*   THE DATA CAN THEN BE UPDATED BY THE USER.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE SCMASIN *)
(**)
  PROCEDURE SCMASIN(VAR IRC          : RET_REC;
                    VAR INCLD       : TEXT;
                    VAR ENTITY_LIST : LISTKEY;
                    VAR ADB_DEFN    : LISTKEY);

    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE WRITES THE MAS ADB DECLARATION TO THE PASCAL
(*)   INCLUDE FILE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   IRC            I/O  RETURN CODE
(*)   INCLD          I    THE FILE NAME
(*)   ENTITY_LIST    I    A LIST OF ENTITIES
(*)   ADB_DEFN       I    A LIST OF ENTITIES THAT HAVE AN ADB
(*)                   DEFINITION GENERATED
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   WRITE THE HEADING TO THE FILE
(*)   WRITE THE FOLLOWING TO THE FILE
(*)       ENTBLOCK = RECORD
(*)           KIND      : INTEGER;
(*)           LENGTH    : INTEGER;
(*)           SYSUSE     : INTEGER;
(*)           VERSION    : INTEGER;
(*)           SYS_IDENT  : INTEGER;
(*)           IDENT      : INTEGER;
(*)           CASE INTEGER OF
(*) SET THE LIST OF ENTITIES TO BE READ FORWARD
(*) CONTINUE TO READ ENTITIES UNTIL THE END OF LIST IS FOUND
(*) GET THE KEY TO THE NEXT ENTITY IN THE LIST
(*)

```

```
(*      GET THE ENTITY'S ADB                                *)
(*      WRITE TO THE FILE K_<ENTITY NAME> :                 *)
(*      IF THE ENTITY IS IN THE LIST OF GENERATED ADB DEFINITIONS *)
(*      WRITE TO THE FILE (<ENTITY NAME> : E_<ENTITY NAME>); *)
(*      ELSE                                                *)
(*      WRITE TO THE FILE ();                               *)
(*      END;                                                *)
(*      $COMMENTS:                                         *)
(*      $CHANGE CONTROL:                                   *)
(*)
```

```
(* %INCLUDE SCMEMAD *)
(**)
PROCEDURE SCMEMAD(VAR IRC : RET_REC;
                  VAR ENT_KEY: ENTKEY;
                  VAR LIST_OF_CNSTS : LISTKEY;
                  VAR CHOSEN_KEY : ENTKEY);
SI'SPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE DISPLAYS A LIST OF MEMBERS FROM WHICH THE *)
(* USER CAN SELECT. THE ENTITY KEY OF THE MEMBER SELECTED IS *)
(* RETURNED. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* IRC 0 THE INTERNAL RETURN CODE INDICATING *)
(* WHETHER AN ERROR HAS OCCURRED. *)
(* ENT_KEY I THE KEY OF THE ENTITY TO WHICH MEMBERS *)
(* ARE BEING ADDED *)
(* LIST_OF_CNSTS I THE LIST OF ALL CONSTITUENTS *)
(* CHOSEN_KEY 0 THE ENTITY KEY OF THE MEMBER SELECTED *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* THIS ROUTINE BEGINS WITH THE CONSTITUENT LIST AND REMOVES *)
(* FROM IT THOSE CONSTITUENTS WHICH ARE ALREADY MEMBERS. IT *)
(* ALSO REMOVES THE CONSTITUENT TO WHICH MEMBERS ARE BEING *)
(* ADDED. THE PANEL DISPLIST DISPLAYS THE MEMBERS AND THE *)
(* USER CAN SELECT ONE. THE ENTITY KEY OF THE MEMBER SE- *)
(* LECTED, IF ANY, IS RETURNED. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE SCMEMLST *)
(**)
  PROCEDURE SCMEMLST(VAR IRC          : RET_REC;
                    VAR TRANS_STACK : TRANSPTR;
                    VAR ENT_TYP      : ENTITY_TYPE);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE LISTS OUT THE CURRENT MEMBERS/CONSTITUENTS OF
(*   THE ENTITY, CLASS, SUBSCHEMA, POINTER, STRUCTURE, GLOBAL
(*   FIELD AND ENUMERATION ENTITIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           O    INTERNAL RETURN CODE
(*   TRANS_STACK   I/O  POINTER TO THE TRANSACTION STACK
(*   ENT_TYP       I    THE ENTITY TYPE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE SEARCHES THROUGH THE TRANSACTION STACK FOR
(*   THE CONSTITUENTS OF THE ENTITY TYPE PASSED IN.  THE CON-
(*   STITUENT NAMES ARE PLACED IN AN ARRAY THAT IS DISPLAYED
(*   BY CALLING THE MENU INTERFACE ROUTINE (LMEM23).
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCPOPKE *)
(**)
  PROCEDURE SCPOPKE(VAR IRC : RET_REG);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE 'POPS' A KEY OFF OF THE KEY STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC            0    THE RETURN CODE
(*
(* $COMMONS:
(*   REF
(*   CURRENT_LIST   0    LIST KEY POPPED OFF OF THE KEY STACK
(*   KEY_STACK      I/O  POINTER TO THE KEY STACK
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   IF THE KEY STACK IS NIL THEN AN ERROR MESSAGE IS RETURNED
(*   TO THE CALLING PROCEDURE.  OTHERWISE, THE KEY DATA IS
(*   'POPPED' OFF OF THE STACK AND THE NODE IS DISPOSED OF BY
(*   CALLING THE BUILT IN PASCAL PROCEDURE 'DISPOSE'.
(*
(* $COMMENTS:
(*   TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND
(*   THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*
(* $CHANGE CONTROL:
(*
```



```

(*) %INCLUDE SCOPTR *)
(**)
  PROCEDURE SCOPTR(VAR IRC : RET_REC;
                  VAR TRANS_STACK : TRANSPTR;
                  VAR DATA : TRANSACTION);
    SUBPROGRAM;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*)   THIS ROUTINE 'POPS' THE TRANSACTION DATA OFF OF THE*)
(*)   DYNAMICALLY ALLOCATED STACK. *)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   IRC           0    INTERNAL RETURN CODE *)
(*)   TRANS_STACK   I/O  POINTS TO THE TRANSACTION STACK *)
(*)   DATA         0    THE TRANSACTION DATA *)
(*)*)
(*) $COMMONS: *)
(*)   NONE *)
(*)*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)*)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)*)
(*) $PROCESSING DESCRIPTION: *)
(*)   IF THE TRANSACTION STACK IS NIL THEN AN ERROR MESSAGE IS *)
(*)   RETURNED TO THE CALLING PROCEDURE. OTHERWISE, THE TRANS- *)
(*)   ACTION DATA IS 'POPPED' OFF OF THE STACK AND THE NODE *)
(*)   IS DISPOSED OF BY CALLING THE BUILT IN PASCAL PROCEDURE *)
(*)   DISPOSE. *)
(*)*)
(*) $COMMENTS: *)
(*)*)
(*) $CHANGE CONTROL: *)
(*)*)

```

```

(*) %INCLUDE SCPRMFL *)
(**)
  PROCEDURE SCPRMFL(VAR IRC           : RET_REC;
                    VAR LIST_OF_ENTITIES : LISTKEY;
                    VAR LISTARRAY       : T_ARRAYTV;
                    VAR INDEX           : INTEGER);
    SUBPROGRAM;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*)   THIS ROUTINE FILLS AN ARRAY WITH THE NAMES OF THE ENTITIES *)
(*)   IN THE LIST OF ENTITIES. *)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS:*)
(*)   NAME           I/O DESCRIPTION *)
(*)   ====           == =====*)
(*)   IRC             0  INTERNAL RETURN CODE *)
(*)   LIST_OF_ENTITIES I  LIST OF ENTITIES TO BE PUT IN ARRAY *)
(*)   LISTARRAY       I/O ARRAY CONTAINING THE ENTITY NAMES *)
(*)   INDEX           I/O INDEX OF THE CURRENT ARRAY POSITION *)
(*)*)
(*) $COMMONS:*)
(*)   NONE *)
(*)*)
(*) $ENVIRONMENT:*)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)*)
(*) $EXECUTION PROCEDURE:*)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)*)
(*) $PROCESSING DESCRIPTION:*)
(*)   THE LIST OF ENTITIES PASSED IN IS TRAVERSED AND EACH *)
(*)   ENTITY'S ADB IS RETRIEVED. THE ENTITY'S ADB DATA IS *)
(*)   TRANSLATED INTO CHARACTER FORMAT IF NECESSARY, AND IS *)
(*)   PLACED INTO THE LISTARRAY. THIS PROCESS CONTINUES UNTIL *)
(*)   THE END OF THE LIST IS ENCOUNTERED OR THE MAXIMUM ARRAY *)
(*)   SIZE IS EXCEEDED. *)
(*)*)
(*) $COMMENTS:*)
(*)*)
(*) $CHANGE CONTROL:*)
(*)*)

```

```
(* %INCLUDE SCPRMRE *)
(**)
  PROCEDURE SCPRMRE(VAR IRC      : RET_REC;
                    VAR ENT_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO REVIEW THE ENTITIES AND
(*   CALLS THE MENU INTERFACE ROUTINES TO DISPLAY THIS DATA.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   IRC        O   INTERNAL RETURN CODE
(*   ENT_KEY     I   KEY OF THE ENTITY TO BE REVIEWED
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   FIRST THE ADB OF THE ENTITY PASSED IN MUST BE RETRIEVED,
(*   THEN ACCORDING TO THE ENTITY'S TYPE THE ADB DATA IS TRANS-
(*   LATED INTO CHARACTER FORMAT IF IT IS NECESSARY. IF THE
(*   ENTITY HAS CONSTITUENTS THAT ARE TO BE DISPLAYED WITH THE
(*   ENTITY'S DATA, THEN THE DATA ABOUT THE CONSTITUENTS MUST
(*   BE PUT INTO A FORMAT THE CAN BE DISPLAYED ALSO. THEN THE
(*   APPROPRIATE MENU INTERFACE ROUTINE IS CALLED TO DISPLAY
(*   THIS INFORMATION.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: JANUARY 1988      C. H. MOHME      DBMA
(*   MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE
(*
(*   REVISED: 09/28/87         C. H. MOHME      DBMA
```

```
(*      INCORPORATED THE SUPERTYPE DATA TYPE.                  *)
(*)                                                                *)
(*)  REVISED: 08/13/87          C. H. MOHME                      DBMA  *)
(*)  ADDED LIST AND SET.  NOTE:  THE LIST AND SET DATA TYPES ARE  *)
(*)  IMPLEMENTED IN THE SOFTWARE AS AN ARRAY.  A FIELD WAS ADDED   *)
(*)  TO THE ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL *)
(*)  ARRAY, LIST, OR SET.                                          *)
(*)                                                                *)
(*)  ORIGINATED: 05/29/86      L. J. BEHAN                      FRMI  *)
(*)                                                                *)
(*)-----*)
(*)                                                                *)
(*)END-----*)
(* END %INCLUDE SCPRMRE *)
```

```
(* %INCLUDE SCPTRCR *)
(**)
PROCEDURE SCPTRCR(VAR IRC : RET_REC;
                  VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(*     POINTER ENTITY AND PUSHES THE DATA ON THE TRANSACTION
(*     STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC            0    THE RETURN CODE
(*     TRANS_STACK    I/O  THE TRANSACTION STACK
(*
(* $COMMONS:
(*     REF
(*     INSIDE          I/O  INDICATES IF THE EXIT OR RETURN OPTION
(*                          IS CHOSEN WITHIN ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*     THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRPNTR
(*     TO DISPLAY THE PANEL.  THE DATA ENTERED ON THE PANEL IS
(*     VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(*     STACK OR THE APPROPRIATE ACTION IS TAKEN.  IF ANY OF THE
(*     DATA IS INVALID THE PANEL IS REDISPLAYED WITH AN ERROR
(*     MESSAGE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE SCPTRUP *)
(**)
  PROCEDURE SCPTRUP(VAR IRC          : RET_REC;
                    VAR POINTER_KEY  : ENTKEY;
                    VAR NUMBER_OF_USERS : INTEGER;
                    VAR DELETE_LIST   : LISTKEY;
                    VAR NEW_KEYS_LIST : LISTKEY);

    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE THE POINTER ENTITY. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   POINTER_KEY   I/O  KEY OF THE ENTITY TO BE UPDATED *)
(*   NUMBER_OF_USERS I  THE NUMBER OF USERS OF THIS ENTITY *)
(*   DELETE_LIST   I/O  LIST OF ENTITIES TO DELETE IF THE *)
(*                   CHANGES MADE ARE SAVED *)
(*   NEW_KEYS_LIST I/O  LIST OF ENTITIES JUST CREATED DURING *)
(*                   THE UPDATE PROCESS *)
(*   IRC           0    RETURN CODE *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (UPPNTR) *)
(*   WHICH DISPLAYS INFORMATION ABOUT THE POINTER ENTITY. THE *)
(*   UPDATE POINTER OPTIONS INCLUDE THE FOLLOWING : *)
(*   UPDATE THE CONSTITUENT LIST BY ADDING OR REMOVING AN *)
(*   ELEMENT, *)
(*   REVIEW A CONSTITUENT, *)
(*   SAVE THE CHANGES MADE, *)
(*   RETURN AND EXIT. *)
(*   IF THE CONSTITUENT LIST WAS CHANGED AND NO MORE CHANGES *)
(*   WAS SELECTED THE OLD POINTER ENTITY KEY IS PLACED ON THE *)
(*   DELETE LIST AND A NEW POINTER ENTITY IS MODELED. THE NEW *)
```

CI PS560240032U
April 1990

```
(*      POINTER ENTITY KEY IS PLACED ON THE NEW KEYS LIST.      *)
(*                                                                *)
(* $COMMENTS:                                                    *)
(*                                                                *)
(* $CHANGE CONTROL:                                              *)
(*                                                                *)
(*                                                                *)
```

```

(*) %INCLUDE SCPUSHKE *)
(**)
  PROCEDURE SCPUSHKE(VAR IRC : RET_REC);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE 'PUSHES' THE KEY DATA ON TO THE DYNAMICALLY
(*)   ALLOCATED KEY STACK.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   IRC            0    THE RETURN CODE
(*)
(*) $COMMONS:
(*)   REF
(*)   CURRENT_LIST  I/O  A KEY TO THE LIST CURRENTLY IN USE
(*)   KEY_STACK     I/O  POINTER TO THE KEY STACK
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   A NODE IS DYNAMICALLY ALLOCATED BY CALLING THE BUILT IN
(*)   PROCEDURE 'NEW'. THE KEY DATA IS THEN 'PUSHED' ONTO THE
(*)   KEY STACK.
(*)
(*) $COMMENTS:
(*)   TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND
(*)   THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*)
(*) $CHANGE CONTROL:
(*)

```



```

(** %INCLUDE SCPUSHTR *)
(**)
PROCEDURE SCPUSHTR(VAR IRC : RET_REG;
                   VAR TRANS_STACK : TRANSPTR;
                   CONST DATA : TRANSACTION);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE 'PUSHES' THE TRANSACTION DATA ON TO THE *)
(* DYNAMICALLY ALLOCATED STACK. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* === === *)
(* IRC 0 INTERNAL RETURN CODE *)
(* TRANS_STACK I/O POINTER TO THE TRANSACTION STACK *)
(* DATA I THE TRANSACTION DATA *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* A NODE IS ALLOCATED BY CALLING THE BUILT IN PASCAL *)
(* PROCEDURE 'NEW'. THE TRANSACTION DATA IS THEN 'PUSHED' *)
(* ONTO THE STACK. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* . *)

```

```
(* %INCLUDE SCRELGR *)
(**)
  PROCEDURE SCRELGR(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(*    REAL ENTITY AND PUSHES THE DATA ON THE TRANSACTION
(*    STACK.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME          I/O  DESCRIPTION
(*    ====          ==  =====
(*    IRC           0    THE RETURN CODE
(*    TRANS_STACK   I/O  THE TRANSACTION STACK
(*
(*  $COMMONS:
(*    REF
(*    INSIDE        I/O  INDICATES IF THE EXIT OPTION OR
(*                        THE RETURN OPTION IS CHOSEN WITHIN
(*                        ANOTHER ROUTINE
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(*  $PROCESSING DESCRIPTION:
(*    THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRREAL
(*    TO DISPLAY THE PANEL.  THE DATA ENTERED ON THE PANEL IS
(*    VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(*    STACK.  OTHERWISE, IF THE DATA ENTERED IS INVALID THE
(*    PANEL IS REDISPLAYED.
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCRELUP *)
(**)
PROCEDURE SCRELUP(VAR IRC : RET_REC;
                  VAR REAL_KEY : ENTKEY;
                  VAR NUMBER_OF_USERS : INTEGER;
                  VAR DELETE_LIST : LISTKEY;
                  VAR NEW_KEYS_LIST : LISTKEY);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE UPDATES THE REAL ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   REAL_KEY       I/O  KEY OF THE ENTITY TO BE UPDATED
(*   NUMBER_OF_USERS I   NUMBER OF USERS OF THE REAL ENTITY
(*   DELETE_LIST    I/O  LIST OF ENTITIES TO BE DELETED
(*   NEW_KEYS_LIST  I/O  LIST OF NEWLY CREATED ENTITIES
(*   IRC           0    RETURN CODE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE DISPLAYS THE PRECISION OF THE REAL ENTITY AND
(*   ALLOWS THE USER TO CHANGE THE INFORMATION IF HE/SHE SO
(*   DESIRES. IF A CHANGE IS MADE AND THE NUMBER OF USERS OF
(*   THE OLD REAL ENTITY IS ONE OR LESS THEN ITS KEY IS PLACED
(*   ON THE DELETE LIST. AFTER THE NEW REAL ENTITY IS CREATED
(*   THIS KEY IS PLACED ON THE NEW KEYS LIST.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE SCREVIEW *)
(**)
  PROCEDURE SCREVIEW(VAR IRC          : RET_REC;
                    VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE DETERMINES THE NEXT MENU TO DISPLAY FROM THE
(*)   EDIT OPTION CHOSEN.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   IRC            0    RETURN CODE
(*)   TRANS_STACK   I/O  POINTS TO THE TRANSACTION STACK
(*)
(*) $COMMONS:
(*)   DEF
(*)       INSIDE      I/O  INDICATES IF THE EXIT OPTION HAS
(*)                       BEEN CHOSEN WITHIN ANOTHER CREATE
(*)                       PROCEDURE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CALLS THE REVIEW MENU INTERFACE ROUTINE (MREVIEW) AND
(*)   PROCESSES THE DATA RECIEVED FROM THE MENU EITHER BY
(*)   CALLING THE APPROPRIATE ROUTINE OR EXITING THE PROCEDURE.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE SCRPTM *)
(**)
  PROCEDURE SCRPTM(VAR IRC      : RET_REC;
                   VAR XREFFILE : TEXT);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE DETERMINES THE REPORT OPTION SELECTED AND *)
(*   CALLS THE ROUTINE WHICH GENERATES THE REPORT. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  ===== *)
(*   IRC        0   RETURN CODE *)
(*   XREFFILE    0   CROSS REFERENCE REPORT FILE *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   CALL THE MENU INTERFACE ROUTINE TO DISPLAY THE REPORT *)
(*   MENU (RPTMENU) AND THEN CALL THE APPROPRIATE ROUTINE *)
(*   TO GENERATED THE REPORT. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE SCSETCR *)
(**)
  PROCEDURE SCSETCR(VAR IRC          : RET_REC;
                    VAR TRANS_STACK : TRANSPTR;
                    VAR CONTEXT     : T_CONTEXT);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(*   SET ENTITY AND PUSHES THE DATA ON THE TRANSACTION
(*   STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           0    THE RETURN CODE
(*   TRANS_STACK   I/O  THE TRANSACTION STACK
(*   CONTEXT       I    THE CONTEXT IN WHICH THE SET IS BEING
(*                      CREATED
(*
(* $COMMONS:
(*   REF
(*   INSIDE        I/O  INDICATES IF THE EXIT OR RETURN OPTION
(*                      IS CHOSEN WITHIN ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRSET
(*   TO DISPLAY THE PANEL.  THE DATA ENTERED ON THE PANEL IS
(*   VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(*   STACK OR THE APPROPRIATE ACTION IS TAKEN.  IF ANY OF THE
(*   DATA ENTERED IS INVALID THEN THE PANEL IS REDISPLAYED
(*   WITH AN ERROR MESSAGE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE SCSETUP *)
(**)
  PROCEDURE SCSETUP(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR;
                    VAR SET_KEY : ENTKEY;
                    VAR NUMBER_OF_USERS : LISTKEY;
                    VAR DELETE_LIST : LISTKEY;
                    VAR NEW_KEYS_LIST : LISTKEY;
                    VAR CONTEXT      : T_CONTEXT);

    SUBPROGRAM;

(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*) THIS ROUTINE GATHERS THE DATA TO UPDATE A SET. *)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) IRC 0 RETURN CODE *)
(*) TRANS_STACK I/O POINTS TO THE TRANSACTION STACK *)
(*) SET_KEY I/O KEY OF THE SET TO BE UPDATED *)
(*) NUMBER_OF_USERS I/O THE NUMBER OF USERS OF THE LIST *)
(*) DELETE_LIST I/O LIST OF ENTITIES TO BE DELETED *)
(*) NEW_KEYS_LIST I/O LIST OF NEWLY CREATED ENTITIES *)
(*) CONTEXT I THE CONTEXT IN WHICH THE SET IS BEING *)
(*) UPDATED *)
(*)*)
(*) $COMMONS: *)
(*) NONE *)
(*)*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)*)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)*)
(*) $PROCESSING DESCRIPTION: *)
(*) THE CURRENT SET DATA IS DISPLAYED ON THE UPSET PANEL. *)
(*) THE DATA CAN THEN BE UPDATED BY THE USER. *)
(*)*)
(*) $COMMENTS: *)
(*)*)
(*) $CHANGE CONTROL: *)
(*)*)

```

```
(* %INCLUDE SCSRTPOS *)
(**)
  PROCEDURE SCSRTPOS(CONST CURRENT : ENTBLOCK;
                    CONST NEXT   : ENTBLOCK;
                    VAR  FLIP    : BOOLEAN;
                    VAR  RRC     : EXT_RET_CODE;
                    VAR  PROC    : ROUTINE);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE IS USED BY THE MAS MALSRT ROUTINE TO SORT
(*   CONSTITUENT LIST ATTRIBUTES BY POSITION NUMBER
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   ENTITY_KEY    I    KEY TO THE ENTITY
(*   ADB           0    DATA STORED IN THE ADB
(*   DATA         I/O  THE USER DEFINED DATA STRUCTURE USED
(*                   TO PASS DATA INTO THIS PROCEDURE AND
(*                   TO GET THE DESIRED OUTPUT FROM THE
(*                   PROCEDURE
(*   RRC           0    EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY THE MAS EXECUTE ROUTINES,
(*   MAKXEQ AND MAEXEQ. THE LIST IS SEARCHED FOR IDENTICAL
(*   DATA AND IF IT IS FOUND THE ENTITY'S KEY IS RETURNED TO
(*   THE CALLING PROCEDURE AS WELL A FLAG INDICATING THAT THE
(*   KEY RETURNED IS THE CORRECT ONE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```



```

(** %INCLUDE SCSTCUP *)
(**)
  PROCEDURE SCSTCUP(VAR IRC          : RET_REC;
                   VAR TRANS_STACK  : TRANSPTR;
                   VAR STRUCTURE_KEY : ENTKEY;
                   VAR DELETE_LIST   : LISTKEY;
                   VAR NEW_KEYS_LIST : LISTKEY;
                   VAR CONTEXT       : T_CONTEXT);

    SUBPROGRAM;

(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE THE STRUCTURE *)
(*   ENTITY. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ===  ===== *)
(*   IRC           0    RETURN CODE *)
(*   TRANS_STACK   I/O  POINTS TO THE TRANSACTION STACK *)
(*   STRUCTURE_KEY I/O  KEY OF THE ENTITY TO BE UPDATED *)
(*   DELETE_LIST   I/O  LIST OF ENTITIES TO BE DELETED *)
(*   NEW_KEYS_LIST I/O  LIST OF NEWLY CREATED ENTITIES *)
(*   CONTEXT       I    THE CONTEXT IN WHICH THE DEFINED TYPE *)
(*                   IS BEING CREATED. *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   THIS ROUTINE UPDATES THE FIELDS IN A STRUCTURE. FIELDS *)
(*   CAN BE ADDED, REMOVED, REVIEWED, AND UPDATED. IF ANY *)
(*   CHANGES WERE MADE TO THE STRUCTURE ENTITY'S CONSTITUENTS *)
(*   THE OLD STRUCTURE ENTITY'S KEY IS PLACED ON THE DELETE LIST *)
(*   AND A NEW STRUCTURE ENTITY IS MODELED AND ITS KEY IS PLACED *)
(*   ON THE NEW KEYS LIST. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)

```

```
(* %INCLUDE SCSTGCR *)
(**)
PROCEDURE SCSTGCR(VAR IRC : RET_REC;
                  VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE GATHERS THE DATA NECESSARY TO CREATE THE
(*     STRING ENTITY AND PUSHES THE DATA ON THE TRANSACTION
(*     STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     IRC           0    THE RETURN CODE
(*     TRANS_STACK   I/O  THE TRANSACTION STACK
(*
(* $COMMONS:
(*     REF
(*     INSIDE        I/O  INDICATES IF THE EXIT OPTION OR
(*                        THE RETURN OPTION IS CHOSEN WITHIN
(*                        ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*     THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE CRSTRING
(*     TO DISPLAY THE PANEL.  THE DATA ENTERED ON THE PANEL IS
(*     VERIFIED AND IF IT IS OK IT IS PUSHED ON THE TRANSACTION
(*     STACK.  OTHERWISE, IF THE DATA ENTERED IS INVALID THE
(*     PANEL IS REDISPLAYED.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCSTGUP *)
(**)
  PROCEDURE SCSTGUP(VAR IRC : RET_REC;
                    VAR STRING_KEY : ENTKEY;
                    VAR NUMBER_OF_USERS : INTEGER;
                    VAR DELETE_LIST : LISTKEY;
                    VAR NEW_KEYS_LIST : LISTKEY);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE UPDATES THE STRING ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   STRING_KEY    I/O  KEY OF THE ENTITY TO BE UPDATED
(*   NUMBER_OF_USERS  I  NUMBER OF USERS OF THE STRING ENTITY
(*   DELETE_LIST    I/O  LIST OF ENTITIES TO BE DELETED
(*   NEW_KEYS_LIST  I/O  LIST OF NEWLY CREATED ENTITIES
(*   IRC           0    RETURN CODE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE DISPLAYS THE LENGTH OF THE STRING ENTITY IN
(*   BYTES AND ALLOWS THE USER TO CHANGE THE INFORMATION IF HE/
(*   SHE SO DESIRES. IF A CHANGE IS MADE AND THE NUMBER OF
(*   USERS OF THE OLD STRING ENTITY IS ONE OR LESS THEN ITS KEY
(*   IS PLACED ON THE DELETE LIST. AFTER THE NEW STRING ENTITY
(*   IS CREATED ITS KEY IS PLACED ON THE NEW KEYS LIST.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCSUBCR *)
(**)
  PROCEDURE SCSUBCR(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA NEEDED TO MODEL THE
(*   SUBSCHEMA ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC            0    INTERNAL RETURN CODE
(*   TRANS_STACK    I/O  POINTS TO THE TRANSACTION STACK
(*
(* $COMMONS:
(*   REF
(*   INSIDE          I/O  INDICATES IF THE EXIT OR RETURN OPTION
(*                        HAS BEEN CHOSEN WITHIN ANOTHER ROUTINE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THE MENU INTERFACE ROUTINE (CRSUBSCM) IS CALLED TO DISPLAY
(*   THE CREATE SUBSCHEMA PANEL.  THE DATA GATHERED IS VERIFIED
(*   AND IF IT IS VALID IT IS PUSHED ON TO THE TRANSACTION
(*   STACK OR THE APPROPRIATE ACTION IS TAKEN.  IF ANY OF THE
(*   DATA IS INVALID THE PANEL IS REDISPLAYED WITH AN ERROR
(*   MESSAGE2.  THIS PANEL CONTINUES TO BE DISPLAYED UNTIL EXIT
(*   OR NO MORE MEMBERS IS CHOSEN.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE SCSUBUP *)
(**)
  PROCEDURE SCSUBUP(VAR IRC          : RET_REC;
                   VAR SUBSCHEMA_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GATHERS THE DATA TO UPDATE THE SUBSCHEMA
(*   ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   SUBSCHEMA_KEY I    KEY OF THE ENTITY TO BE UPDATED
(*   IRC           0    RETURN CODE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINES (UPSUB1 AND
(*   UPSUB2) WHICH DISPLAY THE DATA DESCRIBING THE SUBSCHEMA.
(*   THE UPDATE SUBSCHEMA OPTIONS INCLUDE :
(*       CHANGE THE SUBSCHEMA NAME,
(*       UPDATE THE CONSTITUENT LIST BY ADDING/
(*                               REMOVING ELEMENTS,
(*       REVIEW A CONSTITUENT,
(*       DELETE THE SUBSCHEMA,
(*       SAVE THE CHANGES MADE,
(*       RETURN AND EXIT.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

(* %INCLUDE SCSUPCR *)

(**)

```
PROCEDURE SCSUPCR(VAR IRC           : RET_REC;
                  VAR TRANS_STACK   : TRANSPTR;
                  VAR CREATE_ONLY    : BOOLEAN;
                  VAR BUILD          : BOOLEAN;
                  VAR SUPERTYPE_KEY  : ENTKEY);
```

SUBPROGRAM;

(**)

```
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE GATHERS THE DATA NECESSARY TO MODEL THE *)
(* SUPERTYPE ENTITY. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === *)
(* IRC 0 RETURN CODE *)
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK *)
(* CREATE_ONLY I INDICATES IF SUPERTYPE IS TO BE ONLY *)
(* CREATED OR IF ONE ALREADY EXISTING CAN *)
(* BE REFERENCED. *)
(* *)
(* $COMMONS: *)
(* REF *)
(* INSIDE I/O INDICATES IF THE EXIT OR RETURN OPTION *)
(* IS CHOSEN WITHIN ANOTHER ROUTINE. *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* THIS ROUTINE CALLS THE MENU INTERFACE ROUTINE (CRSUPTYP) *)
(* WHICH DISPLAYS THE CREATE SUPERTYPE PANEL. THE NAME IS *)
(* CHECKED FOR US CHECKED FOR UNIQUENESS. IF THEY ARE UNIQUE *)
(* THEN THE DATA IS PUSHED ONTO THE TRANSACTION STACK AND *)
(* THE ROUTINE (SCFLDCR) IS CALLED TO ENTER THE ENTITY'S *)
(* FIELDS. AFTER ALL OF THE FIELDS HAVE BEEN ENTERED THE *)
(* TRANSACTION PROCESSING ROUTINE IS CALLED TO MODEL THE *)
(* ENTITY. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
```

```
(* %INCLUDE SCSUPUP *)
(**)
PROCEDURE SCSUPUP(VAR IRC           : RET_REC;
                  VAR TRANS_STACK    : TRANSPTR;
                  VAR SUP_KEY        : ENTKEY;
                  VAR REMOVE_SUPERTYPE : BOOLEAN);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE UPDATES THE SUPERTYPE ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME           I/O  DESCRIPTION
(*   ====           ==  =====
(*   IRC             0   RETURN CODE
(*   TRANS_STACK     I/O  POINTS TO THE TRANSACTION STACK
(*   SUP_KEY         I/O  KEY OF THE ENTITY TO BE UPDATED
(*   REMOVE_SUPERTYPE I/O  INDICATES IF SUPERTYPE CAN BE REMOVED
(*                        OR IF ONLY THE REFERENCE TO THE SUPER-
(*                        TYPE CAN BE REMOVED.
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE CALLS THE MENU INTERFACE ROUTINES TO DISPLAY
(*   THE DATA ABOUT THE SUPERTYPE. THE UPDATE SUPERTYPE OPTIONS
(*   INCLUDE THE FOLLOWING:
(*       CHANGE THE SUPERTYPE NAME,
(*       UPDATE THE SUPERTYPE'S CONSTITUENTS (FIELDS) BY
(*       ADDING FIELDS, REMOVING FIELDS OR UPDATING FIELDS,
(*       REVIEW A FIELD,
(*       DELETE THE SUPERTYPE,
(*       SAVE THE CHANGES MADE,
(*       RETURN OR EXIT.
```

CI PS560240032U
April 1990

```
(*      IF SAVE THE CHANGES WAS SELECTED THE SUPERTYPE IS UPDATED *)
(*      AND THE SUPERTYPES ON THE DELETE LIST WILL BE DELETED IF *)
(*      POSSIBLE.  IF RETURN OR EXIT IS SELECTED THE SUPERTYPES ON *)
(*      THE NEW KEYS LIST WILL BE DELETED IF POSSIBLE. *)
(*)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
```



```
(* %INCLUDE SCTSPR *)
(**)
PROCEDURE SCTSPR(VAR IRC : RET_REC;
                 VAR TRANS_STACK : TRANSPTR);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE BEGINS THE PROCESSING OF THE TRANSACTION
(* STACK.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* === ===
(* IRC 0 RETURN CODE
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK
(*
(* $COMMONS:
(* REF
(* CURRENT_LIST I/O POINTS TO THE LIST OF KEYS
(* CURRENTLY IN USE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE CALLS SCPOPTR TO POP THE TRANSACTION STACK.
(* THEN EACH TRANSACTION IS PROCESSED ACCORDING TO ITS TYPE
(* BY CALLING THE APPROPRIATE ROUTINE TO MODEL THE ENTITIES.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* REVISED: JANUARY 1988 C. H. MOHME DBMA
(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE
(*
(* REVISED: 10/09/87 C. H. MOHME DBMA
(* CHANGED TO ALLOW FOR THE CREATION OF A DEFINED TYPE FROM THE
(* MAIN CREATE MENU.
(*
```

CI PS560240032U
April 1990

(* REVISED: 09/28/87 C. H. MOHME DBMA *)
(* INCORPORATED THE SUPERTYPE DATA TYPE. *)
(* *)
(* ORIGINATED: 02/04/86 L. J. BEHAN FRMI *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE SCTRSPR *)

```

(*) %INCLUDE SCTYPIN *)
(**)
  PROCEDURE SCTYPIN(VAR IRC          : RET_REC;
                    VAR INCLD        : TEXT;
                    VAR SUBSCHEMA_KEY : ENTKEY);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE WRITES THE DEFINED TYPE DECLARATIONS TO THE
(*)   PASCAL INCLUDE FILE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   IRC            I/O  RETURN CODE
(*)   INCLD           I   THE FILE NAME
(*)   SUBSCHEMA_KEY  I   THE KEY TO THE SUBSCHEMA
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   WRITE A DESCRIPTION OF THE BASIC TYPES AND HOW THEY ARE
(*)                                     IMPLEMENTED
(*)   WRITE THE DEFINED TYPES HEADING TO THE FILE
(*)   WRITE TYPE TO THE FILE
(*)   MAKE AN INCLUSIVE LIST OF DEFINED TYPES WITHIN THE
(*)   SUBSCHEMA
(*)   DELETE ANY DUPLICATES ON THE LIST OF DEFINED TYPES
(*)   SET THE LIST OF DEFINED TYPES TO BE READ FORWARD
(*)   CONTINUE TO READ ITEMS UNTIL THE END OF LIST IS ENCOUNTERED
(*)   GET THE KEY TO THE NEXT DEFINED TYPE IN THE LIST
(*)   GET THIS DEFINED TYPE'S ADB
(*)   GET THE DEFINED TYPE'S CONSTITUENT
(*)   GET THE CONSTITUENT'S ADB
(*)   WRITE TO THE FILE T_<ADB.DEF_TYP_NAME> =
(*)   CALL THE ROUTINE TO WRITE OUT THE PRIMITIVE TYPES OF
(*)   INTEGER

```

```
(*          REAL                                *)
(*          STRING                              *)
(*          LOGICAL                             *)
(*          ARRAY                               *)
(*          DEFINED TYPE                        *)
(*          ENUMERATION                        *)
(*          STRUCTURE                          *)
(*          POINTER                            *)
(*      END                                    *)
(*      $COMMENTS:                            *)
(*      $CHANGE CONTROL:                      *)
(*      REVISED: JANUARY 1988      C. H. MOHME      DBMA      *)
(*      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE      *)
(*      REVISED: 08/13/87          C. H. MOHME      DBMA      *)
(*      CORRECTED CHECKING OF MAS RETURN CODE.                *)
(*      REVISED: 07/30/87          C. H. MOHME      DBMA      *)
(*      A STRUCTURE IS NO LONGER IMPLEMENTED AS A RECORD TYPE. *)
(*      ORIGINATED: 10/23/86      L. J. BEHAN      DBMA      *)
(*-----*)
(*      *)
(*END-----*)
(* END %INCLUDE SCTYPIN *)
```

```
(* %INCLUDE SCTYPUP *)
(**)
PROCEDURE SCTYPUP(VAR IRC          : RET_REC;
                  VAR TRANS_STACK  : TRANSPTR;
                  VAR CNST_KEY     : ENTKEY;
                  VAR CREATE       : BOOLEAN;
                  VAR NEWTYPE      : ENTITY_TYPE;
                  VAR DELETE_LIST  : LISTKEY;
                  VAR NEW_KEYS_LIST: LISTKEY;
                  VAR CONTEXT      : T_CONTEXT);

SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* THIS ROUTINE UPDATES THE DEFINED TYPE, ARRAY, OR FIELD *)
(* ENTITY'S TYPE. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* ==== === ===== *)
(* TRANS_STACK I/O POINTS TO THE TRANSACTION STACK *)
(* CNST_KEY I/O KEY TO THE TYPE TO BE UPDATED *)
(* CREATE I INDICATES IF A NEW ENTITY IS TO BE MADE *)
(* NEWTYPE I ENTITY TYPE TO UPDATE *)
(* DELETE_LIST I/O KEY TO LIST OF ENTITIES TO BE DELETED *)
(* IF THE OPTION SAVE THE CHANGED IS CHOSEN *)
(* NEW_KEYS_LIST I/O KEY TO LIST OF ENTITIES CREATED DURING *)
(* AN UPDATE *)
(* IRC 0 RETURN CODE *)
(* CONTEXT I THE CONTEXT IN WHICH THE DEFINED TYPE *)
(* IS BEING CREATED *)
(* *)
(* $COMMONS: *)
(* NONE *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* THIS ROUTINE DETERMINES IF A NEW ENTITY IS TO BE CREATED *)
(* OR IF THE OLD ENTITY SHOULD BE UPDATED. THEN IT CALLS *)
(* THE ROUTINE ACCORDING TO THE ENTITY TYPE ENTERED. *)
```

April 1990

```
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* REVISED: JANUARY 1988 C. H. MOHME DBMA *)
(* MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE *)
(* *)
(* REVISED: 10/09/87 C. H. MOHME DBMA *)
(* ADDED PARAMETER TO SCDEFGR CALL. *)
(* *)
(* REVISED: 08/13/87 C. H. MOHME DBMA *)
(* ADDED LIST AND SET. NOTE: THE LIST AND SET DATA TYPES ARE *)
(* IMPLEMENTED IN THE SOFTWARE AS AN ARRAY. A FIELD WAS ADDED *)
(* TO THE ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL *)
(* ARRAY, LIST, OR SET. *)
(* *)
(* ORIGINATED: 08/05/86 L. J. BEHAN DBMA *)
(* *)
(*-----*)
(* *)
(*END-----*)
(* END %INCLUDE SCTYPUP *)
```

```

(*) %INCLUDE SCUNIQUE *)
(**)
  PROCEDURE SCUNIQUE(CONST ENTITY_KEY : ENTKEY;
                    VAR   ADB : ENTBLOCK;
                    VAR   DATA : BLKDATA;
                    VAR   RRC : INTEGER);

    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS ROUTINE VERIFIES THE UNIQUENESS OF NAMES WITHIN THE *)
(*)   SCHEMA FOR THE SUBSCHEMA, CLASS, ENTITY, GLOBAL FIELD, *)
(*)   FIELD, ENUMERITEM, AND DEFINED TYPE ENTITIES AS WELL AS *)
(*)   THE USER DEFINED KIND NUMBERS FOR CLASSES AND ENTITIES. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ===  ===== *)
(*)   ENTITY_KEY    I    THE KEY TO THE ENTITY *)
(*)   ADB           0    THE ADB INFORMATION *)
(*)   DATA         I/O  THE USER DEFINED DATA STRUCTURE USED *)
(*)                   TO PASS DATA INTO THIS PROCEDURE AND *)
(*)                   TO GET THE DESIRED OUTPUT FROM THIS *)
(*)                   PROCEDURE *)
(*)   RRC           0    THE ROUTINE RETURN CODE *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   THIS ROUTINE IS EXECUTED BY CALLING MAKXEQ FOR A GIVEN *)
(*)   KIND. THE NAMES AND USER DEFINED KIND NUMBERS ARE *)
(*)   COMPARED TO ONE ANOTHER TO VERIFY THAT THE DATA ENTERED *)
(*)   IS UNIQUE. THE DATA STRUCTURE BLKDATA CONTAINS THE *)
(*)   INFORMATION WHICH INDICATES THE UNIQUENESS. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)

```

```
(* %INCLUDE SCUNQUEST *)
(**)
PROCEDURE SCUNQUEST(VAR IRC : RET_REC;
                    CONST ENT_KIND : INTEGER;
                    VAR UNIQUE_NAME : T_NAME;
                    VAR UNIQUE_NUMBER : INTEGER;
                    VAR FOUND : MATCH);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS ROUTINE CALLS MAKXEQ TO EXECUTE THE PROCEDURE SCUNIQUE
(* WHICH VERIFIES THE UNIQUENESS OF NAMES WITHIN THE SCHEMA
(* FOR THE SUBSCHEMA, CLASS, ENTITY, GLOBAL FIELD, FIELD,
(* ENUMERITEM, AND DEFINED TYPE ENTITIES AS WELL AS THE USER
(* DEFINED KIND NUMBERS FOR CLASSES AND ENTITIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* IRC 0 RETURN CODE
(* ENT_KIND I THE ENTITY KIND NUMBER
(* UNIQUE_NAME I THE NAME TO BE VERIFIED FOR UNIQUENESS
(* UNIQUE_NUMBER I THE NUMBER TO BE VERIFIED FOR UNIQUENESS
(* FOUND 0 A RECORD WHICH INDICATES WHETHER OR
(* NOT A MATCH TO A NAME OR KIND NUMBER
(* HAS BEEN FOUND
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(* THIS ROUTINE CALLS MAKXEQ TO EXECUTE THE PROCEDURE
(* SCUNIQUE WHICH DOES THE ACTUAL COMPARISON FOR UNIQUENESS.
(* THIS CONTINUES UNTIL A MATCH IS FOUND OR ALL THE ENTITIES
(* HAVE BEEN CHECKED.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```



```
(* %INCLUDE SCUNQPND *)
(**)
  PROCEDURE SCUNQPND(VAR IRC : RET_REC;
                    VAR TRANS_STACK : TRANSPTR;
                    VAR UNIQUE_NAME : T_NAME;
                    VAR UNIQUE_NUMBER : INTEGER;
                    VAR TRANSTYPE : TRANS_TYPE;
                    VAR FLDTYP : T_FIELDTYPE;
                    VAR FOUND : MATCH);

    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE VERIFIES THE UNIQUENESS OF NAMES AND USER *)
(*   DEFINED KIND NUMBERS FOR CLASSES AND ENTITIES, BY SEARCHING *)
(*   THE TRANSACTION STACK FOR THE CLASS, ENTITY, FIELD, *)
(*   ENUMERITEM, AND DEFINED TYPE ENTITIES. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   IRC            0    RETURN CODE *)
(*   TRANS_STACK    I    POINTER TO THE TRANSACTION STACK *)
(*   UNIQUE_NAME     I    THE NAME TO BE VERIFIED FOR UNIQUENESS *)
(*   UNIQUE_NUMBER   I    THE NUMBER TO BE VERIFIED FOR UNIQUENESS *)
(*   TRANSTYPE       I    INDICATES THE TYPE OF TRANSACTION *)
(*   FLDTYP          I    INDICATES FOR THE FIELD ENTITY WHAT TYPE *)
(*                       OF FIELD IT IS *)
(*   FOUND           0    A RECORD WHICH INDICATES WHETHER OR NOT *)
(*                       A MATCH BEEN FOUND *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   THIS ROUTINE SEARCHES THROUGH THE TRANSACTION STACK AND *)
(*   COMPARES THE NAME ENTERED TO THOSE TRANSACTIONS WHICH *)
(*   MUST HAVE UNIQUE NAMES WITHIN THE SCHEMA.  FIELD ENTITY *)
(*   NAMES ARE CHECKED FOR UNIQUENESS AMONG THOSE FIELDS WITHIN *)
```

CI PS560240032U
April 1990

```
(*      THE ENTITY BEING CREATED. THIS CONTINUES UNTIL A MATCH IS      *)
(*      FOUND OR THE END OF THE STACK IS ENCOUNTERED.                  *)
(*      $COMMENTS:                                                       *)
(*      $CHANGE CONTROL:                                                 *)
(*                                                                       *)
```

```

(* %INCLUDE SCUPDATE *)
(**)
  PROCEDURE SCUPDATE(VAR IRC          : RET_REC;
                    VAR TRANS_STACK : TRANSPTR);
    SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*      THIS ROUTINE DETERMINES THE NEXT MENU TO DISPLAY FROM THE
(*      UPDATE OPTION CHOSEN.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*      NAME          I/O  DESCRIPTION
(*      ====          ==  =====
(*      IRC            0    RETURN CODE
(*      TRANS_STACK    I/O  POINTS TO THE TRANSACTION STACK
(*
(*  $COMMONS:
(*      DEF
(*      INSIDE        I/O  INDICATES IF THE EXIT OPTION HAS
(*                          BEEN CHOSEN WITHIN ANOTHER CREATE
(*                          PROCEDURE
(*
(*  $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*      INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(*  $PROCESSING DESCRIPTION:
(*      CALLS THE UPDATE MENU INTERFACE ROUTINE (MUPDATE) AND
(*      PROCESSES THE DATA RECEIVED FROM THE MENU EITHER BY
(*      CALLING THE APPROPRIATE ROUTINE OR EXITING THE PROCEDURE.
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*

```

```
(* %INCLUDE SORTKIND *)
(**)
  PROCEDURE SORTKIND(CONST CURRENT : ENTBLOCK;
                     CONST NEXT   : ENTBLOCK;
                     VAR  FLIP    : BOOLEAN;
                     VAR  RRC     : EXT_RET_CODE;
                     VAR  PROC    : ROUTINE);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE IS THE ORDER FUNCTION CALLED BY MALSRT
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   CURRENT       I    THE ADB OF THE CURRENT ENTITY
(*   NEXT          I    THE ADB OF THE NEXT ENTITY
(*   FLIP          O    INDICATES IF THE ENTITIES SHOULD BE
(*                     FLIPPED
(*   RRC           O    THE ROUTINE'S RETURN CODE
(*                     = 0 OK
(*                     <> 0 ERROR
(*   XRC           O    EXTERNAL RETURN CODE FROM MAS
(*                     = 0 OK
(*                     >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY THE MAS ROUTINE MALSRT. THE
(*   TWO ENTITIES ARE COMPARED AND IF THEY ARE OUT OF ALPHA-
(*   BETICAL ORDER THE FLIP FLAG IS SET TO TRUE OTHERWISE THE
(*   FLAG REMAINS FALSE. IF THE FLAG IS TRUE THE ENTITIES ARE
(*   SWAPPED OTHERWISE THEY ARE NOT.
(*
(* $COMMENTS:
(*
```

CI PS560240032U
April 1990

```
(* $CHANGE CONTROL: *)
(*)
(*) ORIGINATED: 3/06/87 M. H. CHOI DBMA (*)
(*)
(*)-----(*)
(*)-----(*)
(*END-----*)
(* END %INCLUDE SORTKIND *)
```

```
(* %INCLUDE SORTNAME *)
(**)
PROCEDURE SORTNAME(CONST CURRENT : ENTBLOCK;
                   CONST NEXT   : ENTBLOCK;
                   VAR  FLIP     : BOOLEAN;
                   VAR  RRC      : EXT_RET_CODE;
                   VAR  PROC     : ROUTINE);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     THIS ROUTINE IS THE ORDER FUNCTION CALLED BY MALSRT
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     CURRENT       I    THE ADB OF THE CURRENT ENTITY
(*     NEXT          I    THE ADB OF THE NEXT ENTITY
(*     FLIP          O    INDICATES IF THE ENTITIES SHOULD BE
(*                      FLIPPED
(*     RRC           O    THE ROUTINE'S RETURN CODE
(*                      = 0 OK
(*                      <> 0 ERROR
(*     XRC           O    EXTERNAL RETURN CODE FROM MAS
(*                      = 0 OK
(*                      >= 10 ERROR
(*
(* $COMMONS:
(*     NONE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*     THIS ROUTINE IS CALLED BY THE MAS ROUTINE MALSRT. THE
(*     TWO ENTITIES ARE COMPARED AND IF THEY ARE OUT OF ALPHA-
(*     BETICAL ORDER THE FLIP FLAG IS SET TO TRUE OTHERWISE THE
(*     FLAG REMAINS FALSE. IF THE FLAG IS TRUE THE ENTITIES ARE
(*     SWAPPED OTHERWISE THEY ARE NOT.
(*
(* $COMMENTS:
(*
```

CI PS560240032U
April 1990

```
(* $CHANGE CONTROL: *)
(*)
(*) ORIGINATED: 3/10/87 M. H. CHOI DBMA *)
(*)
(*)-----*)
(*)-----*)
(*END-----*)
(* END %INCLUDE SORTNAME *)
```

(* %INCLUDE UPARRAY *)

(**)

```
PROCEDURE UPARRAY(VAR MESS      : MESSAGE;
                  VAR LBND      : CHAR8;
                  VAR HBND      : CHAR8;
                  VAR ATYPE      : ENTITY_TYPE;
                  VAR ULBD      : CHAR8;
                  VAR UHBD      : CHAR8;
                  VAR UTYPE      : ENTITY_TYPE;
                  VAR NEXT_OP    : OPERATIONS;
                  VAR CONTEXT    : T_CONTEXT;
                  VAR RR         : RET_REC);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION: (*)
(*)   THIS FUNCTION: (*)
(*)           DISPLAYS THE UPDATE ARRAY MENU (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME      I/O  DESCRIPTION (*)
(*)   ====      ==  ===== (*)
(*)   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL (*)
(*)   LBND      I    THE LOWER BOUND OF THE ARRAY (*)
(*)   HBND      I    THE UPPER BOUND OF THE ARRAY (*)
(*)   ATYPE      I    THE ARRAY TYPE (*)
(*)   ULBD      O    THE UPDATED LOWER BOUND OF THE ARRAY (*)
(*)   UHBD      O    THE UPDATED HIGHER BOUND OF THE ARRAY (*)
(*)   UTYPE      O    THE UPDATED ARRAY TYPE (*)
(*)   NEXT_OP    O    ENUMERATED TYPE INDICATING THE NEXT (*)
(*)                   OPERATION (*)
(*)   CONTEXT    I    THE CONTEXT IN WHICH THE ARRAY IS BEING (*)
(*)                   UPDATED (*)
(*)   RR         O    INDICATES IF AN ERROR HAS OCCURRED AND, (*)
(*)                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN (*)
(*)
(*) $COMMONS: (*)
(*)   NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)   DDNAMES USED WITH STANDARD FILES: (*)
(*)   NONE (*)
(*)
```


CI PS560240032U
April 1990

```
(* $EXECUTION PROCEDURE: *)
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   DISPLAY THE UPDATE ARRAY PANEL (UPARRAY) BY MAKING ISPLNK *)
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED *)
(*   FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE. *)
(* *)
(* $COMMENTS: *)
(*   NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE UPCLASS1 *)
(**)
PROCEDURE UPCLASS1(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR KNUM       : CHAR8;
                   VAR UNAM       : T_NAME;
                   VAR UNUM       : CHAR8;
                   VAR COMMENT    : CHAR150;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE UPDATE CLASS MENU 1
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME          I/O  DESCRIPTION
(*      ===          ==  =====
(*      MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      KNUM          I    THE ENTITY KIND NUMBER
(*      NAME          I    THE ENTITY NAME
(*      UNAM          O    THE UPDATED ENTITY NAME
(*      UNUM          O    THE UPDATED ENTITY NUMBER
(*      NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                        OPERATION
(*      RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                        IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE UPDATE CLASS PANEL NUMBER ONE (UPCLASS1) BY
(*      MAKING ISPLNK CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO
(*      AN ENUMERATED TYPE.  THIS DATA AS WELL AS OTHER INFORMATION
(*)
```

CI PS560240032U
April 1990

[illegible]

```
(* %INCLUDE UPCLASS2 *)
(**)
PROCEDURE UPCLASS2(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR NUM       : CHAR8;
                   VAR GROUP     : T_ARRAYTV;
                   VAR ARRAY_SIZE : INTEGER;
                   VAR MEMBER     : T_NAME;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE UPDATE CLASS MENU 2
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*      ====      ==  =====
(*      MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      MEMBER     O   THE MEMBER SELECTED TO BE UPDATED
(*      GROUP      I   THE ARRAY OF MEMBERS TO SELECT FROM
(*      ARRAY_SIZE I   THE SIZE OF THE ARRAY OF MEMBERS
(*      NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                      OPERATION
(*      RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                      IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE UPDATE CLASS PANEL NUMBER TWO (UPCLASS2) BY
(*      MAKING ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO
(*      AN ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
(*      GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO-
```

CI PS560240032U
April 1990

(*	CEASURE.	*)
(*		*)
(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```
(* %INCLUDE UPDEFTYP *)
(**)
PROCEDURE UPDEFTYP(VAR MESS      : MESSAGE;
                   VAR NAME      : IDCHAR;
                   VAR FTYPE     : ENTITY_TYPE;
                   VAR UNAM      : CHAR16;
                   VAR UTYPE     : ENTITY_TYPE;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR CONTEXT   : T_CONTEXT;
                   VAR RR        : RET_REC);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE UPDATE DEFINED TYPE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      I    THE NAME OF THE DEFINED TYPE
(*   FTYPE     I    THE TYPE OF THE DEFINED TYPE
(*   UNAM      O    THE UPDATED NAME OF THE DEFINED TYPE
(*   FTYPE     I    THE UPDATED TYPE OF THE DEFINED TYPE
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   CONTEXT   I    THE CONTEXT IN WHICH THE DEFINED TYPE
(*                   IS BEING UPDATED
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CREATE DEFINED TYPE PANEL (UPDEFTYP) BY MAKING
```

```
(*      ISPLNK CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN      *)
(*      ENUMERATED TYPE.  THIS DATA AS WELL AS OTHER INFORMATION    *)
(*      GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO-   *)
(*      CEDURE.                                                       *)
(*                                                                    *)
(* $COMMENTS:                                                         *)
(*      NONE                                                         *)
(*                                                                    *)
(* $CHANGE CONTROL:                                                  *)
(*                                                                    *)
(*      REVISED: JANUARY 1988      C. H. MOHME                      DBMA *)
(*      MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE             *)
(*                                                                    *)
(*      REVISED: 08/13/87          C. H. MOHME                      DBMA *)
(*      CHANGED PANEL OPTION NUMBERS; ADDED LIST AND SET.  NOTE: THE *)
(*      LIST AND SET DATA TYPES ARE IMPLEMENTED IN THE SOFTWARE AS AN *)
(*      ARRAY.  A FIELD WAS ADDED TO THE ARRAY ADB TO SPECIFY WHETHER *)
(*      THE ARRAY IS A CONCEPTUAL ARRAY, LIST, OR SET.             *)
(*                                                                    *)
(*      REVISED: 07/02/87          C. H. MOHME                      DBMA *)
(*      CHANGED CURSOR POSITIONING.                                     *)
(*                                                                    *)
(*      ORIGINATED: 07/31/86        C. H. MOHME                      DBMA *)
(*                                                                    *)
(*-----*)
(*                                                                    *)
(*END-----*)
(* END %INCLUDE UPDEFTYP *)
```

```
(* %INCLUDE UPENTY1 *)
(**)
PROCEDURE UPENTY1(VAR MESS      : MESSAGE;
                  VAR NAME      : T_NAME;
                  VAR KNUM      : CHAR8;
                  VAR UNAM      : T_NAME;
                  VAR UNUM      : CHAR8;
                  VAR COMMENT    : CHAR150;
                  VAR NEXT_OP    : OPERATIONS;
                  VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE UPDATE ENTITY MENU 1
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*      ====      ==  =====
(*      MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      NAME      I   THE ENTITY NAME
(*      KNUM      I   THE ENTITY KIND NUMBER
(*      UNAM      O   THE UPDATED ENTITY NAME
(*      UNUM      O   THE UPDATED ENTITY KIND NUMBER
(*      NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*      RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE UPDATE ENTITY PANEL NUMBER ONE (UPENTY1) BY
(*      MAKING ISPLNK CALLS, THE OPTION CHOSEN IS TRANSLATED INTO
(*      AN ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
```


CI PS560240032U
April 1990

```
(*      GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO- *)
(*      CEDURE.                                                         *)
(*                                                                      *)
(*      $COMMENTS:                                                       *)
(*      NONE                                                             *)
(*                                                                      *)
(*      $CHANGE CONTROL:                                                *)
(*                                                                      *)
(*      REVISED: 09/28/87      C. H. MOHME      DBMA      *)
(*      INCORPORATED THE SUPERTYPE DATA TYPE.                        *)
(*                                                                      *)
(*      REVISED: 08/13/87      C. H. MOHME      DBMA      *)
(*      CHANGED PANEL OPTION NUMBERS.                                *)
(*                                                                      *)
(*      REVISED: 07/02/87      C. H. MOHME      DBMA      *)
(*      CHANGED CURSOR POSITIONING.                                    *)
(*                                                                      *)
(*      ORIGINATED: 07/24/86    C. H. MOHME      DBMA      *)
(*      -----*)
(*      -----*)
(*      -----*)
(*END-----*)
(* END %INCLUDE UPENTY1 *)
```

```
(* %INCLUDE UPENTY2 *)
(**)
PROCEDURE UPENTY2(VAR MESS      : MESSAGE;
                  VAR FIELD_TYPE : T_FIELDTYPE;
                  VAR NAME       : T_NAME;
                  VAR KNUM       : CHAR8;
                  VAR MEM_ARRAY  : T_ARRAYID;
                  VAR SIZE       : INTEGER;
                  VAR MEMBER     : T_NAME;
                  VAR NEXT_OP    : OPERATIONS;
                  VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*          THIS PROCEDURE :
(*          DISPLAYS THE UPDATE ENTITY MENU 2
(*
(* $DESCRIPTION OF ARGUMENTS:
(*          NAME          I/O  DESCRIPTION
(*          ===          ==  =====
(*          MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*          FIELD_TYPE    I    THE TYPE OF FIELD
(*          NAME          I    THE ENTITY OR SUPERTYPE NAME
(*          KNUM          I    THE ENTITY KIND NUMBER
(*          MEM_ARRAY     I    THE ARRAY OF MEMBERS TO SELECT FROM
(*          SIZE          I    THE SIZE OF THE ARRAY OF MEMBERS
(*          MEMBER        O    THE MEMBER SELECTED
(*          NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                          OPERATION
(*          RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                          IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*          NONE
(*
(* $ENVIRONMENT:
(*          LANGUAGE: IBM PASCAL
(*          HARDWARE SYSTEM: IBM 360/370/4341/4381
(*          DDNAMES USED WITH STANDARD FILES:
(*          NONE
(*
(* $EXECUTION PROCEDURE:
(*          SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
```

CI PS560240032U
April 1990

```
(*      DISPLAY THE UPDATE ENTITY PANEL NUMBER TWO (UPENTY2) BY      *)
(*      MAKING ISPLNK CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO  *)
(*      AN ENUMERATED TYPE.  THIS DATA AS WELL AS OTHER INFORMATION *)
(*      GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO-  *)
(*      CEDURE.                                                       *)
(*                                                                    *)
(*      $COMMENTS:                                                    *)
(*      NONE                                                           *)
(*                                                                    *)
(*      $CHANGE CONTROL:                                              *)
(*)                                                                    *)
```

```
(* %INCLUDE UPENUM *)
(**)
PROCEDURE UPENUM(VAR MESS      : MESSAGE;
                  VAR MEMBERS  : T_ARRAYID;
                  VAR SIZE     : INTEGER;
                  VAR NAME     : T_NAME;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE UPDATE ENUMERATION MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MEMBERS   I   THE ARRAY OF MEMBERS TO DISPLAY
(*   SIZE      I   THE SIZE OF THE ARRAY OF NUMBERS
(*   NAME      O   THE MEMBER NAME ENTERED
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISLAY THE REVIEW ENUMERATION MENU (UPENUM) BY MAKING
(*   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE.
(*
(* $COMMENTS:
```

CI PS560240032U
April 1990

(* NONE
(*
(* \$CHANGE CONTROL:
(*

*)
)
)
)
)

(* %INCLUDE UPFIELD *)
(**)

```

PROCEDURE UPFIELD(VAR MESS      : MESSAGE;
                  VAR NAME      : T_NAME;
                  VAR POS       : CHAR8;
                  (* VAR PURP    : CHAR8; *)
                  VAR REQD      : CHAR8;
                  (* VAR DEPD    : CHAR12; *)
                  VAR FTYP      : ENTITY_TYPE;
                  VAR FLD_TYP   : T_FIELDTYPE;
                  VAR UNAM      : T_NAME;
                  VAR UPOS      : CHAR8;
                  (* VAR UPUR    : CHAR8; *)
                  VAR UREQ      : CHAR8;
                  (* VAR UDEP    : CHAR8; *)
                  VAR U_TYP     : ENTITY_TYPE;
                  VAR COM       : CHAR50;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

```

SUBPROGRAM;

```

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS PROCEDURE :
(*       DISPLAYS THE UPDATE FIELD PANEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME          I    THE NAME OF THE FIELD
(*   PURP          I    THE PURPOSE OF THE FIELD
(*   REQD          I    THE REQUIREDNESS OF THE FIELD
(*   DEPD          I    THE DEPENDENCE/INDEPENDENCE OF THE FIELD
(*                   THE FIELD
(*   FTYP          I    THE TYPE OF THE FIELD
(*   FLD_TYP       I    THE TYPE OF FIELD
(*   UNAM          O    THE UPDATED NAME OF THE FIELD
(*   UPUR          O    THE UPDATED PURPOSE OF THE FIELD
(*   UREQ          O    THE UPDATED REQUIREDNESS OF THE FIELD
(*   UDEP          O    THE UPDATED DEPENDENCE/INDEPENDENCE OF
(*   UTYP          O    THE UPDATED TYPE OF THE FIELD
(*   NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION

```

```

(*)      RR              0  INDICATES IF AN ERROR HAS OCCURRED AND,  *)
(*)                                IF ONE HAS, WHAT ROUTINE IT OCCURRED IN  *)
(*)                                                                *)
(*)  $COMMONS:                                                    *)
(*)    NONE                                                        *)
(*)                                                                *)
(*)  $ENVIRONMENT:                                                *)
(*)    LANGUAGE: IBM PASCAL                                         *)
(*)    HARDWARE SYSTEM: IBM 360/370/4341/4381                       *)
(*)    DDNAMES USED WITH STANDARD FILES:                           *)
(*)      NONE                                                       *)
(*)                                                                *)
(*)  $EXECUTION PROCEDURE:                                         *)
(*)    SCHEMA EXECUTIVE MENU INTERFACE ROUTINE                     *)
(*)                                                                *)
(*)  $PROCESSING DESCRIPTION:                                       *)
(*)    DISPLAY THE UPDATE FIELD PANEL (UPFIELD) BY MAKING ISPLNK   *)
(*)    CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED   *)
(*)    TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED     *)
(*)    FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.     *)
(*)                                                                *)
(*)  $COMMENTS:                                                    *)
(*)    NONE                                                        *)
(*)                                                                *)
(*)  $CHANGE CONTROL:                                              *)
(*)                                                                *)
(*)    REVISED: JANUARY 1988      C. H. MOHME                      DBMA  *)
(*)    MODIFIED TO INCORPORATE THE STRUCTURE DATA TYPE             *)
(*)                                                                *)
(*)    REVISED: 09/28/87          C. H. MOHME                      DBMA  *)
(*)    INCORPORATED THE SUPERTYPE DATA TYPE.                       *)
(*)                                                                *)
(*)    REVISED: 08/13/87          C. H. MOHME                      DBMA  *)
(*)    CHANGED PANEL OPTION NUMBERS; CHANGED FIELD ADB DATA; ADDED *)
(*)    LIST AND SET. NOTE: THE LIST AND SET DATA TYPES ARE IMPL-  *)
(*)    MENTED IN THE SOFTWARE AS AN ARRAY. A FIELD WAS ADDED TO THE *)
(*)    ARRAY ADB TO SPECIFY WHETHER THE ARRAY IS A CONCEPTUAL ARRAY, *)
(*)    LIST, OR SET.                                                *)
(*)                                                                *)
(*)    REVISED: 07/02/87          C. H. MOHME                      DBMA  *)
(*)    CHANGED CURSOR POSITIONING.                                    *)
(*)                                                                *)
(*)    ORIGINATED: 07/15/86        C. H. MOHME                      DBMA  *)
(*)                                                                *)
(*)-----*)
(*)-----*)
(*)END-----*)
(*) END %INCLUDE UPFIELD *)

```

```
(* %INCLUDE UPINT *)
(**)
  PROCEDURE UPINT(VAR MESS      : MESSAGE;
                  VAR PREC      : CHAR8;
                  VAR UPREC     : CHAR8;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE UPDATE INTEGER MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS       I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   PREC       I   THE PRECISION OF THE INTEGER
(*   UPREC      O   THE UPDATED PRECISION OF THE INTEGER
(*   NEXT_OP    O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR         O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE UPDATE INTEGER PANEL (UPINT) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED
(*   FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
```


CI PS560240032U
April 1990

```
(* $CHANGE CONTROL: *)
(*) *)
(*) REVISIED: 10/09/87 C. H. MOHME DBMA *)
(*) ADDED DEFAULT PRECISION. *)
(*) *)
(*) REVISIED: 08/13/87 C. H. MOHME DBMA *)
(*) CHANGED PANEL OPTION NUMBERS. *)
(*) *)
(*) REVISIED: 07/02/87 C. H. MOHME DBMA *)
(*) CHANGED CURSOR POSITIONING. *)
(*) *)
(*) ORIGINATED: 08/04/86 C. H. MOHME DBMA *)
(*) *)
(*)-----*)
(*)-----*)
(*)-----*)
(*END-----*)
(* END %INCLUDE UPINT *)
```

```
(* %INCLUDE UPLIST *)
(**)
  PROCEDURE UPLIST(VAR MESS      : MESSAGE;
                   VAR MIN      : CHAR8;
                   VAR MAX      : CHAR8;
                   VAR ATYPE    : ENTITY_TYPE;
                   VAR UMIN     : CHAR8;
                   VAR UMAX     : CHAR8;
                   VAR UTYPE    : ENTITY_TYPE;
                   VAR NEXT_OP  : OPERATIONS;
                   VAR CONTEXT  : T_CONTEXT;
                   VAR RR       : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE UPDATE LIST MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   MIN      I   THE MINIMUM NUMBER OF OCCURRENCES IN THE
(*               LIST
(*   MAX      I   THE MAXIMUM NUMBER OF OCCURRENCES IN THE
(*               LIST
(*   ATYPE    I   THE LIST TYPE
(*   UMIN     O   THE UPDATED MINIMUM NUMBER OF OCCUR-
(*               RENCES IN THE LIST
(*   UMAX     O   THE UPDATED MAXIMUM NUMBER OF OCCUR-
(*               RENCES IN THE LIST
(*   UTYPE    O   THE UPDATED LIST TYPE
(*   NEXT_OP  O   ENUMERATED TYPE INDICATING THE NEXT
(*               OPERATION
(*   CONTEXT  I   THE CONTEXT IN WHICH THE LIST IS BEING
(*               UPDATED
(*   RR      O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
```

```
(*      DDNAMES USED WITH STANDARD FILES:      *)
(*      NONE                                     *)
(*      *)                                       *)
(* $EXECUTION PROCEDURE:                       *)
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(*      *)                                       *)
(* $PROCESSING DESCRIPTION:                   *)
(*      DISPLAY THE UPDATE LIST PANEL (UPLIST) BY MAKING ISPLNK *)
(*      CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*      TYPE.  THIS DATA AS WELL AS OTHER INFORMATION GATHERED *)
(*      FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE. *)
(*      *)                                       *)
(* $COMMENTS:                                *)
(*      NONE                                     *)
(*      *)                                       *)
(* $CHANGE CONTROL:                          *)
(*      *)                                       *)
```

(* %INCLUDE UPPNTR *)

(**)

```

PROCEDURE UPPNTR(VAR MESS      : MESSAGE;
                  VAR GROUP    : T_ARRAYTV;
                  VAR ARRAY_SIZE : INTEGER;
                  VAR MEMBER    : T_NAME;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

```

SUBPROGRAM;

(**)

```

(*)-----*)
(*)
(*) $FUNCTION: *)
(*)      THIS PROCEDURE : *)
(*)      DISPLAYS THE UPDATE POINTER MENU *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)      NAME      I/O  DESCRIPTION *)
(*)      ====      ==  ===== *)
(*)      MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(*)      GROUP     I    THE ARRAY OF MEMBERS TO SELECT FROM *)
(*)      ARRAY_SIZE I    THE SIZE OF THE ARRAY OF MEMBERS *)
(*)      MEMBER     O    THE MEMBER SELECTED TO BE UPDATED *)
(*)      NEXT_OP    O    ENUMERATED TYPE INDICATING THE NEXT *)
(*)                   OPERATION *)
(*)      RR         O    INDICATES IF AN ERROR HAS OCCURRED AND, *)
(*)                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(*) *)
(*) $COMMONS: *)
(*)      NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)      LANGUAGE: IBM PASCAL *)
(*)      HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)      DDNAMES USED WITH STANDARD FILES: *)
(*)      NONE *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)      DISPLAY THE UPDATE POINTER PANEL (UPPNTR) BY MAKING ISPLNK *)
(*)      CALLS.  THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*)      TYPE. *)
(*) *)
(*) $COMMENTS: *)
(*)      NONE *)
(*) *)
(*) $CHANGE CONTROL: *)

```

```
(* %INCLUDE UPREAL *)
(**)
PROCEDURE UPREAL(VAR MESS      : MESSAGE;
                  VAR PREC     : CHAR8;
                  VAR UPREC    : CHAR8;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*           DISPLAYS THE UPDATE REAL MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   PREC      I    THE REAL SIZE IN DECIMAL DIGIT FORM
(*   UPREC     O    THE UPDATED REAL SIZE IN DECIMAL DIGIT
(*                   FORM
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE UPDATE REAL PANEL (UPREAL) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED
(*   FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
(*
```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(* NONE *)
(* $CHANGE CONTROL: *)
(* REVISIED: 10/09/87 C. H. MOHME DBMA *)
(* ADDED DEFAULT PRECISION. *)
(* REVISIED: 08/13/87 C. H. MOHME DBMA *)
(* CHANGED PANEL OPTION NUMBERS. *)
(* REVISIED: 07/02/87 C. H. MOHME DBMA *)
(* CHANGED CURSOR POSITIONING. *)
(* ORIGINATED: 08/05/86 C. H. MOHME DBMA *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE UPREAL *)
```

(* %INCLUDE UPSET *)

(**)

```
PROCEDURE UPSET(VAR MESS      : MESSAGE;
                VAR MIN       : CHAR8;
                VAR MAX       : CHAR8;
                VAR ATYPE     : ENTITY_TYPE;
                VAR UMIN      : CHAR8;
                VAR UMAX      : CHAR8;
                VAR UTYPE     : ENTITY_TYPE;
                VAR NEXT_OP   : OPERATIONS;
                VAR CONTEXT   : T_CONTEXT;
                VAR RR        : RET_REC);
```

SUBPROGRAM;

(**)

```
(*-----*)
(*)
(*) $FUNCTION:
(*)   THIS FUNCTION:
(*)       DISPLAYS THE UPDATE SET MENU
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O  DESCRIPTION
(*)   ====      ==  =====
(*)   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*)   MIN      I    THE MINIMUM NUMBER OF OCCURRENCES IN THE
(*)           SET
(*)   MAX      I    THE MAXIMUM NUMBER OF OCCURRENCES IN THE
(*)           SET
(*)   ATYPE     I    THE SET TYPE
(*)   UMIN      O    THE UPDATED MINIMUM NUMBER OF OCCUR-
(*)           RENCES IN THE SET
(*)   UMAX      O    THE UPDATED MAXIMUM NUMBER OF OCCUR-
(*)           RENCES IN THE SET
(*)   UTYPE     O    THE UPDATED SET TYPE
(*)   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*)           OPERATION
(*)   CONTEXT   I    THE CONTEXT IN WHICH THE SET IS BEING
(*)           UPDATED
(*)   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*)           IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
```

```
(*      DDNAMES USED WITH STANDARD FILES:      *)
(*      NONE                                     *)
(*      $EXECUTION PROCEDURE:                   *)
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(*      $PROCESSING DESCRIPTION:                 *)
(*      DISPLAY THE UPDATE SET PANEL (UPSET) BY MAKING ISPLNK *)
(*      CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED *)
(*      TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED *)
(*      FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE. *)
(*      $COMMENTS:                               *)
(*      NONE                                     *)
(*      $CHANGE CONTROL:                         *)
(*)
```



```
(* %INCLUDE UPSTRING *)
(**)
  PROCEDURE UPSTRING(VAR MESS      : MESSAGE;
                     VAR LEN       : CHAR8;
                     VAR ULEN      : CHAR8;
                     VAR NEXT_OP   : OPERATIONS;
                     VAR RR        : RET_REC);

    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE UPDATE STRING MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   MESS           I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   LEN            I   THE LENGTH OF THE STRING IN BYTES
(*   ULEN           O   THE UPDATED LENGTH OF THE STRING IN
(*                       BYTES
(*   NEXT_OP        O   ENUMERATED TYPE INDICATING THE NEXT
(*                       OPERATION
(*   RR             O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                       IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE UPDATE STRING PANEL (UPSTRING) BY MAKING ISPLNK
(*   CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN ENUMERATED
(*   TYPE. THIS DATA AS WELL AS OTHER INFORMATION GATHERED
(*   FROM THE PANEL IS PASSED BACK TO THE CALLING PROCEDURE.
(*
(* $COMMENTS:
(*   NONE
(*
```

```
(* $CHANGE CONTROL: *)
(*)
(*) REVISIED: 10/09/87 C. H. MOHME DBMA *)
(*) ADDED DEFAULT STRING LENGTH. *)
(*)
(*) REVISIED: 08/13/87 C. H. MOHME DBMA *)
(*) CHANGED PANEL OPTION NUMBERS. *)
(*)
(*) REVISIED: 07/02/87 C. H. MOHME DBMA *)
(*) CHANGED CURSOR POSITIONING. *)
(*)
(*) ORIGINATED: 08/05/86 C. H. MOHME DBMA *)
(*)
(*)-----*)
(*)
(*)END-----*)
(* END %INCLUDE UPSTRING *)
```

```

(** %INCLUDE UPSTRUC *)
(**)
  PROCEDURE UPSTRUC(VAR MESS      : MESSAGE;
                    VAR CLAS      : T_ARRAYID;
                    VAR ARRAY_SIZE : INTEGER;
                    VAR MEMBER     : T_NAME;
                    VAR NEXT_OP    : OPERATIONS;
                    VAR RR         : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(*
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE UPDATE STRUCTURE MENU
(*      RECEIVES THE NAME OF A STRUCTURE TO BE UPDATED
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*      ====      ==  =====
(*      MESS      I   THE ERROR MESSAGE RECEIVED FROM MAINLINE
(*      CLAS      I   THE ARRAY OF FIELDS
(*      ARRAY_SIZE I   THE SIZE OF THE ARRAY OF MEMBERS
(*      MEMBER     0   THE MEMBER SELECTED
(*      NEXT_OP    0   TELLS THE MAINLINE WHAT PANEL TO CALL
(*                      NEXT
(*      RR         0   TELLS THE MAINLINE IF THERE IS AN ERROR
(*                      AND IN WHAT ROUTINE IT OCCURS
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE UPDATE STRUCTURE PANEL (UPSTRUC) BY MAKING
(*      ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*      ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
(*      GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING
(*      PROCEDURE.
(*

```

CI PS560240032U
April 1990

(* \$COMMENTS:
(*
(* \$CHANGE CONTROL:
(*

*)
)
)
)
)

```
(* %INCLUDE UPSUB1 *)
(**)
PROCEDURE UPSUB1(VAR MESS      : MESSAGE;
                  VAR NAME     : T_NAME;
                  VAR UNAM     : T_NAME;
                  VAR COMMENT  : CHAR150;
                  VAR NEXT_OP  : OPERATIONS;
                  VAR RR       : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION: (*)
(*)   THIS FUNCTION: (*)
(*)           DISPLAYS THE UPDATE SUBSCHEMA PANEL 1 (*)
(*)           INPUT OF THE NAME OF A SUBSCHEMA (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME      I/O  DESCRIPTION (*)
(*)   ====      ==  ===== (*)
(*)   MESS      I   THE ERROR MESSAGE RECEIVED FROM MAINLINE (*)
(*)   NAME      I   THE SUBSCHEMA NAME (*)
(*)   UNAM      O   THE UPDATED SUBSCHEMA NAME (*)
(*)   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT (*)
(*)               OPERATION (*)
(*)   XRC      O   INDICATES IF AN ERROR HAS OCCURRED AND, (*)
(*)               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN (*)
(*)
(*) $COMMONS: (*)
(*)   NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)   DDNAMES USED WITH STANDARD FILES: (*)
(*)   NONE (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)   DISPLAY THE UPDATE SUBSCHEMA PANEL (UPSUB1) BY MAKING (*)
(*)   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN (*)
(*)   ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION (*)
(*)   GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING (*)
(*)   PROCEDURE. (*)
```

CI PS560240032U
April 1990

(*		*)
(*	\$COMMENTS:	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)

```
(* %INCLUDE UPSUB2 *)
(**)
PROCEDURE UPSUB2(VAR MESS      : MESSAGE;
                  VAR NAME     : T_NAME;
                  VAR MEMBERS   : T_ARRAYTV;
                  VAR ARRAY_SIZE : INTEGER;
                  VAR MEMBER    : T_NAME;
                  VAR NEXT_OP   : OPERATIONS;
                  VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS FUNCTION:
(*       DISPLAYS THE REVIEW SUBSCHEMA PANEL 2
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      I   THE NAME OF THE SUBSCHEMA TO BE UPDATED
(*   MEMBERS   I   THE ARRAY OF MEMBERS TO SELECT FROM
(*   ARRAY_SIZE I   THE SIZE OF THE ARRAY OF MEMBERS
(*   MEMBER    O   THE MEMBER SELECTED
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE UPDATE SUBSCHEMA PANEL NUMBER TWO BY MAKING
(*   ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO AN
(*   ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
(*   GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING
```

CI PS560240032U
April 1990

(*	PROCEDURE.	*)
(*		*)
(*	\$COMMENTS:	*)
(*	NONE	*)
(*		*)
(*	\$CHANGE CONTROL:	*)
(*		*)


```
(* %INCLUDE UPSUPER *)
(**)
PROCEDURE UPSUPER(VAR MESS          : MESSAGE;
                  VAR NAME          : T_NAME;
                  VAR UNAM          : T_NAME;
                  VAR REMOVE_SUPERTYPE : BOOLEAN;
                  VAR NEXT_OP       : OPERATIONS;
                  VAR RR            : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*      THIS PROCEDURE :
(*      DISPLAYS THE UPDATE SUPERTYPE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*      NAME          I/O  DESCRIPTION
(*      ====          ==  =====
(*      MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*      NAME          I    THE ENTITY NAME
(*      KNUM          I    THE ENTITY KIND NUMBER
(*      UNAM          O    THE UPDATED ENTITY NAME
(*      UNUM          O    THE UPDATED ENTITY KIND NUMBER
(*      NEXT_OP       O    ENUMERATED TYPE INDICATING THE NEXT
(*                        OPERATION
(*      RR            O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                        IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*      NONE
(*
(* $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*      DDNAMES USED WITH STANDARD FILES:
(*      NONE
(*
(* $EXECUTION PROCEDURE:
(*      SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*      DISPLAY THE UPDATE ENTITY PANEL NUMBER ONE (UPSUPER) BY
(*      MAKING ISPLNK CALLS. THE OPTION CHOSEN IS TRANSLATED INTO
(*      AN ENUMERATED TYPE. THIS DATA AS WELL AS OTHER INFORMATION
(*      GATHERED FROM THE PANEL IS PASSED BACK TO THE CALLING PRO-
(*      CEDURE.
```

CI PS560240032U
April 1990

(* \$COMMENTS:

(* NONE

(*

(* \$CHANGE CONTROL:

(*

*)

*)

*)

*)

*)

```
(* %INCLUDE XATTDATA *)
(**)
  PROCEDURE XATTDATA(VAR IRC      : RET_REC;
                    VAR XREFFILE : TEXT);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE GENERATES A LIST OF ALL "ENTITIES" CONTAINING
(*   A PARTICULAR "ENTITY."
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   IRC        0   RETURN CODE
(*   XREFFILE    0   CROSS REFERENCE REPORT FILE
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   DISPLAY THE CROSS REFERENCE MENU.
(*   DETERMINE WHICH OPTION WAS CHOSEN AND CALL THE APPROPRIATE
(*   ROUTINE.
(*   INTEGER DATA TYPE.  DISPLAY MENU TO OBTAIN DESIRED
(*   PRECISION.
(*   REAL DATA TYPE.  DISPLAY MENU TO OBTAIN DESIRED PRECISION.
(*   STRING DATA TYPE.  DISPLAY MENU TO OBTAIN DESIRED LENGTH.
(*   OTHER DATA TYPES.
(*   ARRAY, LIST, OR SET DATA TYPE.
(*   THE LIST AND SET DATA TYPES ARE IMPLEMENTED AS ARRAYS.
(*   THEREFORE, SPECIAL CHECKING MUST BE PERFORMED.
(*   IF A PRECISION WAS SPECIFIED FOR THE INTEGER, REAL, OR STRING
(*   DATA TYPE, THEN SELECT FROM THE DATA TYPE LIST THOSE
(*   ENTITIES WITH THE SPECIFIED PRECISION.
(*   MAKE A LIST OF THE USERS OF THE DATA TYPE AND CALL THE
(*   ROUTINE TO DISPLAY THE ATTRIBUTES AND USER ENTITIES.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE XATTNAME *)
(**)
  PROCEDURE XATTNAME(VAR IRC      : RET_REC;
                    VAR XREFFILE : TEXT);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE GENERATES A LIST OF ALL ENTITIES HAVING AN *)
(*   ATTRIBUTE WITH A SPECIFIED NAME. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  ===== *)
(*   IRC        0   RETURN CODE *)
(*   XREFFILE    0   THE CROSS REFERENCE REPORT FILE *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   DETERMINE THE ATTRIBUTE NAME. *)
(*   DETERMINE IF A VALID IDENTIFIER WAS ENTERED. *)
(*   MAKE A LIST OF ATTRIBUTES OF THE SPECIFIED NAME. *)
(*   MAKE A LIST OF THE ENTITY USERS. *)
(*   PREPARE TO DISPLAY THE RESULTS. *)
(*   DISPLAY THE RESULTS. *)
(*   RETURN TO MAIN MENU. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(* %INCLUDE XATTRES *)
(**)
  PROCEDURE XATTRES(VAR IRC           : RET_REC;
                    VAR ATTRIBUTE_LIST : LISTKEY;
                    VAR DATA_TYPE_KIND : INTEGER;
                    VAR PRECISION      : INTEGER;
                    VAR COM1            : CHAR50;
                    VAR COM2            : CHAR50;
                    VAR XREFFILE        : TEXT;
                    VAR OPTION          : OPERATIONS);

    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   DISPLAYS CROSS REFERENCE REPORT RESULTS *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  ===== *)
(*   IRC        0   RETURN CODE *)
(*   ATTRIBUTE_LIST  I   THE LIST OF ATTRIBUTES *)
(*   DATA_TYPE_KIND  I   THE DATA TYPE KIND TO BE FOUND IN THE *)
(*                       ATTRIBUTE LIST *)
(*   PRECISION      I   THE INTEGER OR REAL PRECISION, OR THE *)
(*                       STRING LENGTH *)
(*   COM1           I   THE MENU COMMENTS *)
(*   COM2           I   THE MENU COMMENTS *)
(*   XREFFILE       I/O  THE CROSS REFERENCE REPORT FILE *)
(*   OPTION         0   THE OPTION SELECTED *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   READ EACH ATTRIBUTE OFF OF THE LIST OF ATTRIBUTES *)
(*   FROM THE ATTRIBUTE ADB OBTAIN THE ATTRIBUTE NAME AND PUT IT *)
(*   INTO AN ARRAY. *)
(*   FIND THE LIST OF USERS OF THE ATTRIBUTE.  THIS LIST WILL *)
(*   ALWAYS CONSIST OF ONE AND ONLY ONE MEMBER. *)

```

```
(* FROM THE USER ADB, OBTAIN THE USER NAME AND PUT IT INTO AN *)
(* ARRAY. IF THE USER IS A GLOBAL ATTRIBUTE, PUT THE KEYWORD *)
(* "GLOBAL" INTO THE ARRAY. *)
(* DEFINE THE COMMENTS FOR THE CROSS REFERENCE REPORT *)
(* DISPLAY THE CROSS REFERENCE REPORT RESULT MENU *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE XEXPREC *)
(**)
PROCEDURE XEXPREC(VAR   IRC       : RET_REC;
                  CONST ENT_KIND : INTEGER;
                  VAR   XREFFILE  : TEXT;
                  VAR   MSG       : MESSAGE);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   THIS ROUTINE GENERATES A LIST OF ALL EXISTING PRECISIONS *)
(*   FOR THE INTEGER, REAL, OR STRING DATA TYPE. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ====          ==  ===== *)
(*   IRC           0    RETURN CODE *)
(*   ENT_KIND      0    THE ENTITY KIND *)
(*   XREFFILE      0    THE CROSS REFERENCE REPORT FILE *)
(*   MSG           0    PANEL MESSAGE *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   ESTABLISH THE MAXIMUM PRECISION AND THE DISPLAY MESSAGE. *)
(*   THE EXISTS ARRAY IS USED TO HOLD BOOLEAN VALUES. EXISTS(.X.) *)
(*   IS TRUE IF THE X PRECISION EXISTS FOR THE INTEGER, REAL, OR *)
(*   STRING DATA TYPE IN THE SCHEMA MODEL. *)
(*   MAKE A LIST OF THE INTEGER, REAL, OR STRING DATA TYPES AND *)
(*   DETERMINE THE EXISTING PRECISIONS. *)
(*   PREPARE TO DISPLAY THE RESULTS. *)
(*   DISPLAY THE RESULTS. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(*) %INCLUDE XFNDARY *)
(**)
  PROCEDURE XFNDARY(CONST ENTITY_KEY : ENTKEY;
                    VAR   ADB         : ENTBLOCK;
                    VAR   DATA       : BLKDATA;
                    VAR   RRC         : EXT_RET_CODE);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS ROUTINE FINDS ALL ARRAY DATA TYPES OF SET, LIST, OR *)
(*)   ARRAY AND PUTS THEM ON A LIST. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O  DESCRIPTION *)
(*)   ====      ==  ===== *)
(*)   ENTITY_KEY I    KEY TO THE ENTITY *)
(*)   ADB        O    DATA STORED IN THE ADB *)
(*)   DATA      I/O  THE USER DEFINED DATA STRUCTURE USED *)
(*)               TO PASS DATA INTO THIS PROCEDURE AND *)
(*)               TO GET THE DESIRED OUTPUT FROM THE *)
(*)               PROCEDURE *)
(*)   RRC        O    EXTERNAL RETURN CODE *)
(*)               = 0  OK *)
(*)               >= 10 ERROR *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   THIS ROUTINE IS CALLED BY A MAS EXECUTE ROUTINE. ALL *)
(*)   ARRAY DATA TYPES OF SET, LIST, OR ARRAY ARE PUT ONTO A *)
(*)   LIST. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```



```
(* %INCLUDE XFNDKEY *)
(**)
PROCEDURE XFNDKEY(VAR IRC          : RET_REC;
                  VAR CHAR_STRING  : T_NAME;
                  VAR BAD_NAME     : BOOLEAN;
                  VAR BAD_KIND_NUMBER : BOOLEAN;
                  VAR NUMBER       : INTEGER;
                  CONST ENT_KIND   : INTEGER);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE DETERMINES THE KEY FOR A GIVEN ENTITY NAME
(*   OR NUMBER
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   IRC        0   RETURN CODE
(*   CHAR_STRING I   THE GIVEN CHARACTER STRING
(*   BAD_NAME    0   INDICATES IF THE GIVEN CHARACTER STRING
(*                   IS A NAME
(*   BAD_KIND_NUMBER 0   INDICATES IF THE GIVEN CHARACTER STRING
(*                   IS A NUMBER
(*   NUMBER      0   THE NUMBER CONTAINED IN THE CHARACTER
(*                   STRING
(*   DATA_REC   0   THE RECORD USED BY MAKXEQ
(*   ENT_KIND    I   THE ENTITY KIND
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   INITIALIZE THE RECORD USED BY MAS IN MAKXEQ.  THIS RECORD
(*   CONTAINS AN ENTITY NAME OR KIND NUMBER.  THE MAKXEQ ROUTINE
(*   WILL RETURN THE ENTITY KEY, IF THE ENTITY EXISTS IN THE
(*   MODEL.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(* %INCLUDE XFNDNAME *)
(**)
PROCEDURE XFNDNAME(CONST ENTITY_KEY : ENTKEY;
                   VAR   ADB         : ENTBLOCK;
                   VAR   DATA       : BLKDATA;
                   VAR   RRC         : EXT_RET_CODE);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE FINDS ALL ATTRIBUTES WITH THE GIVEN NAME AND
(*   PUTS THEM ON A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   ENTITY_KEY    I    KEY TO THE ENTITY
(*   ADB           0    DATA STORED IN THE ADB
(*   DATA         I/O  THE USER DEFINED DATA STRUCTURE USED
(*                       TO PASS DATA INTO THIS PROCEDURE AND
(*                       TO GET THE DESIRED OUTPUT FROM THE
(*                       PROCEDURE
(*   RRC           0    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY A MAS EXECUTE ROUTINE. ALL
(*   ATTRIBUTES WITH THE GIVEN NAME ARE RETURNED IN A LIST.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```
(* %INCLUDE XFNDPREC *)
(**)
  PROCEDURE XFNDPREC(CONST ENTITY_KEY : ENTKEY;
                    VAR   ADB         : ENTBLOCK;
                    VAR   DATA       : BLKDATA;
                    VAR   RRC         : EXT_RET_CODE);
    SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE FINDS ALL INTEGERS, REALS, OR STRINGS WITH
(*   THE SPECIFIED PRECISION AND PUTS THEM ON A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   ENTITY_KEY    I    KEY TO THE ENTITY
(*   ADB           0    DATA STORED IN THE ADB
(*   DATA         I/O  THE USER DEFINED DATA STRUCTURE USED
(*                       TO PASS DATA INTO THIS PROCEDURE AND
(*                       TO GET THE DESIRED OUTPUT FROM THE
(*                       PROCEDURE
(*   RRC           0    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       >= 10 ERROR
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   THIS ROUTINE IS CALLED BY A MAS EXECUTE ROUTINE. ALL
(*   INTEGERS, REALS, OR STRINGS WITH THE SPECIFIED PRECISION
(*   ARE FOUND AND PUT ONTO A LIST.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

```

(*) %INCLUDE XLISTENT *)
(**)
  PROCEDURE XLISTENT(VAR IRC                : RET_REC;
                    CONST SPECIFIED_KEY_KIND : INTEGER;
                    CONST RESULT_LIST_KIND   : INTEGER;
                    VAR  XREFFILE            : TEXT);

    SUBPROGRAM;

(**)
(*)-----*)
(*)*)
(*) $FUNCTION:*)
(*) THIS ROUTINE GENERATES A LIST OF ALL "ENTITIES" CONTAINING*)
(*) A PARTICULAR ENTITY. *)
(*)*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) IRC O RETURN CODE *)
(*) SPECIFIED_KEY_KIND I THE ENTITY KIND OF THE PARTICULAR ENTITY *)
(*) TO BE FOUND. *)
(*) RESULT_LIST_KIND I THE ENTITY KIND OF THE RESULTING LIST *)
(*) XREFFILE O THE CROSS REFERENCE REPORT FILE *)
(*)*)
(*) $COMMONS: *)
(*) NONE *)
(*)*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)*)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)*)
(*) $PROCESSING DESCRIPTION: *)
(*) DISPLAY THE CROSS REFERENCE MENU *)
(*) DEFINE THE MENU COMMENTS TO DISPLAY FOR THE USER'S INFORMATION *)
(*) DISPLAY THE SPECIFICATION MENU, IF APPROPRIATE *)
(*) DETERMINE WHICH OPTION WAS CHOSEN AND CALL THE APPROPRIATE *)
(*) ROUTINE *)
(*) DETERMINE IF A VALID NAME OR NUMBER WAS ENTERED *)
(*) DETERMINE THE KEY OF THE NAME OR NUMBER SPECIFIED *)
(*) CREATE THE DESIRED LIST AND DEFINE THE CROSS REFERENCE REPORT *)
(*) COMMENTS *)
(*) FIND ALL ATTRIBUTES OF THE SPECIFIED DEFINED TYPE *)
(*) FIND ALL CLASSES CONTAINING THE SPECIFIED ENTITY *)
(*) FIND ALL SUBSCHEMAS CONTAINING THE SPECIFIED ENTITY *)
(*) FIND ALL SUBSCHEMAS CONTAINING THE SPECIFIED CLASS *)

```

CI PS560240032U
April 1990

(* DISPLAY THE RESULTS
(* RETURN TO MAIN MENU
(*
(* \$COMMENTS:
(*
(* \$CHANGE CONTROL:
(*

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

```

(*) %INCLUDE XMAIN *)
(**)
  PROCEDURE XMAIN(VAR IRC      : RET_REC;
                  VAR XREFFILE : TEXT);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   THIS ROUTINE DETERMINES THE CROSS REFERENCE OPTION DESIRED *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O  DESCRIPTION *)
(*)   ====      ==  ===== *)
(*)   IRC        O   RETURN CODE *)
(*)   XREFFILE    O   CROSS REFERENCE REPORT FILE *)
(*)
(*) $COMMONS: *)
(*)   NONE *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   DISPLAY THE CROSS REFERENCE MAIN MENU. *)
(*)   DETERMINE WHICH OPTION WAS CHOSEN AND CALL THE APPROPRIATE *)
(*)   ROUTINE. *)
(*)   LIST ALL ATTRIBUTES OF A PARTICULAR DATA TYPE (THIS LIST OF *)
(*)   ATTRIBUTES ALSO INCLUDES THE ATTRIBUTES' USER ENTITY). *)
(*)   LIST ALL ENTITIES HAVING AN ATTRIBUTE OF A PARTICULAR *)
(*)   DEFINED TYPE. *)
(*)   LIST ALL ENTITIES HAVING AN ATTRIBUTE WITH A PARTICULAR *)
(*)   NAME. *)
(*)   LIST ALL CLASSES CONTAINING A PARTICULAR ENTITY. *)
(*)   LIST ALL SUBSCHEMAS CONTAINING A PARTICULAR ENTITY. *)
(*)   LIST ALL SUBSCHEMAS CONTAINING A PARTICULAR CLASS. *)
(*)   LIST ALL EXISTING PRECISIONS FOR THE INTEGER DATA TYPE. *)
(*)   LIST ALL EXISTING PRECISIONS FOR THE REAL DATA TYPE. *)
(*)   LIST ALL EXISTING LENGTHS FOR THE STRING DATA TYPE. *)
(*)   RETURN TO MAIN MENU *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)

```

```
(* %INCLUDE XMATTRES *)
(**)
PROCEDURE XMATTRES(VAR MESS      : MESSAGE;
                   VAR CLAS      : T_ARRAYRV;
                   VAR ARRAY_SIZE : INTEGER;
                   VAR COM1       : CHAR50;
                   VAR COM2       : CHAR50;
                   VAR XREFFILE   : TEXT;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DISPLAYS A CROSS REFERENCE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   CLAS      I   THE ARRAY OF MEMBERS
(*   ARRAY_SIZE I   THE SIZE OF THE ARRAY OF MEMBERS
(*   COM1      I   THE MENU COMMENTS
(*   COM2      I   THE MENU COMMENTS
(*   XREFFILE  I/O  THE CROSS REFERENCE REPORT FILE
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   INITIALIZE THE VARIABLES
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
```

```
(*      CREATE THE TABLE                                *)
(*      PREPARE TO DISPLAY THE CROSS REFERENCE DISPLAY MENU *)
(*      LOAD THE TABLE AND WRITE THE CROSS REFERENCE REPORT RESULTS TO *)
(*      FILE                                              *)
(*      DISPLAY THE CROSS REFERENCE DISPLAY MENU          *)
(*      DETERMINE THE ACTION SELECTED FROM THE MENU      *)
(*      REMOVE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM *)
(*      VARIABLES                                          *)
(*      REMOVE THE TABLE                                *)
(*                                                      *)
(*      $COMMENTS:                                       *)
(*      NONE                                             *)
(*                                                      *)
(*      $CHANGE CONTROL:                                *)
(*                                                      *)
```



```

(** %INCLUDE XMMAIN *)
(**)
  PROCEDURE XMMAIN(VAR MESS      : MESSAGE;
                   VAR NEXT_OP  : OPERATIONS;
                   VAR RR       : RET_REC);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   DISPLAYS A CROSS REFERENCE MENU *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  ===== *)
(*   MESS      I   THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(*   NEXT_OP   O   ENUMERATED TYPE INDICATING THE NEXT *)
(*               OPERATION *)
(*   RR        O   INDICATES IF AN ERROR HAS OCCURRED AND, *)
(*               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*   DDNAMES USED WITH STANDARD FILES: *)
(*   NONE *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   INITIALIZE THE VARIABLES *)
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM *)
(*   VARIABLES *)
(*   DISPLAY THE CROSS REFERENCE MENU *)
(*   DETERMINE THE ACTION SELECTED FROM THE MENU *)
(*   REMOVE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM *)
(*   VARIABLES *)
(* *)
(* $COMMENTS: *)
(*   NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)

```

CI PS560240032U
April 1990

```
(*  REVISED: 02/26/88          C. H. MOHME          DBMA      *)
(*  MODIFIED TO INCORPORATE ALLOCATE, FREE, BROWSE, AND PRINT      *)
(*  CAPABILITIES.                                                  *)
(*                                                                *)
(*  ORIGINATED: 11/16/87      C. H. MOHME          DBMA      *)
(*                                                                *)
(*-----*)
(*                                                                *)
(*END-----*)
(* END %INCLUDE XMMAIN *)
```

```
(* %INCLUDE XMNAMSPE *)
(**)
PROCEDURE XMNAMSPE(VAR MESS      : MESSAGE;
                   VAR NAME      : T_NAME;
                   VAR COM1      : CHAR50;
                   VAR COM2      : CHAR50;
                   VAR NEXT_OP   : OPERATIONS;
                   VAR RR        : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DISPLAYS A CROSS REFERENCE REPORT MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   NAME      I    THE ENTITY NAME
(*   COM1      I    THE MENU COMMENTS
(*   COM2      I    THE MENU COMMENTS
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   INITIALIZE THE VARIABLES
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
(*   DISPLAY THE ENTITY SPECIFICATION MENU
(*   DETERMINE THE ACTION SELECTED FROM THE MENU
(*   REMOVE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
(*
```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(* NONE *)
(* $CHANGE CONTROL: *)
(* REVISED: MM/DD/YY NAME GROUP *)
(* COMMENTS *)
(* REVISED: MM/DD/YY NAME GROUP *)
(* COMMENTS *)
(* REVISED: MM/DD/YY NAME GROUP *)
(* COMMENTS *)
(* ORIGINATED: 11/17/87 C. H. MOHME DBMA *)
(*-----*)
(*END-----*)
(* END %INCLUDE XMNAMSPE *)
```

```
(* %INCLUDE XMPRESPE *)
(**)
PROCEDURE XMPRESPE(VAR MESS          : MESSAGE;
                   VAR DATA_TYPE_KIND : INTEGER;
                   VAR SIZE            : CHAR8;
                   VAR NEXT_OP         : OPERATIONS;
                   VAR RR              : RET_REC);

    SUBPROGRAM;

(**)
(*-----*)
(*
(* $FUNCTION:
(*   DISPLAYS A CROSS REFERENCE REPORT MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   MESS          I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   DATA_TYPE_KIND I  THE DATA TYPE KIND
(*   SIZE          0    THE PRECISION OF THE REAL ENTERED
(*   NEXT_OP       0    ENUMERATED TYPE INDICATING THE NEXT
(*                      OPERATION
(*   RR           0    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                      IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   INITIALIZE THE VARIABLES
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
(*   DISPLAY THE CROSS REFERENCE MENU
(*   DETERMINE THE ACTION SELECTED FROM THE MENU
(*   REMOVE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES
(*
```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(*      NONE      *)
(*      *)
(* $CHANGE CONTROL: *)
(*      *)
(* ORIGINATED: 11/19/87      C. H. MOHME      DBMA      *)
(*      *)
(*-----*)
(*      *)
(*END-----*)
(* END %INCLUDE XMPRESPE*)
```

```
(* %INCLUDE XMRESULT *)
(**)
PROCEDURE XMRESULT(VAR MESS      : MESSAGE;
                   VAR CLAS      : T_ARRAYTV;
                   VAR ARRAY_SIZE : INTEGER;
                   VAR COM1       : CHAR50;
                   VAR COM2       : CHAR50;
                   VAR XREFFILE   : TEXT;
                   VAR NEXT_OP    : OPERATIONS;
                   VAR RR         : RET_REC);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DISPLAYS A CROSS REFERENCE MENU
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL
(*   CLAS      I    THE ARRAY OF MEMBERS
(*   ARRAY_SIZE I    THE SIZE OF THE ARRAY OF MEMBERS
(*   COM1      I    MENU COMMENTS
(*   COM2      I    MENU COMMENTS
(*   XREFFILE  I/O  THE CROSS REFERENCE REPORT FILE
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT
(*                   OPERATION
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND,
(*                   IF ONE HAS, WHAT ROUTINE IT OCCURRED IN
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*   DDNAMES USED WITH STANDARD FILES:
(*   NONE
(*
(* $EXECUTION PROCEDURE:
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   INITIALIZE THE VARIABLES
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM
(*   VARIABLES.
```

CI PS560240032U
April 1990

```
(*      CREATE THE TABLE                      *)
(*      PREPARE TO DISPLAY THE CROSS REFERENCE DISPLAY MENU      *)
(*      LOAD THE TABLE AND WRITE THE CROSS REFERENCE REPORT RESULTS *)
(*      TO FILE                                                  *)
(*      DISPLAY THE CROSS REFERENCE DISPLAY MENU                  *)
(*      DETERMINE THE ACTION SELECTED FROM THE MENU              *)
(*      REMOVE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM *)
(*      VARIABLES                                                *)
(*      REMOVE THE TABLE                                        *)
(*                                                                *)
(*      $COMMENTS:                                              *)
(*      NONE                                                    *)
(*                                                                *)
(*      $CHANGE CONTROL:                                         *)
(*)
```



```
(* %INCLUDE XMTYPSPE *)
(**)
PROCEDURE XMTYPSPE(VAR MESS      : MESSAGE;
                   VAR NEXT_OP  : OPERATIONS;
                   VAR RR       : RET_REC);
    SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   DISPLAYS A CROSS REFERENCE MENU *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  ===== *)
(*   MESS      I    THE ERROR MESSAGE DISPLAYED ON THE PANEL *)
(*   NEXT_OP   O    ENUMERATED TYPE INDICATING THE NEXT *)
(*               OPERATION *)
(*   RR        O    INDICATES IF AN ERROR HAS OCCURRED AND, *)
(*               IF ONE HAS, WHAT ROUTINE IT OCCURRED IN *)
(* *)
(* $COMMONS: *)
(*   NONE *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*   DDNAMES USED WITH STANDARD FILES: *)
(*   NONE *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   SCHEMA EXECUTIVE MENU INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(*   INITIALIZE THE VARIABLES *)
(*   PERMIT THE COMMUNICATION BETWEEN PANEL VARIABLES AND PROGRAM *)
(*   VARIABLES *)
(*   DISPLAY THE CROSS REFERENCE MENU *)
(*   DETERMINE THE ACTION SELECTED FROM THE MENU *)
(*   REMOVE CORRESPONDENCE BETWEEN PANEL VARIABLES AND PROGRAM *)
(*   VARIABLES *)
(* *)
(* $COMMENTS: *)
(*   NONE *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(*) %INCLUDE XNAMENUM *)
(**)
  PROCEDURE XNAMENUM(VAR IRC          : RET_REC;
                    VAR CHAR_STRING   : T_NAME;
                    VAR BAD_NAME      : BOOLEAN;
                    VAR BAD_KIND_NUMBER : BOOLEAN;
                    VAR NUMBER        : INTEGER);

    SUBPROGRAM;

(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   THIS ROUTINE DETERMINES IF A CHARACTER STRING IS A NAME
(*)   OR NUMBER
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O  DESCRIPTION
(*)   ====      ==  =====
(*)   IRC        0   RETURN CODE
(*)   CHAR_STRING I   THE GIVEN CHARACTER STRING
(*)   BAD_NAME    0   INDICATES IF THE GIVEN CHARACTER STRING
(*)                   IS A NAME
(*)   BAD_KIND_NUMBER 0   INDICATES IF THE GIVEN CHARACTER STRING
(*)                   IS A NUMBER
(*)   NUMBER      0   THE NUMBER CONTAINED IN THE CHARACTER
(*)                   STRING
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   DETERMINE IF THE CHARACTER STRING (CHAR_STRING) IS A VALID
(*)   NAME OR KIND NUMBER.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE XRESULT *)
(**)
PROCEDURE XRESULT(VAR IRC          : RET_REC;
                  VAR RESULT_LIST  : LISTKEY;
                  VAR COM1         : CHAR50;
                  VAR COM2         : CHAR50;
                  VAR XREFFILE     : TEXT;
                  VAR OPTION       : OPERATIONS);

SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   THIS ROUTINE DISPLAYS THE CROSS REFERENCE REPORT RESULTS
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   IRC           O    RETURN CODE
(*   RESULT_LIST   I    THE CROSS REFERENCE RESULT LIST
(*   COM1          I    MENU COMMENTS DESCRIBING THE OUTPUT
(*   COM2          I    MENU COMMENTS DESCRIBING THE OUTPUT
(*   XREFFILE      I/O  CROSS REFERENCE REPORT FILE
(*   OPTION        O    OPTION SELECTED
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE SCHEMA EXECUTIVE
(*
(* $PROCESSING DESCRIPTION:
(*   READ EACH KEY OFF OF THE RESULT LIST.
(*   FROM EACH ENTITY ADB OBTAIN THE ENTITY NAME (AND KIND NUMBER,
(*   IF APPLICABLE).
(*   FILL UP THE TABLE OF NAMES AND KIND NUMBERS FOR DISPLAY ON
(*   THE CROSS REFERENCE MENU.
(*   DISPLAY THE CROSS REFERENCE REPORT RESULTS MENU.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```