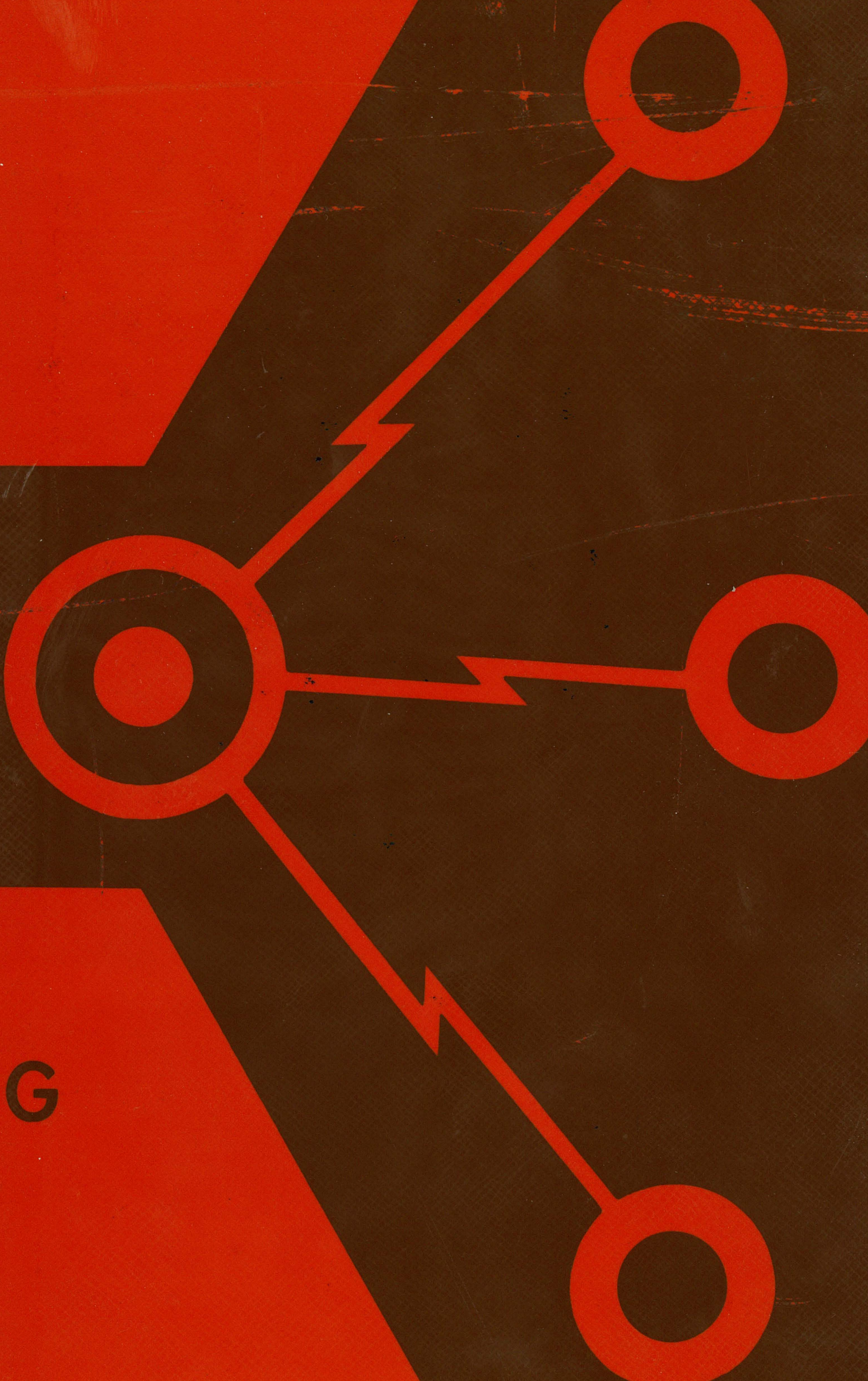
PROGRAMMING STANDARDS AND REFERENCE MANUAL

Warrenville
Data
Center



MANUFACTURING SWITCHING EQUIPMENT DIVISION

Western Electric

DATA PROCESSING SYSTEM PROGRAMMING AIDS

FUNCTION/AID CROSS REFERENCE

SECTION 1 - GENERAL

1. INTRODUCTION

1.1 The Function/Aid Cross Reference List (Appendix A) was designed to assist the user in locating, by function, information pertaining to cataloged procedures, commercial software products, IBM utility programs, and WE macros, utility programs and subroutines, contained in chapters of this Division.

1.2 The functions are arranged in alphabetical sequence and each contains a list of programming aids available. The aids are identified by a section code, footnoted at the bottom of each page, and its name. The section code identifies the Chapter and Section in which the named aid is found and/or indexed for reference to its specific location.

1.3 Many functions indicate the availability of one or more programming aids and/or type of programming aid. The sequence in which the programming aids are listed is alphabetically by aid name and type of programming aid and is not intended to be a recommendation or a preference for use of the first stated aid. The user will have to determine which aid is best suited for the application being considered.

FUNCTION/AID CROSS REFERENCE LIST

--- A ---

ABNORMAL TERMINATION OF A JOB: (PG) WEABEND; (SR) GETDATE

ADD MEMBERS: (CP) CHGPDS; (IBM) IEBUPDAT, IEBUPDTE

ALGOL PROGRAMMING LANGUAGE:
- COMPILE A SOURCE PROGRAM: (CP) ALGOFC
- COMPILE AND EXECUTE USING THE IBM LOADER: (CP) ALGOFCG
- COMPILE AND LINKAGE EDIT: (CP) ALGOFCL
- COMPILE, LINKAGE EDIT, AND EXECUTE: (CP) ALGOFCLG

APPLY AN INTERVAL TO:
- A FIVE DIGIT FISCAL DATE (YYWWD): (SR) CMPDATE5
- A SIX DIGIT CALENDAR DATE (YYMMDD): (SR) CMPDATE6

ASSEMBLER-F PROGRAMMING LANGUAGE:
- COMPILE A SOURCE PROGRAM: (CP) ASMFC
- COMPILE AND EXECUTE USING THE IBM LOADER: (CP) ASMFCG
- COMPILE AND LINKAGE EDIT: (CP) ASMFCL
- COMPILE, LINKAGE EDIT, AND EXECUTE: (CP) ASMFCLG

ASSIGN SEQUENCE NUMBERS: (IBM) IEBUPDAT, IEBUPDTE

--- B ---

BLANKS, REMOVE LEADING: (SR) ADJUSTL

BLOCK SIZE, CHANGE: (CP) WECOPY; (PG) WECOPY, WECOPY2

BUILD A GENERATION DATA GROUP INDEX: (IBM) IEHPROGM

BUILD AN INDEX OR AN INDEX ALIAS: (IBM) IEHPROGM

--- C ---

CALCULATE DIRECT ACCESS:
- SORT WORK SPACE: (CP) SORTSPAC; (PG) SORTSPAC
- SPACE UTILIZATION: (CP) DASPACE; (PG) DASPACE

CALENDAR DATE:
- APPLY AN INTERVAL TO: (SR) CMPDATE6
- CONVERSION: (SEE DATE CONVERSION)

CARD TO STANDARD LABEL TAPE WITH CATALOGING: (CP) WECT

CATALOG:
- A DATA SET: (IBM) IEHPROGM
- A GENERATION DATA SET: (IBM) IEHPROGM

CHANGE:
- BLOCK SIZE: (PG) WECOPY, WECOPY2
- LOGICAL RECORD LENGTH OF A DATA SET: (IBM) IEBGENER
- ORGANIZATION OF A DATA SET: (IBM) IEBUPDTE

- -

CP - CATALOGED PROCEDURE, CH.5
CSP - COMM. SOFTWARE PROD., CH.2, SEC.3
IBM - IBM OS UTIL. PGM., CH.2, SEC.1
M - MACRO, CH.4
PG - W.E. PGM., CH.2, SEC.2
SR - SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

COBOL PROGRAMMING LANGUAGE:
- COMPILE A SOURCE PROGRAM: (CP) COBUC
- COMPILE AND EXECUTE USING THE IBM LOADER: (CP) COBUCG
- COMPILE AND LINKAGE EDIT: (CP) COBUCL
- COMPILE AND OPTIMIZE OBJECT MODULE: (CP) COBUOPT; (CSP) OPTIMIZER II
- COMPILE, LINKAGE EDIT, AND EXECUTE: (CP) COBUCLG
- CONVERT COBOL-F TO ANS COBOL: (CP) META; (CSP) METACOBOL
- LINKAGE EDIT AND EXECUTE: (CP) COBLG

COMPARE DISK AND DISK, TAPE AND TAPE: (IBM) IECOMPR

COMPILE A SOURCE PROGRAM: (CP) ALGOFC, ASMFC, COBUC, FORTGC/FORTHC, PL1LFC, RPGEC

COMPILE AND EXECUTE A SOURCE PROGRAM USING THE IBM LOADER: (CP) ALGOFCG, ASMFCG, COBUCG

COMPILE AND LINKAGE EDIT A SOURCE PROGRAM: (CP) ALGOFCL, ASMFCL, COBUCL, FORTGCL/FORTHCL, PL1FCL, RPGECL

COMPILE AND OPTIMIZE OBJECT MODULE: (CP) COBUOPT; (CSP) OPTIMIZER II

COMPILE, LINKAGE EDIT, AND EXECUTE A SOURCE PROGRAM: (CP) ALGOFCLG, ASMFCLG, COBUCLG, FORTGCLG/FORTHCLG, PL1LFCLG, RPGECLG

COMPRESS A DATA SET: (CP) COMPRESS; (IBM) IEBCOPY

CONDITION CODE:
- PASS: (SR) GETDATE
- SET: (SR) GETDATE

CONNECT VOLUMES: (IBM) IEHPROGM

CONSOLE OPERATOR, COMMUNICATION WITH: (PG) WEWTOR

CONVERSION, GENERAL MEDIA: (PG) WECOPY, WECOPY2

CONVERT:
- CALENDAR AND FISCAL DATES FROM A USER SUPPLIED JULIAN DATE: (SR) GETDATE
- COBOL-F SOURCE PROGRAMS TO ANS COBOL: (CP) META; (CSP) METACOBOL
- FIVE DIGIT FISCAL DATE (YYWWD) TO JULIAN FORMAT (YYDDD) : (SR) FISCALWK
- FIVE DIGIT FISCAL DATE (YYWWD) TO SIX DIGIT CALENDAR DATE (YYMMDD): (SR) CVDATE
- INDEXED SEQUENTIAL TO SEQUENTIAL: (IBM) IEBDG, IEBISAM
- PARTITIONED DATA SET TO SEQUENTIAL DATA SET: (IBM) IEBUPDTE
- SEQUENTIAL DATA SET TO PARTITIONED DATA SET: (IBM) IEBGENER,IEBUPDTE
- SIX DIGIT CALENDAR DATE (YYMMDD) TO JULIAN FORMAT (YYDDD): (SR) CALENDAR
- SIX DIGIT CALENDAR DATE (YYMMDD) TO SIX DIGIT FISCAL DATE (YYMMWD): (SR) CVDATE6F
- SIX DIGIT FISCAL DATE (YYMMWD) TO JULIAN FORMAT (YYDDD): (SR) FISCALPD
- SIX DIGIT FISCAL DATE (YYMMWD) TO SIX DIGIT CALENDAR DATE (YYMMDD): (SR) CVDATEF6

CP - CATALOGED PROCEDURE, CH.5
CSP - COMM. SOFTWARE PROD., CH.2, SEC.3
IBM - IBM OS UTIL. PGM., CH.2, SEC.1
M - MACRO, CH.4
PG - W.E. PGM., CH.2, SEC.2
SR - SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

COPY: (ALSO SEE MOVE)

- A CATALOG: (CP) MOVE; (IBM) IEHMOVE
- A MEMBER OF A PARTITIONED DATA SET: (IBM) IEBCOPY; (PG) WECOPY, WECOPY2
- A PARTITIONED DATA SET: (IBM) IEBCOPY, IEHMOVE
- A SEQUENTIAL DATA SET: (IBM) IEBGENER, IEBUPDTE, IEHMOVE; (PG) WECOPY, WECOPY2
- A VOLUME OF DATA SETS: (IBM) IEHMOVE
- AN INDEXED SEQUENTIAL DATA SET: (IBM) IEBISAM
- CARD TO DISK: (IBM) IEBGENER
- CARD TO PRINTER: (IBM) IEBGENER
- CARD TO PUNCH: (IBM) IEBPTPCH
- CARD TO TAPE: (IBM) IEBGENER
- CATALOGED SEQUENTIAL DATA SETS: (IBM) IEHMOVE
- CONSOLE JCL DECKS: (CP) OPRPNCH
- DIRECT ACCESS FILES, CATALOGS, PDS MEMBERS: (CP) MOVE
- DISK TO DISK: (IBM) IEBCOPY, IEBUPDTE
- DUMPED DATA FROM TAPE TO DIRECT ACCESS: (IBM) IEHDASDR
- JOB STEPS: (IBM) IEBEDIT
- LOAD MODULES, PROCEDURES, OR CONTROL RECORDS: (CP) MOVE
- MEMBERS: (IBM) IEBGENER, IEBUPDAT, IEBUPDTE
- MEMBERS OR DATA SETS: (IBM) IEBUPDTE
- SELECTED MEMBERS: (IBM) IEBCOPY, IEHMOVE
- SELECTED RECORDS OF A SEQUENTIAL DATA SET: (PG) WECOPY, WECOPY2

CORE REQUIREMENTS, REDUCTION ACHIEVED BY SUBSTITUTION OF EFFICIENT CODE: (CP) COBUOPT; (CSP) OPTIMIZER II

CREATE:

- A BACK-UP COPY OF A DATA SET: (IBM) IEBCOPY, IEBGENER
- A MEMBER: (IBM) IEBDG
- A SEQUENTIAL BACK-UP COPY: (IBM) IEBISAM
- A SEQUENTIAL OUTPUT DATA SET: (IBM) IEBDG
- AN INDEX: (IBM) IEHPROGM
- AN INDEXED SEQUENTIAL DATA SET: (IBM) IEBISAM
- AN INPUT STREAM: (IBM) IEBEDIT
- AN OUTPUT DATA SET CONTAINING A SELECTION OF JOBS OR JOB STEPS: (IBM) IEBEDIT
- AN OUTPUT JOB STREAM: (IBM) IEBEDIT
- AN OUTPUT TEST DATA SET: (IBM) IEBDG
- AND UPDATE SYMBOLIC LIBRARIES: (IBM) IEBUPDTE
- USER LABELS ON SEQUENTIAL OUTPUT DATA SETS: (IBM) IEBGENER

--- D ---

DATA BASE / DATA COMMUNICATION SYSTEM: (CSP) IMS (DB & DB/DC)

DATA RETRIEVAL / REPORT FORMATTING SYSTEM: (CP) REACT; (CSP) RE-ACT

DATE CONVERSION:

- CALENDAR AND FISCAL DATES FROM A USER SUPPLIED JULIAN DATE: (SR) GETDATE
- FIVE DIGIT FISCAL (YYWWD) TO JULIAN FORMAT (YYDDD): (SR) FISCALWK
- FIVE DIGIT FISCAL (YYWWD) TO SIX DIGIT CALENDAR (YYMMDD): (SR) CVDATE
- SIX DIGIT CALENDAR (YYMMDD) TO JULIAN FORMAT (YYDDD): (SR) CALENDAR
- SIX DIGIT CALENDAR (YYMMDD) TO SIX DIGIT FISCAL (YYMMWD): (SR) CVDATE6F
- SIX DIGIT FISCAL (YYMMWD) TO JULIAN FORMAT (YYDDD): (SR) FISCALPD
- SIX DIGIT FISCAL (YYMMWD) TO SIX DIGIT CALENDAR (YYMMDD): (SR) CVDATEF6

CP - CATALOGED PROCEDURE, CH.5
CSP - COMM. SOFTWARE PROD., CH.2, SEC.3
IBM - IBM OS UTIL. PGM., CH.2, SEC.1
M - MACRO, CH.4
PG - W.E. PGM., CH.2, SEC.2
SR - SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

DATE MANIPULATION: (SR) GETDATE

DATES COMPUTED FROM A USER SUPPLIED JULIAN DATE (YYDDD): (SR) GETDATE

DEBUGGING AIDS: (SEE DUMP)

DELETE:
  AN INDEX STRUCTURE: (IBM) IEHPROGM
  DATA SETS: (PG) DELETE
  RECORDS: (IBM) IEBUPDAT, IEBUPDTE; (PG) WECOPY2

DIRECT ACCESS:
  FILE DUMP LISTING: (CP) DADUMP
  SORT WORK SPACE, CALCULATE: (CP) SORTSPAC; (PG) SORTSPAC
  SPACE UTILIZATION, CALCULATE: (CP) DASPACE; (PG) DASPACE
  UTILITY SORT: (CP) DASORT

DISK DATA SET INFORMATION: (PG) SLIST

DIVIDE A DATA SET: (PG) WESELECT

DUMP:
  A GROUP OF TRACKS: (IBM) IEHDASDR
  AN ENTIRE DIRECT ACCESS VOLUME: (IBM) IEHDASDR
  CALLABLE SYSUDUMP TYPE CORE DUMP: (SR) WESNAP
  DIRECT ACCESS FILE LISTING: (CP) DADUMP; (PG) WEDEBEPT
  TAPE AND DISK FILES: (PG) WEDEBEPT
  TAPE LISTING: (CP) TPDUMP; (PG) WEDEBEPT

DYNAMIC DEBUGGING INFORMATION FACILITY: (M) DEBUG

--- E ---

EDIT:
  AND CONVERT TO PARTITIONED: (IBM) IEBGENER, IEBUPDTE
  AND COPY A JOB STREAM: (IBM) IEBEDIT
  AND COPY A SEQUENTIAL DATA SET: (IBM) IEBGENER, IEBUPDTE
  AND PRINT/PUNCH: (IBM) IEBPTPCH

ENTER A PROCEDURE INTO A PROCEDURE LIBRARY: (IBM) IEBUPDTE

EXCLUDE:
  DISK TO DISK: (IBM) IEBCOPY
  MEMBERS FROM BEING COPIED: (IBM) IEBCOPY, IEHMOVE
  RECORDS FROM BEING COPIED: (PG) WECOPY2

EXPAND:
  A PARTITIONED DATA SET: (IBM) IEBCOPY, IEBGENER
  A SEQUENTIAL DATA SET: (IBM) IEBGENER
  DISK TO DISK: (IBM) IEBCOPY
  OR COMPRESS: (IBM) IEBGENER

EXTRACT DESIRED RECORD TYPES: (PG) WESELECT

CP	- CATALOGED PROCEDURE, CH.5	M	- MACRO, CH.4
CSP	- COMM. SOFTWARE PROD., CH.2, SEC.3	PG	- W.E. PGM., CH.2, SEC.2
IBM	- IBM OS UTIL. PGM., CH.2, SEC.1	SR	- SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

--- F ---

FIELD SIZE, LENGTHEN/SHORTEN: (SR) ADJUSTL

FISCAL DATE:
  APPLY AN INTERVAL TO: (SR) CMPDATE5
  CONVERSION: (SEE DATE CONVERSION)

FLOWCHART SOURCE LANGUAGE STATEMENTS: (CP) AUTOFLOW; (CSP) AUTOFLOW

FORCE ABNORMAL TERMINATION OF A JOB: (PG) WEABEND

FORMATTING FILES FOR TRANSMISSION: (SR) PUTTMT

FORTRAN (G & H) PROGRAMMING LANGUAGE:
  COMPILE A SOURCE PROGRAM: (CP) FORTGC/FORTHC
  COMPILE AND LINKAGE EDIT: (CP) FORTGCL/FORTHCL
  COMPILE, LINKAGE EDIT, AND EXECUTE: (CP) FORTGCLG/FORTHCLG
  LINKAGE EDIT AND EXECUTE: (CP) FORTLG

--- G ---

GENERAL MEDIA CONVERSION: (PG) WECOPY, WECOPY2

GENERATE:
  FLOWCHART, SOURCE LISTING, AND CROSS REFERENCE LISTING:
    (CP) AUTOFLOW; (CSP) AUTOFLOW
  INSTRUCTIONS FOR RESTORING GENERAL REGISTERS AND RETURN CONTROL TO
   THE CALLING PROGRAM: (M) EXIT
  INSTRUCTIONS REQUIRED AT THE BEGINNING OF AN ASSEMBLER LANGUAGE
   PROGRAM: (M) ENTER
  TEST DATA: (IBM) IEBDG

GENERATION DATA GROUP INDEX, BUILD A: (IBM) IEHPROGM

GET ALTERNATE TRACKS ON A DIRECT ACCESS VOLUME:
  (IBM) IEHATLAS, IEHDASDR

--- I ---

INITIALIZE A DIRECT ACCESS VOLUME: (IBM) IEHDASDR

INSERT RECORDS: (IBM) IEBUPDTE

ISAM FILE INFORMATION: (PG) LYNETTE

ISSUE ''RESERVE'' FOR UPDATING:
  PUBLIC CARD IMAGE LIBRARIES: (PG) ATTUPDTE
  PUBLIC LIBRARIES: (PG) ATTCOPY
  PUBLIC LOAD MODULE LIBRARIES: (PG) ATTIEWL

--- J ---

JCL CONSOLE DECKS, COPY: (CP) OPRPNCH

JULIAN FORMAT (YYDDD) CONVERSION: (SEE DATE CONVERSION)

CP - CATALOGED PROCEDURE, CH.5	M - MACRO, CH.4
CSP - COMM. SOFTWARE PROD., CH.2, SEC.3	PG - W.E. PGM., CH.2, SEC.2
IBM - IBM OS UTIL. PGM., CH.2, SEC.1	SR - SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

--- L ---

LIBRARIAN:
- GENERALIZED DATA STORAGE AND RETRIEVAL SYSTEM: (CSP) LIBRARIAN
- PARTITIONED DATA SET OUTPUT: (CP) LIBP
- SEQUENTIAL OUTPUT: (CP) LIBS
- UTILITY PROCEDURE: (CP) LIBU

LINK EDIT AND PRODUCE A LOAD MODULE: (CP) LKED

LINK EDIT, PRODUCE A LOAD MODULE, AND EXECUTE: (CP) LKEDG

LIST (ALSO SEE PRINT):
- A MEMBER OF A CARD IMAGE PARTITIONED DATA SET: (CP) LISTOUT
- A VOLUME TABLE OF CONTENTS: (IBM) IEHLIST
- ATTRIBUTES FOR DISK DATA SETS: (CP) LISTDISK; (PG) SLIST
- CATALOGS: (CP) LIST, WELIST; (IBM) IEHLIST
- CONTENTS OF A DIRECT ACCESS VOLUME AND ATTRIBUTES OF DIRECT ACCESS DATA SETS: (CP) LISTDISK
- CONTENTS OF ALL MEMBERS OF A PARTITIONED DATA SET: (CP) LISTPDS
- CONTENTS OF TAPE LIBRARY SYSTEM BIN TABLE: (CP) TLSBINL
- ENTRIES IN A CATALOG: (IBM) IEHLIST
- ENTRIES IN A VOLUME TABLE OF CONTENTS: (CP) WELIST; (IBM) IEHLIST
- ENTRIES IN THE DIRECTORIES OF ONE OR MORE PARTITIONED DATA SETS: (IBM) IEHLIST
- NUMBER OF UNUSED DIRECTORY BLOCKS AND TRACKS: (IBM) IEBCOPY
- PARTITIONED DATA SET DIRECTORIES: (CP)LIST, WELIST; (IBM) IEHLIST
- TABLE OF CONTENTS OF A DISK PACK: (CP) LISTDISK
- VOLUME TABLE OF CONTENTS OF THE SYSTEM RESIDENT PACK: (CP) LIST, WELIST

LOAD:
- AN UNCATALOGED DATA SET: (IBM) IEHMOVE
- TAPE TO DISK: (IBM) IEBISAM

LOGICAL RECORD LENGTH, REDUCE: (PG) WECOPY, WECOPY2

--- M ---

MEASURE EFFICIENCY OF PROBLEM PROGRAM: (CP) PPANAL,PPEXT; (CSP)SMS/360

MEDIA CONVERSION, GENERAL: (PG) WECOPY, WECOPY2

MERGE:
- PARTITIONED DATA SETS: (IBM) IEBCOPY, IEHMOVE
- RECORDS: (CP) DASORT; (CSP) SORT/MERGE SM1

MODIFY AN EXISTING DATA SET: (IBM) IEBUPDTE

MOVE (ALSO SEE COPY):
- A CATALOG: (IBM) IEHMOVE
- A GROUP OF CATALOGED DATA SETS: (IBM) IEHMOVE
- A VOLUME OF DATA SETS: (IBM) IEHMOVE
- PARTITIONED DATA SETS: (IBM) IEHMOVE
- SEQUENTIAL DATA SETS: (IBM) IEHMOVE

- -

CP - CATALOGED PROCEDURE, CH.5
CSP - COMM. SOFTWARE PROD., CH.2, SEC.3
IBM - IBM OS UTIL. PGM., CH.2, SEC.1

M - MACRO, CH.4
PG - W.E. PGM., CH.2, SEC.2
SR - SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

--- N ---

NUMBER RECORDS: (IBM) IEBUPDTE

--- O ---

OBTAIN COPY OF CONSOLE JCL DECKS: (CP) OPRPNCH

OBTAIN EXEC CARD PARM INFORMATION: (SR) PARMEXCD

OPTIMIZE OBJECT MODULE:
  ANS COBOL: (CP) COBUOPT; (CSP) OPTIMIZER II

ORGANIZATIONAL INFORMATION FOR AN ISAM FILE: (PG) LYNETTE

OUTPUT SUMMARY GENERATION: (CP) WEOUTPUT

--- P ---

PARM INFORMATION, OBTAIN FROM EXEC CARD: (SR) GETDATE, PARMEXCD

PARM PASSING: (SR) GETDATE

PL/I PROGRAMMING LANGUAGE:
  COMPILE A SOURCE PROGRAM: (CP) PL1LFC
  COMPILE AND LINKAGE EDIT: (CP) PL1LFCL
  COMPILE, LINKAGE EDIT, AND EXECUTE: (CP) PL1LFCLG
  LINKAGE EDIT AND EXECUTE: (CP) PL1LFLG

PREPARATION FOR UPDATE OF PRODUCTION LIBRARIES: (CP) PLCPREP

PRINT (ALSO SEE LIST):
  A SEQUENTIAL FILE WITH UNKNOWN CHARACTERISTICS: (CP) DADUMP, TPDUMP;
    (PG) WEDEBEPT
  AN INDEXED SEQUENTIAL DATA SET: (IBM) IEBISAM
  CARD TO SYSTEM OUTPUT: (IBM) IEBGENER
  FIXED BLOCKED RECORDS OF LENGTHS UNACCEPTABLE TO ASP:
    (PG) WECOPY, WECOPY2
  GENERAL PURPOSE REGISTERS AND SELECTED AREAS OF CORE: (M) DEBUG
  SELECTED RECORDS: (IBM) IEBPTPCH
  SELECTED RECORDS IN CHARACTER AND/OR HEX FORMAT: (PG) VFPRINT
  TAPE/DISK TO SYSTEM OUTPUT: (IBM) IEBPTPCH

PRINTER TESTING, ASP RJP WORKSTATION: (PG) WETESTPR

PROBLEM PROGRAM EFFICIENCY:
  ANALYZER: (CP) PPANAL; (CSP) SMS/360
  EXTRACTOR: (CP) PPEXT; (CSP) SMS/360

PRODUCE A PARTITIONED DATA SET FROM SEQUENTIAL INPUT: (IBM) IEBGENER

PRODUCE FLOWCHARTS FROM SOURCE LANGUAGE STATEMENTS:
  (CP) AUTOFLOW; (CSP) AUTOFLOW

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

PROGRAM:
  INITIALIZATION - ASSEMBLER LANGUAGE: (M) ENTER
  PERFORMANCE STUDY: (CSP) SMS/360
  TERMINATION - ASSEMBLER LANGUAGE: (M) EXIT

PRODUCTION LIBRARIES, PREPARATION FOR UPDATE: (CP) PLCPREP

PUNCH: (IBM) IEBPTPCH

--- R ---

REBLOCK A DATA SET: (IBM) IEBDG, IEBGENER, IEHMOVE; (PG) WECOPY, WECOPY2

REDUCE LOGICAL RECORD LENGTH: (PG) WECOPY, WECOPY2

REMOVE A GROUP OF RECORDS FROM THE INTERIOR OF A DATA SET: (PG)WECOPY2

REMOVE LEADING BLANKS AND LEFT ADJUST A FIELD: (SR) ADJUSTL

RENAME:
  A DATA SET: (IBM) IEHPROGM
  DISK TO DISK: (IBM) IEBCOPY
  MOVED OR COPIED MEMBERS: (IBM) IEHMOVE

RENUMBER RECORDS: (IBM) IEBUPDAT, IEBUPDTE

REORGANIZE AN INDEXED SEQUENTIAL DATA SET: (IBM) IEBISAM

REPLACE:
  IDENTICALLY NAMED DATA SET MEMBERS: (IBM) IEBCOPY
  RECORDS: (IBM) IEBCOPY, IEBUPDAT, IEBUPDTE
  SELECTED MEMBERS IN A MOVE OR COPY OPERATION: (IBM) IEBCOPY, IEHMOVE

RESERVE, ISSUE FOR UPDATING:
  PUBLIC CARD IMAGE LIBRARIES: (PG) ATTUPDTE
  PUBLIC LIBRARIES: (PG) ATTCOPY
  PUBLIC LOAD MODULE LIBRARIES: (PG) ATTIEWL

RESTORE DATA ON DIRECT ACCESS VOLUMES: (IBM) IEHDASDR

RETRIEVE AND UPDATE CONTROL TYPE RECORDS IN PLACE: (SR) CCRETREV/CCUPDATE

RPG PROGRAMMING LANGUAGE:
  COMPILE A SOURCE PROGRAM: (CP) RPGEC
  COMPILE AND LINKAGE EDIT: (CP) RPGECL
  COMPILE, LINKAGE EDIT, AND EXECUTE: (CP) RPGECLG

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

--- S ---

SCRATCH:
- A CATALOG: (CP) MOD, WEMOD
- A DATA SET OR A MEMBER: (IBM) IEHPROGM; (PG) DELETE
- A VOLUME TABLE OF CONTENTS: (IBM) IEHPROGM

SELECT:
- AND PRINT RECORDS IN CHARACTER AND/OR HEX FORMAT: (PG) VFPRINT
- MEMBERS TO BE COPIED: (IBM) IEBCOPY
- SPECIFIED RECORDS: (PG) WESELECT

SET CONDITION CODE: (SR) GETDATE

SHARING DISK DATA SETS BETWEEN CPU'S: (PG) ATTCOPY, ATTIEWL, ATTUPDTE

SORT, CALCULATE DISK WORK SPACE: (CP) SORTSPAC; (PG) SORTSPAC

SORT/MERGE:
- RECORDS: (CP) DASORT; (CSP) SORT/MERGE-SMI
- WITH NO ROUTINES OR WITH ROUTINES THAT DO NOT REQUIRE LINK EDITING: (CP) SORTD
- WITH ROUTINES THAT REQUIRE LINK EDITING: (CP) SORT

SOURCE PROGRAM:
- DOCUMENTATION: (CP) AUTOFLOW; (CSP) AUTOFLOW
- MAINTENANCE: (CP) LIBP, LIBS, LIBU; (CSP) LIBRARIAN

STORAGE OF CARD IMAGE STATEMENTS: (CP) LIBP, LIBS, LIBU; (CSP) LIBRARIAN

SUMMARY LISTING OF PRODUCTION SYSTEM OUTPUT WHICH REQUIRES FURTHER PROCESSING: (CP) WEOUTPUT

--- T ---

TAPE:
- DUMP: (CP) TPDUMP; (PG) WEDEBEPT
- TO PRINT: (PG) WEDEBEPT

TEXT EDITING AND FORMATTING: (CSP) MITS

TRANSFER A SOURCE PDS TO A LIBRARIAN DATA SET: (PG) LIBRLIST

TRANSMISSION FILES:
- FORMATTING: (SR) PUTTMT
- UNFORMATTING: (SR) GETTMT

- -

CP - CATALOGED PROCEDURE, CH.5	M - MACRO, CH.4
CSP - COMM. SOFTWARE PROD., CH.2, SEC.3	PG - W.E. PGM., CH.2, SEC.2
IBM - IBM OS UTIL. PGM., CH.2, SEC.1	SR - SUBROUTINE, CH.3

FUNCTION/AID CROSS REFERENCE LIST (CONT.)

--- U ---

UNCATALOG DATA SETS: (IBM) IEHPROGM; (PG) DELETE

UNFORMATTING TRANSMISSION FILES: (SR) GETTMT

UNLOAD:
  A PARTITIONED DATA SET: (IBM) IEHMOVE
  A SEQUENTIAL DATA SET: (IBM) IEHMOVE
  AN INDEXED SEQUENTIAL DATA SET: (IBM) IEBISAM

UPDATE:
  AN EXISTING CARD IMAGE PDS: (CP) CHGPDS
  CONTROL RECORDS IN PLACE: (SR) CCRETREV/CCUPDATE
  IN PLACE: (IBM) IEBUPDTE
  PRODUCTION LIBRARIES, PREPARATION FOR: (CP) PLCPREP

--- V ---

VTOC, LIST CONTENTS OF SYSTEM RESIDENT PACK: (CP) LIST, WELIST

--- W ---

WRITE TO OPERATOR: (PG) WEWTOR

DATA PROCESSING SYSTEM PROGRAMMING AIDS

UTILITY PROGRAMS

SECTION 1 - I.B.M. OS UTILITIES

1. INTRODUCTION

1.1 I.B.M. Utility Programs are presented in this Section for the purpose of informing the application programmer of the general functions and services performed by these programs. This material is not presented as a tutorial on I.B.M. Utility Programs, hence, no mention is made of Utility Control Statements, Exit Routine Linkage, Invoking Utility Programs, Generation Data Groups, etc. For a complete description of I.B.M. Utility Programs and an explanation on how to use these programs, refer to I.B.M. Operating System Utilities Manual, GC28-6586.

1.2 The Operating System Utility Programs perform functions to assist programmers responsible for creating and maintaining operating system data. These functions are requested through control statements, which can be used individually or in combination, to perform a variety of housekeeping and support operations. The function that the utility program performs and the manner in which the program is controlled determines the program class of the utility program, either System Utility program or Data Set Utility program.

2. SYSTEM UTILITY PROGRAMS

2.1 System Utility programs manipulate collections of data and system control information. The programs are used to maintain system control data at an organizational or system level. Several programs in this class may be useful to an application programmer. The following paragraphs present a brief description of the general functions of these programs.

2.11 IEHDASDR - a program used to prepare direct access volumes for operating system use and to ensure that any permanent hardware errors (i.e., defective tracks) incurred on direct access volumes do not seriously degrade the performance of those volumes. The program can be used to dump the entire contents of a direct access volume to a volume of the same device type, to a tape volume, or to a system output device. Data that is dumped to a tape volume is arranged so that it can subsequently be restored to its original organization by IEHDASDR.

2.12 IEHLIST - a program used to list entries in a catalog, entries in the directory of one or more partitioned data sets, or entries in a volume table of contents.

2.13 IEHMOVE - a program used to move or copy collections of data including:

a. A data set residing on from one to five volumes
b. A group of cataloged data sets

c. A catalog, or portions of a catalog
d. A volume of data sets

The scope of a basic move or copy operation can be enlarged by:

a. Merging members from two or more partitioned data sets
b. Including or excluding selected members
c. Renaming moved or copied members
d. Replacing selected members
e. Including or excluding data sets from a move or copy operation

When using the IEHMOVE program, the user should be aware of the difference between a move operation and a copy operation. A move operation scratches source data (from direct access volumes only) while a copy operation leaves source data intact. In addition, for cataloged data sets, a move operation updates the catalog to refer to the moved version (unless otherwise specified), while a copy operation leaves the catalog unchanged.

2.14 IEHPROGM - a program used to modify system control data and to maintain data sets at an organizational level. The program can be used to:

a. Scratch a data set or a member
b. Rename a data set or a member
c. Catalog or uncatalog a data set
d. Build or delete an index or an index alias
e. Connect or release two volumes
f. Build and maintain a generation data group index
g. Maintain data set passwords

3. DATA SET UTILITY PROGRAMS

3.1 Data Set Utility programs manipulate partitioned, sequential, or indexed sequential data sets provided as input to the programs. Data ranging from fields within a logical record to entire data sets can be manipulated. Programs in this class and a brief description of the general functions performed by each program are presented in the following paragraphs.

3.11 IEBCOMPR - a program used to compare two identically organized data sets at the logical record level. Data sets to be compared can be either sequential or partitioned. The program can be used to:

a. Verify a back-up copy of a sequential or partitioned data set
b. Verify portions of records within a sequential or partitioned data set

3.12 IEBCOPY - a program used to copy one or more partitioned data sets or to merge partitioned data sets. Specified members of partitioned data sets can be selected for, or excluded from, a copy process. The program can be used to:

a. Create a back-up copy
b. Copy data sets

c. Select members, from one or more data sets, to be copied
d. Replace identically named members on data sets
e. Replace selected data set members
f. Rename selected members
g. Exclude members, from one or more data sets, from being copied
h. Compress a data set in place
i. Merge data sets
j. Recreate a data set that has exhausted its primary, secondary, or directory space allocation

3.13 IEBDG - a program that provides a "pattern" of test data to be used as a programming debugging aid. An output test data set, containing records of any format, can be created through the use of utility control statements, with or without input data. Sequential, indexed sequential, and partitioned data sets can be used for input or output.

3.14 IEBEDIT - a program that creates an output data set containing a selection of jobs or job steps which, at a later time, can be used as an input stream for job processing. The output data set is created by editing and selectively copying a job stream provided as input. The program can copy:

a. An entire job or jobs, including JOB statements and associated JOBLIB statements, if any
b. Portions of a job (i.e., selected job steps), including the JOB statement and its associated JOBLIB statement, if any

3.15 IEBGENER - a program that can copy a sequential data set or a partitioned member, or it can create a partitioned data set from a sequential data set or from a partitioned member used as input. The program can expand an existing partitioned data set by creating partitioned members and merging them into the data set that is to be expanded. The program can be used to:

a. Create a back-up copy of a sequential data set or a partitioned member
b. Produce a partitioned data set from sequential input
c. Expand an existing partitioned data set
d. Produce an edited sequential or partitioned data set
e. Reblock or change the logical record length of a data set
f. Create user labels on sequential output data sets

3.16 IEBISAM - a program that can copy an indexed sequential data set directly from one direct access volume to another. Alternatively, the program can be used to reorganize an indexed sequential data set into a sequential (unloaded) data set on a direct access volume or a magnetic tape volume. The unloaded data set is in a form that can be converted back into an indexed sequential data set. The program can be used to:

a. Copy an indexed sequential data set
b. Create a sequential back-up copy of source data from an indexed sequential data set
c. Create an indexed sequential data set from an unloaded data set
d. Print an indexed sequential data set

3.17 IEBPTPCH - a program that prints or punches all, or selected portions, of a sequential or partitioned data set. Records can be printed or punched to meet either standard specifications or user specifications. The program provides optional editing facilities with each application of the program. The program can be used to:

a. Print or punch a sequential or partitioned data set in its entirety
b. Print or punch selected members from a partitioned data set
c. Print or punch selected records from a sequential or partitioned data set
d. Print or punch the directory of a partitioned data set
e. Print or punch an edited version of a sequential or partitioned data set

3.18 IEBUPDAT - a program that incorporates both I.B.M. and user-generated source language modifications into symbolic libraries. The program can be used to:

a. Add, copy, and replace members
b. Add, delete, replace, and renumber the records within an existing member
c. Assign sequence numbers to the records of a new member

3.19 IEBUPDTE - a program that incorporates both I.B.M. and user-generated source language modifications into sequential data sets or into partitioned data sets. The program is used to create and update card image libraries, incorporate changes to partitioned members or sequential data sets, and change the organization of a data set from sequential to partitioned or partitioned to sequential. The program can:

a. Add, copy, and replace members or data sets
b. Add, delete, replace, and renumber the records within an existing member or data set
c. Assign sequence numbers to the records of a member or data set
d. Convert sequential input into partitioned output or vice versa

DATA PROCESSING SYSTEM PROGRAMMING AIDS

UTILITY PROGRAMS

SECTION 2 - WESTERN ELECTRIC UTILITIES

1. INTRODUCTION

1.1 The programs described in thi s section were developed by Western Electric personnel or are programs which were modified by Western Electric personnel to meet local processing requirements.

1.2 In general, this material should enable the user to make effective use of these programs. For those programs which call IBM Programs, reference is made to an IBM manual where additional information is available to the user.

1.3 For many programs, reference is made to a cataloged procedure which has been prepared to facilitate use of the program.

1.4 Appendix A, Western Electric Utility Program Reference List, directs the user to a description of the programs and examples of use.

WESTERN ELECTRIC UTILITY PROGRAM REFERENCE LIST

PROGRAM	FUNCTION	APPENDIX	PAGE
ATTCOPY	Issue "RESERVE" for updating public libraries	B	3
ATTIEWL	Issue "RESERVE" for updating public load module libraries	C	4
ATTUPDTE	Issue "RESERVE" for updating public card image libraries	D	5
CATALIST	List catalog node points	X	39
CATAUTIL	Catalog maintenance	Y	41
DASPACE	Compute direct access space utilization data for fixed length records	R	29
DELETE	Scratch/Uncatalog data sets	E	6
DPMPSIL	List contents of the PSI table	S	32
LIBRLIST	Partitioned data set to librarian data set	Q	27
LYNETTE	ISAM file information	F	7
SLIST	Disk data set information	G	8
SORTSPAC	Calculate direct access sort workspace	H	9
TRMUST06	Check for entries in a generation data group	W	38
VFPRINT	Selective print dump	P	24
WEABEND	Force abnormal termination	I	11
WECOPY	General media conversion	J	12
WECOPY2	General media conversion (modified version of WECOPY)	K	15
WEDEBEPT	Tape and "disk file" dump	N	20
WEMFCOPY	Multiple file copy	T	33
WESELECT	Select specified records	L	16
WESFSPRT	Sequential file selective print	U	35
WETESTPR	ASP RJP workstation printer testing	O	23
WEWTOR	Write to operator	M	19
ZEROCORE	Zero user region in main storage	V	37

TRMUST06 - CHECK FOR ENTRIES IN A GENERATION DATA GROUP

1. GENERAL

1.1 TRMUST06 is a Western Electric utility program developed for application systems interfacing with Mini-WEDGE. The function of this program is to determine if there are any entries in the generation data group specified by the user via a PARM parameter in the JCL EXEC statement.

1.2 An application program can determine if an input file has been UNSTACKed by checking a condition code. The condition code (General Register 15) is set to zero (0) if relative generation zero (0) or other explicitly specified relative generation exists, otherwise it is set to 04 with one exception: if a PARM value is not supplied by the user, the condition code is set to 08.

1.3 The program does not make a validity check for a valid GDG DSNAME base (or index).

1.4 The minimum core requirement is 6K.

2. EXAMPLES OF USE

2.1 In the following examples, assume ABC.DEF to be the DSNAME base (or index) of a generation data group.

2.11
```
//   EXEC   PGM=TRMUST06,PARM='ABC.DEF'
```

2.12 An equivalent expression to the above example is:

```
//   EXEC   PGM=TRMUST06,PARM='ABC.DEF(0)'
```

2.13 The existence of old generations may also be tested

```
//   EXEC   PGM=TRMUST06,PARM='ABC.DEF(-2)'
```

The return code will be set to 04 unless at least three generations are cataloged.

3. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

WESTERN ELECTRIC UTILITY PROGRAM REFERENCE LIST

PROGRAM	FUNCTION	APPENDIX	PAGE
ATTCOPY	Issue "RESERVE" for updating public libraries	B	3
ATTIEWL	Issue "RESERVE" for updating public load module libraries	C	4
ATTUPDTE	Issue "RESERVE" for updating public card image libraries	D	5
DASPACE	Compute direct access space utilization data for fixed length records	R	29
DELETE	Scratch/Uncatalog data sets	E	6
DPMPSIL	List contents of the PSI table	S	32
LIBRLIST	Partitioned data set to librarian data set	Q	27
LYNETTE	ISAM file information	F	7
SLIST	Disk data set information	G	8
SORTSPAC	Calculate direct access sort workspace	H	9
TRMUST06	Check for entries in a generation data group	W	38
VFPRINT	Selective Print Dump	P	24
WEABEND	Force abnormal termination	I	11
WECOPY	General media conversion	J	12
WECOPY2	General media conversion (modified version of WECOPY)	K	15
WEDEBEPT	Tape and "disk file" dump	N	20
WEMFCOPY	Multiple file copy	T	33
WESELECT	Select specified records	L	16
WESFSPRT	Sequential file selective print	U	35
WETESTPR	ASP RJP Workstation Printer Testing	O	23
WEWTOR	Write to operator	M	19
ZEROCORE	Zero user region in main storage	V	37

ATTCOPY - ISSUE "RESERVE" FOR UPDATING PUBLIC LIBRARIES

1. GENERAL

1.1 ATTCOPY, a Western Electric program which calls IEBCOPY, is designed to insure integrity of partitioned data sets accessible by multiple CPU's. The simultaneous accessing of a disk pack by more than one CPU could result in the action of one interferring with the other, resulting in lost or destroyed data. Protection of the data sets is provided by ATTCOPY's issuance of "RESERVE" for the direct access unit and effectively prevents access by other CPUs until the device is "DEQUEUED" by the updating program.

1.2 ATTCOPY must be used when updating any of the public libraries, for example:

DEV.LOADLIB	PLC.CONTROL	PLC.PROCLIB
DEVnn.LOADLIB	PLC.JCLLIB	
*PRDn.PARMLIB	PLC.LOADLIB	

*PRDn.PARMLIB can be updated using CCUPDATE because the update is "INPLACE". (See Division 3, Chapter 3, Section 1, Subroutines).

2. USING ATTCOPY

2.1 ATTCOPY issues a "reserve" for the output data set specified in a SYSUT2 DD statement, and calls the IBM Utility Program, IEBCOPY.

2.2 The JCL for executing ATTCOPY is the same as that for executing IEBCOPY with the following exception: The output data set must be specified on a DD statement with the DDNAME of SYSUT2. The absence of the SYSUT2 DD statement will result in a user 0012 abend.

2.3 Information about using IEBCOPY may be found in the IBM OS Utilities Manual, GC28-6586.

3. PRECAUTIONS

3.1 Severe system degradation could occur through the unnecessary use of reserve as all other CPU's are locked out of the entire disk pack (not just the data set) for as long as the "reserve" is in effect.

3.2 Many public cataloged procedures execute programs which issue the reserve. Many do not issue the reserve as they are not designed to update public data sets. Programmers must know what programs are executed by the procedures they use. Listings of cataloged procedures are available in the programmer service area.

3.3 Programmers updating private data sets, including all temporary data sets, will probably find the use of "reserve" unnecessary.

ATTIEWL - ISSUE "RESERVE" FOR UPDATING PUBLIC LOAD MODULE LIBRARIES

1. GENERAL

1.1 ATTIEWL, a Western Electric program which calls IEWL, is designed to insure integrity of partitioned data sets accessible by multiple CPU's. The simultaneous accessing of a disk pack by more than one CPU could result in the action of one interfering with the other, resulting in lost or destroyed data. Protection of the data sets is provided by ATTIEWL's issuance of a "RESERVE" for the direct access unit and effectively prevents access by other CPU's until the device is "DEQUEUED" by the updating program.

1.2 ATTIEWL must be used when updating any of the public load module libraries, for example:

DEV.LOADLIB

DEVnn.LOADLIB

2. USING ATTIEWL

2.1 ATTIEWL issues a "reserve" for the output data set specified in the SYSLMOD DD statement, and calls the IBM Linkage Editor.

2.2 The JCL for executing ATTIEWL is the same as that for executing the Linkage Editor. Information about using the Linkage Editor may be found in the IBM Linkage Editor Manual, C28-6538.

3. PRECAUTIONS

3.1 Severe system degradation could occur through the unnecessary use of reserve as all other CPU's are locked out of the entire disk pack (not just the data set) for as long as the "reserve" is in effect.

3.2 Many public cataloged procedures execute programs which issue the reserve. Many do not issue the reserve as they are not designed to update public data sets. Programmers must know what programs are executed by the procedures they use. Listings of cataloged procedures are available in the programmer service area.

3.3 Programmers updating private data sets, including all temporary data sets, will probably find the use of "reserve" unnecessary.

ATTUPDTE - ISSUE "RESERVE" FOR UPDATING PUBLIC CARD IMAGE LIBRARIES

1. GENERAL

1.1 ATTUPDTE, a Western Electric program which calls IEBUPDTE, is designed to insure integrity of partitioned data sets accessible by multiple CPU's. The simultaneous accessing of a disk pack by more than one CPU could result in the action of one interferring with the other, resulting in lost or destroyed data. Protection of the data sets is provided by ATTUPDTE's issuance of a "RESERVE" for the direct access unit and effectively prevents access by other CPU's until the device is "DEQUEUED" by the updating program.

1.2 ATTUPDTE must be used when updating any of the public card image libraries, for example:

PDNn.PARMLIB*
PLC.CONTROL
PLC.JCLLIB
PLC.PROCLIB

*PRDn.PARMLIB can be updated using CCUPDATE because the update is "INPLACE". (See Division 3, Chapter 3, Section 1, Subroutines).

2. USING ATTUPDTE

2.1 ATTUPDTE issues a "reserve" for the output data set specified in a SYSUT2 DD statement, and calls the IBM Utility Program, IEBUPDTE.

2.2 The JCL for executing ATTUPDTE is the same as that for executing IEBUPDTE. Information about using IEBUPDTE may be found in the IBM OS Utilities Manual, GC28-6586.

3. PRECAUTIONS

3.1 Severe system degradation could occur through the unnecessary use of reserve as all other CPU's are locked out of the entire disk pack (not just the data set) for as long as the "reserve" is in effect.

3.2 Many public cataloged procedures execute programs which issue the reserve. Many do not issue the reserve as they are not designed to update public data sets. Programmers must know what programs are executed by the procedures they use. Listings of cataloged procedures are available in the programmer service area.

3.3 Programmers updating private data sets, including all temporary data sets, will probably find the use of "reserve" unnecessary.

DELETE - SCRATCH/UNCATALOG DATA SETS

1. GENERAL

1.1 DELETE is a Western Electric utility program which may be used to scratch and/or uncatalog data sets from storage packs (see Division 2, Direct Access Storage). While the IBM Utility Program IEHPROGM performs the same function, DELETE differs in several respects:

1. No knowledge of where the data sets are located is required. DELETE will search for the data sets on every storage pack.

2. If a data set exists on more than one pack, all will be deleted. Only one control statement is necessary.

3. If the data set is cataloged, it will be automatically uncataloged.

4. DD statements allocating the storage packs are not required. Thus, it is not necessary to know what storage packs are currently mounted.

5. Generation data sets on disk may be specified with a relative generation number in parentheses.

2. CODING REQUIREMENTS

2.1 Two DD statements are required for proper execution of DELETE:

1. SYSIN defines a control statement input data set.
2. SYSPRINT defines a diagnostic output data set.

2.2 The name of the data set to be uncataloged and/or scratched should be punched in columns 1 through 44 of the control statement. Columns 45 through 80 are ignored and may be used for comments. Any number of control statements may be submitted at a time.

2.3 JCL statements for executing DELETE are:

```
//    JOB card
//  EXEC  PGM=DELETE,REGIØN=30K
//SYSPRINT DD  SYSØUT=A
//SYSIN    DD  *
DSNAME1
DSNAME2
 .
DSNAMEn
/*
```

LYNETTE - ISAM FILE INFORMATION

1. GENERAL

1.1 LYNETTE is a Western Electric modified version of an IBM Type III Utility Program which may be used to provide organizational information for an Indexed Sequential Access Method (ISAM) file. The program produces information which is useful to the user in the maintenance of an ISAM file.

1.2 The following organizational statistics are provided by the LYNETTE program:

1. Number of records residing in the prime data area
2. Number of records residing in an overflow area
3. Number of records tagged for deletion
4. Number of overflow tracks allocated per cylinder
5. Number of cylinder overflow areas full
6. Number of tracks remaining in the independent overflow area
7. Number of accesses to overflow records other than the first
8. Number of index levels
9. Number of tracks used for highest index
10. Number of bytes needed for highest index
11. Number of possible physical records per prime track
12. Blocksize
13. Logical record length
14. Number of possible records per overflow track

1.3 JCL statements for executing LYNETTE are:

```
//  EXEC  PGM=LYNETTE,REGIØN=60K
//SYSUT1    DD  Parameters defining input data set
//SYSPRINT DD  SYSØUT=A
//SYSØUT    DD  SYSØUT=A
//SYSIN     DD  *
  INFØ
/*
```

SLIST - DISK DATA SET INFORMATION

1. GENERAL

1.1 SLIST is a Western Electric modified version of an IBM Type III Utility Program which may be used to provide information about the attributes of disk data sets. The following statistics are provided by the SLIST program:

1. Creation date
2. Expiration date
3. Data set type
4. Record format
5. Logical record length
6. Blocksize
7. Relative beginning track
8. Relative ending track
9. Secondary extents
10. Tracks allocated
11. Tracks used
12. Tracks unused
13. Data set name

1.2 JCL statements for executing SLIST are:

```
//    EXEC   PGM=SLIST,PARM=xxxx,REGIØN=30K
//SYSPRINT DD   SYSØUT=A
//DDn   DD   UNIT=DISK,DISP=ØLD,VØL=SER=volser
```

1.21 PARM=xxxx is an option which, if used, compares the four character PARM value with the first 4 characters of the DSNAME's on each volume. Only those DSNAME's resulting in an equal compare will be listed. If no PARM value is specified, the entire VTOC will be listed.

1.22 DDn - up to nine DD cards with DDNAME's DD1 through DD9 may be specified. A separate report will be generated for each volume specified. A volume may be identified through the catalog or pass queue by specifying a dsname.

2. NOTES ON SLIST OUTPUT

2.1 The quantities TRALLOC, TRUSED, and TRUNUSD are in tracks and include any secondary extents.

2.2 The quantities RBEGTR and RENDTR refer to the relative (to zero) beginning and ending tracks of the primary allocation on the volume.

2.3 Quantities for data sets which were allocated in terms of blocksize (not CYL or TRK) and which have secondary extents will be inaccurate.

2.4 Quantities for null "data sets" will be inaccurate.

2.5 Quantities for indexed sequential data sets are inaccurate.

3. CATALOGED PROCEDURE

3.1 A cataloged procedure LISTDISK, has been prepared to facilitate use of SLIST. A description of the procedure may be found in Division 3, Chapter 5, Cataloged Procedures.

SORTSPAC - CALCULATE DIRECT ACCESS SORT WORK SPACE

1. PURPOSE

1.1 SORTSPAC is a Western Electric developed program whose purpose is to perform the calculation to determine the amount of direct access work space required to perform a direct access sort given the parameters, logical record length, record count, number of auxiliary sort work areas and direct access device type.

2. DESCRIPTION

2.1 SORTSPAC performs its calculation on the basis of two formulas given in OS SORT/MERGE Programmer's Guide SC33-4007. The formulas are:

Formula 4, Balanced Technique
$T=(S(N)/K(N-1)) + 2N$

Formula 5, Crisscross Technique
$T=1.25S/K$

where:
T is total number of tracks
N is the number of intermediate storage areas
S is the number of input records
K=B/L
B is 7000 for 2314 or 12000 for 3330
L is input logical record length (maximum for variable length records)

2.2 The user supplies input consisting of fixed format cards with four parameters, logical record length, record count, number of work areas and device type, 3330 or 2314.

2.3 The program output is a SYSOUT display of the input cards and the computed number of tracks per work area and the number of cylinders per work area for each set of parameters.

2.5 If an input card is in error diagnostic messages are produced.

3. CODING REQUIREMENTS

3.1 The JCL required to execute SORTSPAC is:

3.11
```
//        EXEC   PGM=SØRTSPAC,REGIØN=30K
//SYSØUT   DD   SYSØUT=A
//PARMS    DD   *
  one or more input cards
/*
```

SORTSPAC - CALCULATE DIRECT ACCESS SORT WORK SPACE (CONT.)

3.2 The control card is of the following format:

Field Name	Columns	Length	Default
Logical record length	1 - 5	5	
Record count	7 - 14	8	
Number of work areas	16 - 17	2	6
Direct Access Device type	19 - 22	4	3330

3.21 Fields require leading zeros.

3.22 If work areas and device type are omitted the defaults are 06 and 3330.

4. EXAMPLE OF USE

```
//        EXEC  PGM=SØRTSPAC,REGIØN=30K
//SYSØUT  DD  SYSØUT=A
//PARMS   DD  *
00050,00025000,04,3330
```

Output from the above example appeared as:

```
SORTSPAC PROGRAM CALCULATIONS BASED ON:
00050,00025000,04,3330
LRECL IS    50, COUNT IS   25000, NO. OF WK AREAS IS  4, DASD IS 3330
USE    37 TRACKS PER WORK AREA OR    2 CYLS PER WORK AREA.
```

5. CATALOGED PROCEDURE

A cataloged procedure, SORTSPAC, has been prepared to facilitate use of this program. A description of the procedure may be found in Division 3, Chapter 5, Cataloged Procedures.

WEABEND - FORCE ABNORMAL TERMINATION

1. GENERAL

1.1 WEABEND is a Western Electric utility program which may be used to force abnormal termination of a job. The program is generally used in conjunction with a condition code test. Execution of WEABEND results in a user completion code of 0012.

2. EXAMPLE OF USE

```
//STEP1  EXEC  PGM=PRØGA
     .
     .
//STEP2  EXEC  PGM=WEABEND,CØND=(5,GT,STEP1),REGIØN=30K
//STEP3  EXEC  PGM=PRØGB
     .
     .
/*
```

In the above example, the job will abnormally terminate if PROGA returns a condition code greater than 4. If it does not, STEP2, the ABEND step, will be bypassed and the job will continue to STEP3.

WECOPY - GENERAL MEDIA CONVERSION

1. GENERAL

1.1 WECOPY is a Western Electric utility program which may be used for general media conversion. The program can:

1. Copy a sequential data set
2. Copy a member of a partitioned data set (PDS)
3. Change blocksize
4. Selectively copy records on a sequential data set
5. Reduce logical record length

1.2 JCL statements for executing WECOPY are:

```
//   EXEC  PGM=WECØPY,PARM=(max,skip),REGIØN=30K
//SYSPRINT  DD  SYSØUT=A
//SYSUT1    DD  parameters defining input data set
//SYSUTxxx  DD  parameters defining output data set (see 1.22 and 1.23)
```

1.21 PARM=(max,skip) is an optional field which allows the user to selectively copy input records. 'Max' is the maximum number of records to be copied from the input file and 'skip' is the number of initial records in the input file which are to be skipped before beginning the copy operation. The parameters must be numeric and may be of any length up to 9 digits. An invalid parameter will cause the PARM field to be ignored and the entire file will be copied.

1.22 DDNAME assignment - The output file for WECOPY is a DD statement with a DDNAME in the form of SYSUTxxx (other than SYSUT1 which always defines the input data set), where xxx is from one to three characters assigned by the user. The facility to specify a different DDNAME for the output may be useful under ASP. When WECOPY is executed several times within a job ASP //*FORMAT cards can refer uniquely to each output data set.

If WECOPY is used to obtain punched card output, the user must specify a DD statement with a DDNAME in the form of SYSUTPxx and SYSOUT=P. (For example, //SYSUTP1 DD SYSØUT=P). DCB parameters for punched card output are given in Paragraph 1.23.

1.23 DCB parameters - If any of the RECFM, LRECL, or BLKSIZE parameters are omitted from the DD statement defining the output data set (DDNAME=SYSUTxxx), the corresponding parameters will be supplied from the SYSUT1 DD statement. If punched card output is specified (DDNAME=SYSUTPxx and SYSØUT=P), the effective DCB parameters must be: RECFM=F,LRECL=80,BLKSIZE=80.

```
//CR       EXEC  WECOPY

//SYSUT1   DD *
//SYSUT2   DD  DSN=&&CR1,UNIT=DISK,
               DISP=(NEW,PASS),SPACE=(CYL,(1,1),RLSE)
//
```

WECOPY - GENERAL MEDIA CONVERSION (CONT.)

2. EXAMPLES OF USE

Example 1: `// EXEC PGM=WECØPY,PARM=(100,25),REGIØN=30K`
Result: The first 25 records will be skipped. Copying will begin with the 26th record and a maximum of 100 records will be copied.

The two parameters may also be coded separately:

Example 2: `// EXEC PGM=WECØPY,PARM=100,REGIØN=30K`
Result: Copy a maximum of 100 records beginning with the first input record.

Example 3: `// EXEC PGM=WECØPY,PARM=(,50),REGIØN=30K`
Result: The first 50 records will be skipped. Copying will begin with the 51st record and the remainder of the file will be copied.

3. WECOPY OUTPUT MESSAGE

3.1 After successfully copying the input data set, WECOPY will format a message indicating the number of records copied. For example:

```
'1500 RECORDS - WECOPY*JOBNAME STEPNAME'
```

This message will be written on the SYSPRINT file if a SYSPRINT DD statement is included in the job step. Otherwise, the message is directed to the system console and job's SYSMSG data set.

3.2 If an uncorrectable I/O error occurs, a message similar to the above message will be printed indicating the number of records, if any, successfully copied before the error occurred. In addition, a system generated message describing the nature of the error will be printed, and the program terminates (see Paragraph 5).

4. SYSOUT PROCESSING

All SYSOUT from WECOPY conform to ASP standards. There is no restriction on output LRECL or BLKSIZE. If BLKSIZE is not specified for output, 1024 will be assumed. If carriage control is not indicated on either input or output, WECOPY will insert ASA control characters for single space and page ejection.

5. CONDITION CODE

The condition code is set as follows:

00 - Successful completion
04 - Uncorrectable I/O error associated with SYSUT1
08 - Uncorrectable I/O error associated with SYSUTxxx

WECOPY - GENERAL MEDIA CONVERSION (CONT.)

6. I/O ERROR PROCESSING

When an I/O error occurs, WECOPY produces a descriptive error message, and immediately terminates normally without an abend. Through judicious use of the EROPT field of the DCB, WECOPY can be made to do the following:

a. Abend for input or output errors.
b. Copy only those input records which are error free.
c. Unconditionally copy a file, despite any I/O errors.

An error message will be generated for each I/O error that occurs.

7. CATALOGED PROCEDURE

A cataloged procedure, WECOPY, has been prepared to facilitate use of the WECOPY program. A description of the procedure and sample JCL for executing the procedure are described in Division 3, Chapter 5, Cataloged Procedures.

WECOPY2 - GENERAL MEDIA CONVERSION (MODIFIED VERSION OF WECOPY)

1. GENERAL

1.1 WECOPY2, a modified version of the WECOPY Utility Program, may be used to remove a group of records from the interior of a sequential data set. This is accomplished by the addition of a third parameter in the PARM field which specifies the number of records to be copied before deleting records.

1.2 The format of the PARM field is:

```
//    EXEC   PGM=WECØPY2,PARM=(max,skip,init),REGIØN=30K
```

where "max" is the maximum number of records to be copied after deleting records, "skip" is the number of records to be deleted, and "init" is the number of initial records to be copied prior to the delete phase.

1.3 JCL statements for executing WECOPY2 are the same as for WECOPY. If the third parameter of the PARM value is omitted, WECOPY2 functions exactly as WECOPY.

2. EXAMPLE OF USE

Example 1: `//   EXEC   PGM=WECØPY2,PARM=(40,15,10),REGIØN=30K`
Result: The first 10 records are copied, the next 15 records are skipped, and the next 40 records are copied.

Example 2: `//   EXEC   PGM=WECØPY2,PARM=(,15,10),REGIØN =30K`
Result: The first 10 records are copied, the next 15 records are skipped, and the remainder of the file is copied.

3. REFERENCE

For additional information on using WECOPY2, refer to the appendix that describes WECOPY.

WESELECT - SELECT SPECIFIED RECORDS

1. GENERAL

1.1 WESELECT is a Western Electric utility program which permits the user to select specified records from a data set. The program may be used to:

1. Extract desired record types
2. Divide a data set

2. CODING REQUIREMENTS

2.1 WESELECT functions are controlled by parameters in the PARM field of the EXEC statement:

PARM='displ,field,[STOP/STRT]'

2.11 displ is the displacement into the record. Note that the first byte of a record has a displacement of 0 (zero). The maximum value is 255, which is the displacement of the 256th byte of data. If the record is variable length, the 4-byte record length field must be included in the displ value.

2.12 field is a value used for comparison against input records. The value must be unique to the records desired. The field length is limited to 90 bytes and must not contain internal commas. If a hexidecimal value is desired, use the equivalent multi-punch code.

2.13 [STOP/STRT] - an optional parameter used only to divide a data set. STOP designates that spooling to SYSUT2 is to stop with this record. When the first record containing the specified field is found, that record becomes the last record on SYSUT2. STRT designates that spooling to SYSUT3 is to begin with this record. When the first record containing the specified field is found, that record becomes the first record on SYSUT3.

2.2 JCL statements for executing WESELECT are:

```
//     JOB CARD
//STEP1  EXEC PGM=WESELECT,PARM='displ,field,[STØP/STRT]',REGIØN=30K
//SYSUT1  DD  Parameters defining input data set
//SYSUT2  DD  1st Output (non-selected records) or ( spooled data as
//SYSUT3  DD  2nd Output (selected records) or     ( specified through
//PRINTFIL DD  SYSØUT=A                            ( use of optional
//SYSUDUMP DD  SYSØUT A                            ( STOP/STRT  parameter
```

DCB's for SYSUT1 will be used for SYSUT2 and SYSUT3 unless SYSUT2 and/or SYSUT3 are expressly specified in the JCL statements. The user may specify DD DUMMY to bypass writing of unwanted records on SYSUT2 or SYSUT3.

WESELECT - SELECT SPECIFIED RECORDS (CONT.)

3. EXAMPLES OF USE

Example 1: User wants to select all records with sub-code 20. The sub-code field is in bytes 9 and 10 of a fixed length record.

PARM coding: PARM='8,20'

Result: SYSUT2 has all records except sub-code 20 records (if the user did not specify DD DUMMY).

SYSUT3 has all sub-code 20 records.

Example 2: User discovers that some records on his data set have a date of 7315A due to input preparation errors. The date field is in bytes 15 - 19 of a fixed length record.

PARM coding: PARM='14,7315A'

Result: SYSUT2 has all records with a date other than 7315A.
SYSUT3 has all records with 7315A in the date field.

Example 3: Due to an abend condition, the user knows records have been "mod" onto a data set and wants to remove these records before restarting the job. The user knows that the first record that was "mod" onto the original data set has a 16-byte area that would not be found in any of the original records. This field is in bytes 70-85 of a fixed length record and has the entry: HAWTH73186151111.

PARM coding: PARM='69,HAWTH73186151111,STRT'

Result: SYSUT2 has all records up to, but not including, the first record found with the given value.
SYSUT3 has the given record and all succeeding records.

Example 4: Same as example 3 except the record known is the last record on the original data set (not the first record that was "mod" onto).

PARM coding: PARM='69,HAWTH73186151111,STØP'

Result: SYSUT2 has all records up to, and including, the given record.
SYSUT3 has all the succeeding records that were on the input data set.

WESELECT - SELECT SPECIFIED RECORDS (CONT.)

Example 5: The user intends to run several tests on a program that will process all records having a G in position 10 of a variable length record. The input data set is quite large and includes record types other than G. The user wants to extract only the G-type records.

PARM coding: PARM='13,G' (Displacement=9 bytes + 4 length bytes).

Result: SYSUT2 has all records except G type records (if the user did not specify DD DUMMY).
SYSUT3 has all G type records.

WEWTOR - WRITE TO OPERATOR

1. GENERAL

1.1 WEWTOR is a Western Electric utility program which facilitates communication with the console operator on a job step basis. The program may be used in conjunction with COND parameters in the EXEC statement to initiate a console message under given conditions.

1.2 The message text to be displayed on the console is supplied as a PARM value in the EXEC statement and may be up to 100 characters in length (a maximum of 61 characters may be coded in a single card). The program will add the term "-REPLY U" to the displayed message text, which will require an operator reply.

2. JCL STATEMENTS FOR EXECUTING WEWTOR

```
//STEPXX  EXEC  PGM=WEWTØR,CØND=(5,GT,STEPX),REGIØN=30K,
// PARM='PSI-03 MESSAGE REQUIRING ACKNOWLEDGEMENT'
```

STEPXX will issue the following console message:

```
nn    PSI-03 MESSAGE REQUIRING ACKNOWLEDGEMENT-REPLY U
```

where nn is a system generated number used by the console operator when responding to a message.

3. RESTRICTIONS

Division 2, Chapter 2, Section 1 contains a Warrenville Data Center Standard governing the use of console messages initiated by a user program. It is recommended that the user become acquainted with this Standard prior to using the WEWTOR program.

WEDEBEPT - TAPE AND "DISK FILE" DUMP

1. GENERAL

1.1 WEDEBEPT is a Western Electric print utility whose purpose is to print a sequential file whose characteristics may not be known. It prints in increments of 100 bytes and terminates each physical record when the inter-block gap (IBG) is detected. The first print line of each block is indicated with the block record count as well as the block size. Print output is routed to the system output stream (SYSOUT). This may be used to print standard tape labels.

1.2 WEDEBEPT incorporates the following facilities:

1.21 For tape and disk files process an input block of up to 8000 bytes. The user may specify the following:

1. Starting block number.
2. Number of blocks to be printed.
3. Printout representation in Hexadecimal or EBCDIC characters.

1.22 Multiple data sets on tape may be processed with the following user options:

1. Print all of the files
2. Select which file is to be printed
3. Select the starting file number

1.3 The following default parameters have been incorporated into the program:

1. Start printing at file number one
2. Begin printing block number one
3. One file is to be printed
4. Print 25 blocks and terminate job
5. Printing is in character representation

Any modification of these parameters will require a 19 byte PARM value in the EXEC statement (see PARM FORMAT, 2.2).

2. PARM FORMAT

2.1 Before WEDEBEPT is executed, a check is made for a PARM modification. If none exists, the default options are in effect (see paragraph 1.3). A PARM change must contain 19 characters as defined in paragraph 2.2 or else the program will not process. Notification of job failure is directed to the system console device and the reason for termination is directed to SYSOUT.

WEDEBEPT - TAPE AND "DISK FILE" DUMP (CONT.)

2.2 The format of the PARM value is as follows:

PARM='ff,bbbbb,dd,nnnnn,r' where:

2.21 ff is the starting file number. A value of 01 through 99 is valid for tape, a value of 01 is valid for disk.

2.22 bbbbb is the starting block number. A value of 00001 through 99999 is valid.

2.23 dd is the number of data sets to be printed from this volume. A value of 01 through 99 is valid for tape, 01 is valid for disk.

2.24 nnnnn is the number of blocks to be printed. A value of 00001 through 99999 is valid as is "ALLPT", which will print the entire file. (This format, ALLPT, is not recommended).

2.25 r designates printout representation with "C" specifying character and "H" specifying Hexadecimal.

Note: The 3rd, 9th, 12th, and 18th positions of the PARM value are commas separating the parameter values stated above.

3. JCL STATEMENTS

3.1 In general the following JCL is required

```
// EXEC  PGM=WEDEBEPT,REGIØN=30K
//SYSUT1 DD parameters describing input
//SYSUT2 DD SYSØUT=A
```

3.2 For tape, the SYSUT1 dd statement should include:

```
//SYSUT1 DD UNIT=TAPE,VØL=SER=SSSSSS,DSN=dsname
```

(Continued on next page)

Reason for reissue: Reference to ASP SETUP card deleted.

WEDEBEPT - TAPE AND "DISK FILE" DUMP (CONT.)

4. EXAMPLES OF USE

Example 1: Print the VOL,HDR1, and HDR2 information from a standard label tape or use the defaults to print an unlabeled tape.

```
// EXEC PGM=WEDEBEPT,REGIØN=30K
//SYSUT1 DD UNIT=TAPE,VØL=SER=123456,DSN=MYTAPE,LABEL=(,BLP),
// DISP=(ØLD,KEEP)
//SYSUT2 DD SYSØUT=A
/*
```

Example 2: Print a disk file passed from a previous step in hexadecimal.

```
// EXEC PGM=WEDEBEPT,REGIØN=30K,PARM='01,00001,01,ALLPT,H'
//SYSUT1 DD DSN=&&TEMPFILE,DISP=(ØLD,DELETE)
//SYSUT2 DD SYSØUT=A
```

Example 3: Print standard label information and 10 blocks of input data in character representation.

```
// EXEC PGM=WEDEBEPT,PARM='01,00001,02,00010,C'
```

SYSUT1 and SYSUT2, see Example 1.

Example 4: Print an unlabeled tape. Start at block 500 and print 20 blocks of input data in Hexadecimal representation.

```
// EXEC PGM=WEDEBEPT,PARM='01,00500,01,00020,H'
```

SYSUT1 and SYSUT2, see Example 1.

Example 5: A multiple data set has four files. Print 15 blocks from files three and four in character representation.

JCL Coding:
```
// EXEC PGM=WEDEBEPT,PARM='03,00001,02,00015,C'
```

SYSUT1 and SYSUT2, See Example 1.

5. REFERENCE

See catalogued procedures DADUMP and TPDUMP in Division 3, Chapter 5, Section 1.

Reason for reissue: Example 1 changed to omit reference to ASP SETUP card.

WETESTPR - ASP RJP WORKSTATION PRINTER TESTING

1. GENERAL

WETESTPR is a Western Electric program that allows an ASP RJP Workstation to test printers on-line. The remote Workstation is assumed to have an IBM QN print chain or equivalent (60-GRAPHICS, 45 Preferred) with 132 print positions.

2. TEST PATTERN

2.1 TEST 1 - Print each of the 60 character set 10 times, 60 lines per page.

2.2 TEST 2 - A ripple print effect, the initial print line repeats the 60 character set 2 times, and the first 12 characters for a 132 character print line. Each subsequent print line is offset 1 character for a total of 60 iterations per page. Twenty pages are printed before the test is terminated.

3. JCL REQUIREMENTS

```
//  JOB CARD
//*FØRMAT PR,DDNAME=PRINTER,DEST=RM00XPRY
//  EXEC PGM=WETESTPR,REGIØN=30K
//PRINTER DD SYSØUT=A
```

where X represents terminal number, Y represents the specific printer to be tested.

Multiple printers can be tested by submitting additional ASP //*FORMAT cards, specifying in the DEST=parameter the applicable device.

4. TEST FUNCTIONS

4.1 Verifies QN UCS buffer is still properly loaded.

4.2 Synchronization of print line can be observed. (wavy printing)

4.3 Dropped characters or worn print positions can be noted.

5. REFERENCE

Questions may be directed to D. J. Wilfenger, 9411 Warrenville.

VFPRINT - SELECTIVE PRINT DUMP

1. GENERAL

1.1 VFPRINT is a print utility program, useful as a debugging aid because of its ability to print records in two formats and to select records for printing on the basis of several options. The formats are hexadecimal or normal character representation or an alternating combination of the two. The select options include starting point and record count controls or field comparisons.

1.2 Input to VFPRINT may be tape or disk, fixed, fixed blocked, variable or variable blocked, with a maximum LRECL of 2800 bytes.

1.3 Output of VFPRINT is printed in lines of 120 characters for as many lines as necessary to list the entire physical record. The first printed line of each record contains the physical record number in positions 122-132. A line is skipped between the printout of each physical record.

2. CONING REQUIREMENTS

2.1 VFPRINT is controlled by parameters of the PARM field and/or parameters of a control statement.

2.2 JCL statements required are:

```
// EXEC PGM=VFPRINT,PARM=[options],REGION=30K
//SYSPRINT DD SYSØUT=A
//SYSUT1 DD Parameters defining input data set
//SYSUT2 DD SYSOUT=A
//SYSIN DD * (optional)
    VFPRINT control statement  (optional)
```

2.3 If PARM is omitted from the EXEC statement the entire input file is printed in character format.

2.4 The PARM may be used with positional parameters to give the starting point, count controls, and print format.

2.41 The first positional parameter gives the number of records to be printed.

2.42 The second positional parameter gives the number of records to be skipped before printing begins.

2.43 The third positional parameter specifies the format of the printed output. If it is omitted the format is character representation. If it is HX, the output is printed in hexadecimal representation. If it is CX, the output is printed alternately in hexadecimal and character representation.

2.5 If PARM=SYSIN is specified, the user must choose from six parameters to establish a mode of record selection. These keyword parameters are entered in a control statement.

VFPRINT - SELECTIVE PRINT DUMP (CONT.)

2.501 LN - A positional parameter designating the length of the compare field. This must be the first parameter.

2.502 SL - Starting location of the compare field in the input record.

2.503 EQ - The value which will be equated to the compare field. Input records whose compare fields are equal to this value will be printed. When EQ is designated, parameters GT and LT may not be used.

2.504 GT - A value to designate the lower limit of a range. Input records whose compare field is greater than this value will be printed. If GT is used, LT must be used also.

2.505 LT - A value to designate the upper limit of a range. Input records whose compare field is less than this value will be printed. If LT is used, GT must be used also.

2.506 FM - A value to designate print format. If it is omitted the format will be character. A value of HX will cause hexadecimal representation. A value of CX will cause alternating hexadecimal and character representation.

2.507 ALL - This signifies that the search governed by EQ, LT and GT will be performed on the entire file. If ALL is not specified, the search assumes the records are in ascending sequence on the comparison field and may terminate without searching the entire file. (i.e. The first unequal after an equal comparison will terminate the search.)

2.508 Keyword parameters except LN, may be entered in any order and starting in any column.

2.509 A control card may be continued in another card by placing a comma after the last parameter on a card, however, a parameter and its value may not be split over two cards.

2.510 Values supplied for EQ, GT or LT may be any string of characters equal to the length defined by the LN parameter.

3. EXAMPLES OF USE

Example 1: Print the first 100 records in character format.

```
// EXEC PGM=VFPRINT,PARM=100
```

Example 2: Skip 50 records and print the next 100 in character format.

```
// EXEC PGM=VFPRINT,PARM=(100,50)
```

Example 3: Print an entire file in hexadecimal.

```
// EXEC PGM=VFPRINT,PARM=(,,HX)
```

VFPRINT - SELECTIVE PRINT DUMP (CONT.)

Example 4: Print 50 records of a file in alternating hexadecimal and character format.

```
// EXEC PGM=VFPRINT,PARM=(50,,CX)
```

Example 5: Print all records with a value of 12345 starting in position 7.

```
// EXEC PGM=VFPRINT,PARM=SYSIN
LN=5,SL=7,EQ=12345
```

Example 6: Print, in hexadecimal, all records with a field starting in position 25 having values ranging from 575 to 1250.

```
// EXEC PGM=VFPRINT,PARM=SYSIN
LN=4,SL=25,GT=0574,LT=1251,FM=HX,ALL
```

4. REFERENCE

W. J. Zatloukal, programmer, Warrenville.

LIBRLIST - PARTITIONED DATA SET TO LIBRARIAN DATA SET

1. GENERAL

1.1 The LIBRLIST utility is a Western Electric utility program which is used to set up data transfer from a source partitioned data set to a LIBRARIAN data set. With one execution of LIBRLIST, an entire IBM-type library could become part of a separate LIBRARIAN library.

1.2 The LIBRLIST utility produces input for a subsequent LIBRARIAN execution step. The input produced by LIBRLIST is in the standard format required by LIBRARIAN and is as follows:

```
-ØPT  NØEXEC,INDEX
-ADD  module name,(contents of parmfield, if any)
-AUX  SOURCE(module name)
-EMØD
```

1.21 The last three cards are repeated for each module on the input source library.

1.22 The last statement produced by LIBRLIST will be an -END card.

1.23 LIBRLIST will produce LIBRARIAN input for modules with the same names as the corresponding modules on the input IBM data set.

1.24 The LIBRARIAN data set must already exist and be initialized prior to using LIBRLIST.

2. CODING REQUIREMENTS

2.1 The JCL required to execute LIBRLIST is:

```
//STEP EXEC PGM=LIBRLIST,PARM='parameters defining what will appear on
  the -ADD card after the module name' (see paragraph 2.11)
//SYSUT1 DD parameters defining input data set (see paragraph 2.12)
//SYSPRINT DD parameters defining output data set (see paragraph 2.13)
```

2.11 The PARM field is optional. If omitted, default parameters that apply when adding modules to a LIBRARIAN data set will be in effect when the LIBRARIAN step is executed. When specified, it will provide the options that will be placed on all -ADD cards produced by LIBRLIST. For example, PARM=',SEQ=/1,6,100,100/' would provide this type of sequencing for all the modules added. All valid options for the -ADD card may be included on the PARM field.

Note the "comma" used in the example. This must always appear as the first character in the PARM field after the quotes because LIBRLIST will take whatever is in the PARM field and place it immediately after the module name on the -ADD cards.

2.12 SYSUT1 is used to describe the parameters used as input to the LIBRLIST utility. It must describe an IBM partitioned data set whose members are to become modules in a LIBRARIAN data set.

2.13 SYSPRINT is used to describe the output of LIBRLIST. This should specify a temporary data set, which must be blocked 80, which will contain the LIBRARIAN control cards produced by the LIBRLIST utility. This data set will be the SYSIN input to a subsequent LIBRARIAN step.

2.2 LIBRLIST requires 30K for execution.

2.3 The LIBRARIAN step following the execution of the LIBRLIST utility must contain a DD statement with the ddname SOURCE and the dsname of the IBM-type partitioned data set specified in SYSUT1 of the LIBRLIST utility. The SOURCE dd card would have to describe the IBM-type partitioned data set fully enough in the JCL to pick up the necessary information.

3. EXAMPLES OF USE

```
//STEP1 EXEC PGM=LIBRLIST,PARM=',LIST,SEQ=CØBØL,SYNCHK(CA,C)',REGIØN=90K
//SYSUT1 DD DSN=WN.PGMS,DISP=SHR
//SYSPRINT DD DSN=&&PGMS,UNIT=DISK,DISP=(NEW,PASS,DELETE),
//     DCB=(BLKSIZE=80),SPACE=(TRK,(2))
/*
//STEP2 EXEC LIBP,MASTER='PRD6.SØURCE',REGIØN=90K
//LIBP.SØURCE DD DSN=WN.PGMS,DISP=SHR
//LIBP.SYSIN DD DSN=&&PGMS,DISP=(ØLD,DELETE)
```

These two steps will transfer all the source modules from WN.PGMS and add them to the LIBRARIAN data set PRD6.SOURCE with options as specified on the PARM card.

4. REFERENCES

LIBRARIAN User Reference Manual
LIBRARIAN procedures - Warrenville S & R Manual, Div. 3, Chap. 5, Sect. 1
ERC Technical Report CC005292 (12-13-72) "LIBRLIST,"
 Author - E. Wasserman, 222 Broadway

DASPACE - DIRECT ACCESS SPACE UTILIZATION WITH VARIED BLOCKING FACTORS - 3330

1. PURPOSE

1.1 DASPACE is a Western Electric developed program whose purpose is to compute direct access space utilization data for fixed length records using a variety of blocking factors from 1 to the maximum allowed by track size. The programmer may review the generated data and select a factor which will best serve his needs and yet make efficient use of the direct access space allocated. If a record estimate is provided the program also computes the tracks, cylinders or blocks required for the file.

2. DESCRIPTION

2.1 DASPACE performs its calculations on the basis of the formula provided in the IBM Reference Manual for IBM 3330 Disk Storage GA-1592.

$$N = 13165 / (135 + C + KL + DL)$$

where:

N is the number of physical records per track
KL is key length
C is 0 if KL is zero
C is 56 if KL is not zero
DL is data length (physical block)

2.2 Input consists of fixed format control cards with three parameters, logical record length, estimated record count, and key length (if keys are used).

2.3 The programmer may specify a complete or condensed form of the listing by specifying a parm value of 'L' for the complete listing or omitting the parm for the condensed version. The condensed version lists only the block sizes which give maximum number of records per block for each integral number of blocks per track.

2.4 The program output consists of a report under the DDNAME of "USAGE" and error messages under the name "SYSOUT".

2.5 The maximum block size permitted for SYSOUT under ASP is printed and identified.

3. CODING REQUIREMENTS

3.1 The JCL required is as follows:

3.11
```
//  EXEC  PGM=DASPACE,REGIØN=30K,PARM=x
//SYSØUT  DD  SYSØUT=A
//USAGE  DD  SYSØUT=A,DCB=BLKSIZE=121
//PARMS  DD  *
    one or more input cards
/*
```

The DCB sub-parameter BLKSIZE is required for the USAGE statement. It should be a multiple of 121.

DASPACE - DIRECT ACCESS SPACE UTILIZATION WITH VARIED BLOCKING FACTORS - 3330 (CONT.)

3. CODING REQUIREMENTS (CONT.)

3.12 PARM=L will cause a full listing. If omitted a condensed listing is generated.

3.2 The control card is of the following format:

Field Name	Columns	Length	Default
Logical record length	1-5	5	none
Estimated record count	7-14	8	zero
Key length	16-18	3	zero

Fields require leading zeros.

4. EXAMPLE OF USE

```
//   EXEC   PGM=DASPACE,REGIØN=30K
//SYSØUT  DD   SYSØUT=A
//USAGE  DD  SYSØUT=A,DCB=BLKSIZE=1936
//PARMS  DD  *
00100
00080,00025000
00075,00000000,015
```

5. CATALOGUED PROCEDURE

A catalogued procedure, DASPACE, has been prepared to facilitate use of this program. A description of the procedure may be found in Division 3, Chapter 5, Catalogued Procedures.

6. REFERENCE

E. W. Rzym, programmer, Warrenville.

(Continued)

DASPACE - DIRECT ACCESS SPACE UTILIZATION WITH VARIED BLOCKING FACTORS - 3330 (CONT.)

7. SAMPLE OUTPUT

```
                    DIRECT ACCESS SPACE GUIDE FUR 3330

EST NO  LRECL  KEY  RCDS   BLK    RCDS  BLKS  BYTES  % USER  TRKS  CYLS    BLKS
 RCDS               /BLK   SIZE   /TRK  /TRK  DATA    DATA   REQD  REQD    REQD

  9500    75         01      75     62    62   4650   35.3    154     9    9500
  9500    75         02     150     92    46   6900   52.4    104     6    4750
  9500    75         03     225    108    36   8100   61.5     88     5    3167
  9500    75         04     300    120    30   9000   68.3     80     5    2375
  9500    75         05     375    125    25   9375   71.2     76     4    1900
  9500    75         06     450    132    22   9900   75.1     72     4    1584
  9500    75         07     525    133    19   9975   75.7     72     4    1358
  9500    75         08     600    136    17  10200   77.4     70     4    1188
  9500    75         09     675    144    16  10800   82.0     66     4    1056
  9500    75         10     750    140    14  10500   79.7     68     4     950
  9500    75         11     825    143    13  10725   81.4     67     4     864
  9500    75         12     900    144    12  10800   82.0     66     4     792
  9500    75         14    1050    154    11  11550   87.7     62     4     679
  9500    75         15    1125    150    10  11250   85.4     64     4     634
  9500    75         17    1275    153     9  11475   87.1     63     4     559
  9500    75         20    1500    160     8  12000   91.1     60     4     475
  9500    75         23    1725    161     7  12075   91.7     59     4     414
  9500    75         27    2025    162     6  12150   92.2     59     4     352
  9500    75         33    2475    165     5  12375   93.9     58     4     288
  9500    75         42    3150    168     4  12600   95.7     57     3     227
  9500    75         56    4200    168     3  12600   95.7     57     3     170
  9500    75         85    6375    170     2  12750   96.8     56     3     112
  9500    75        173   12975    173     1  12975   98.5     55     3      55
```

```
                    DIRECT ACCESS SPACE GUIDE FOR 3330

EST NO  LRECL  KEY  RCDS   BLK    RCDS  BLKS  BYTES  % USER  TRKS  CYLS    BLKS
 RCDS               /BLK   SIZE   /TRK  /TRK  DATA    DATA   REQD  REQD    REQD

          120        01     120     51    51   6120   46.4
          120        02     240     70    35   8400   63.8
          120        03     360     78    26   9360   71.0
          120        04     480     84    21  10080   76.5
          120        05     600     85    17  10200   77.4
          120        06     720     90    15  10800   82.0
          120        07     840     91    13  10920   82.9
          120        08     960     96    12  11520   87.5
          120        09    1080     90    10  10800   82.0
          120        11    1320     99     9  11880   90.2
          120        12    1440     96     8  11520   87.5
          120        14    1680     98     7  11760   89.3
          120        16    1920     96     6  11520   87.5                     ASP SYSOUT LIMIT
          120        17    2040    102     6  12240   92.9
          120        20    2400    100     5  12000   91.1
          120        26    3120    104     4  12480   94.7
          120        35    4200    105     3  12600   95.7
          120        53    6360    106     2  12720   96.6
          120       108   12960    108     1  12960   98.4
```

DPMPSIL - LIST THE CONTENTS OF THE PSI TABLE

1. GENERAL

1.1 DPMPSIL is a Western Electric utility program which may be used to produce a formatted listing of the contents of the PSI table by organization number. A parameter may be specified to limit the list to one location or to one organization within a location.

1.2 Possible formats for a user supplied parameter are:

PARM=LL where LL is an alphabetic location code. The PSI list produced will include only PSI's registered for this location.

PARM=LLDDDD where LL is an alphabetic location code and DDDD is a department number (numeric). The PSI list produced will include only PSI's registered for this organization.

If no parameter is specified, the entire PSI table will be listed.

2. EXECUTION

2.1 Execution of DPMPSIL should be accomplished via the cataloged procedure DPMPSIL which is documented in Division 3, Chapter 5, Cataloged Procedures.

WEMFCOPY - MULTIPLE FILE COPY

1. GENERAL

1.1 WEMFCOPY is a Western Electric utility program for creating "DUMMY" files and/or for copying multiple files in one program step.

1.2 JCL statements for executing WEMFCOPY are:

```
//  EXEC PGM=WEMFCOPY,REGION=60K
//STEPLIB  DD DSN=DEV08.LOADLIB,DISP=SHR (see par. 1.26)
//SYSPRINT DD SYSOUT=A
//Dxxxxxxx DD parameter defining "DUMMY" files
                  .
                  .
//Fyyyyyyy DD parameters of input file to be copied
                  .
                  .
//Tyyyyyyy DD parameters of copied (output) file
                  .
                  .
```

1.21 Dxxxxxxx represents a ddname which starts with "D". Any "D" prefixed name represents a "DUMMY" file (one containing O.S. header and trailer labels only). Up to 10 dummy files can be created. Such files are processed in their respective JCL sequence and before any copy operations.

1.22 Fyyyyyyy represents a ddname which starts with "F" and denotes an input sequential data set which is to be copied.

1.23 Tyyyyyyy represents a ddname which starts with "T" and denotes an output sequential data set.

1.24 In order to copy a file, the ddname defining the input data set must be prefixed with "F" and the ddname defining the output data set must be prefixed with "T". The other seven characters in the respective "F" and "T" prefixed ddnames must be equal.

1.25 Up to ten (10) files can be copied, in addition to "DUMMY" file processing.

WEMFCOPY - MULTIPLE FILE COPY (CONT.)

1.26 A DCB, specifying LRECL, BLKSIZE, and RECFM, must be specified for each input file to be copied. If undefined record length (RECFM=U) is specified, the user must include a STEPLIB ddname card as shown in par. 1.2. If DCB information is missing on an output DD, then the DCB supplied for the respective input will be utilized.

1.27 Files are copied according to the sequence of the "F" prefixed ddnames. Only one file is copied at a time. UNIT=AFF can be utilized to share a common device (s) on output.

2. WEMFCOPY REPORT

2.1 After successfully copying data files, WEMFCOPY will issue a report containing information about each copy operation. The report contains the following :

a. TIME - copy operation complete
b. FILENAME - last seven characters of like "F" and "T" ddnames
c. RECCNT - record count
d. BLKCNT - block count
e. RF - RECFM
f. REC-LEN - LRECL of output
g. BLK-LEN - BLKSIZE of output
h. VOL-SER - output volume serial
i. PER-CNT-FUL - percent of a 2000 foot reel of tape that the file would occupy
j. DSNAME - DSNAME of the output data set
k. RECIN - LRECL of input file if unit record is used as output
l. BLK-IN - BLKSIZE of input file if unit record is used as output
m. EXPIRE - retention date.

3. REFERENCE

Oklahoma City Software Support Group

WESFSPRT - SEQUENTIAL FILE SELECTIVE PRINT

1. GENERAL

1.1 WESFSPRT is a general purpose print program to print sequential files or portions of sequential files in either HEX or normal format.

1.2 Input to WESFSPRT is sequential tape or disk files in fixed, fixed blocked, variable, and variable blocked format.

1.3 Output of WESFSPRT is printed 100 characters/line until each physical record is printed. A line is skipped between each physical record with a heading line at the top and bottom of each page.

2. CODING REQUIREMENTS

2.1 WESFSPRT is controlled by a PARM field and a control card for special conditions.

2.2 JCL statements are:

```
//  EXEC PGM=WESFSPRT,PARM='(options)',REGION=30K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD (user data set)
//SYSIN DD *
 Control card (Optional - see par. 2.46)
/*
```

2.3 If PARM is omitted from the EXEC statement, the entire input file is printed in normal format.

2.4 PARM options.

2.41 SKIP=n skip "n" records before starting to print.

2.42 PICK=n print every "n" record.

2.43 LIST=n print only "n" records.

2.44 HEX print data in HEX format (unpacked data prints in normal format).

2.45 NXP do not space between physical records.

WESFSPRT - SEQUENTIAL FILE SELECTIVE PRINT (CONT.)

2.4 PARM Options (Cont.)

2.46 CARD to specify control card option.

Control card format:

```
*L,START,LENGTH,ARGUMENT
*H,START,LENGTH,ARGUMENT
*E,START,LENGTH,ARGUMENT
*AE,START,LENGTH,ARGUMENT
```

where:

*	Is used for control purposes
L	Low
H	High
E	Equal
AE	All Equal
START	Is the starting location of the data to be tested
LENGTH	Is the length of the data to be tested
ARGUMENT	Is the data to be tested

With the above options you can print all records with the control less than the Argument, greater than the Argument, or equal to the Argument. Also you can select all records that are equal to the Argument regardless of the sequence of the inputs.

Example: *L,1,5,08381
This would print out all records that have a control of less than 08381 starting in position 1 for a length of 5. The commas are required to separate the fields.

3. REFERENCE

Oklahoma City Software Support Group.

ZEROCORE - ZERO USER REGION IN MAIN STORAGE

1. GENERAL

1.1 When a job completes on an IBM 360/370 CPU the main storage assigned to that job is left intact as last used by the job. When that region in main storage is assigned to another job the core remains as previously stated. If the second job abends and the user has requested a dump he will obtain a dump of all the main storage assigned to him even though he has not necessarily used it, thus giving the user a dump of some of the core as left by the previous user. To prevent another job from obtaining information in this manner, ZEROCORE zeroes the main storage, I/O buffers and subpools and hence insures that confidential information may not be obtained via system dumps.

2. CODING REQUIREMENTS

2.1 To execute ZEROCORE one merely provides an EXEC card with PGM=ZERØCØRE and REGIØN=xxK where xx is region in thousands to be zeroed. This EXEC card would be the last step of a job if the region for all steps in the job is the same or if the class of the job is A,B,or C. If the associated job uses descending regions and a CLASS of J,K, or L, then to ensure that all main storage is zeroed in that particular job a ZEROCORE step must be inserted before the region is decreased.

2.2 To be able to zero core even if the job abends, a CØND=EVEN must be coded on the EXEC card.

3. VERIFICATION

3.1 The zeroing of core may be verified by use of PARM='VERIFY' on the EXEC card. This parameter will cause two 'DEBUG' dumps to be taken. One is after main storage is zeroed and the second is after the subpools and buffers are cleared. In this case a //VERIFIED DD SYSØUT=A card must be placed immediately after the EXEC card.

4. REFERENCE

R. A. Burns, Programmer, Lisle.

TRMUST06 - CHECK FOR ENTRIES IN A GENERATION DATA GROUP

1. GENERAL

1.1 TRMUST06 is a Western Electric utility program developed for application systems interfacing with Mini-WEDGE. The function of this program is to determine if there are any entries in the generation data group specified by the user via a PARM parameter in the JCL EXEC statement.

1.2 An application program can determine if an input file has been UNSTACKed by checking a condition code. The condition code (General Register 15) is set to zero (0) if relative generation zero (0) or other explicitly specified relative generation exists, otherwise it is set to 04 with one exception: if a PARM value is not supplied by the user, the condition code is set to 08.

1.3 The program does not make a validity check for a valid GDG DSNAME base (or index).

1.4 The minimum core requirement is 6K.

2. EXAMPLES OF USE

2.1 In the following examples, assume ABC.DEF to be the DSNAME base (or index) of a generation data group.

2.11
```
// EXEC PGM=TRMUST06,PARM='ABC.DEF'
```

2.12 An equivalent expression to the above example is:

```
// EXEC PGM=TRMUST06,PARM='ABC.DEF(0)'
```

2.13 The existence of old generations may also be tested

```
// EXEC PGM=TRMUST06,PARM='ABC.DEF(-2)'
```

The return code will be set to 04 unless at least three generations are cataloged.

3. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

CATALIST - CATALOG LISTING

1. GENERAL

1.1 CATALIST is a Western Electric utility program that will selectively list the contents of an OS catalog. This program provides more information than the IBM IEHLIST utility and prints the data in a more readable format.

2. PARM FORMAT

2.1 To list members of a Production or Development catalog, one must use the PARM field which has the following format:

```
PARM='cvoln1(node1,node2),cvoln2(node3,node4)'
```

2.11 cvol n - is the serial number of the control volume to be listed.

2.12 node n - is the node point to be listed.

2.2 If no PARM is specified, CATALIST will list the entire catalog on the system residence volume of the particular system on which the job is running.

3. JCL STATEMENTS

3.1 In general the following JCL is required:

```
//   EXEC   PGM=CATALIST,REGION=14K
//SYSPRINT   DD   SYSOUT=A
```

3.2 If the control volume is not permanently mounted, a DD statement should be added to mount the volume.

4. EXAMPLES OF USE:

4.1 Example 1: List the entire catalog on the system residence volume.

```
//   EXEC   PGM=CATALIST,REGION=14K
//SYSPRINT   DD   SYSOUT=A
```

4.2 Example 2: List the entire catalog on volume PROD06.

```
//   EXEC   PGM=CATALIST,REGION=14K,PARM=PROD06
//SYSPRINT   DD   SYSOUT=A
```

CATALIST - CATALOG LISTING (CONT.)

4.3 Example 3: List the catalog below node points DEV05.F00 and DEV05.F96 on volume TEST05.

```
//  EXEC  PGM=CATALIST,REGION=14K,
// PARM='TEST05(DEV05.F00,DEV05.F96)'
//SYSPRINT  DD  SYSOUT=A
```

4.4 Example 4: List the catalog below node points DEV05.F96 on volume TEST05 and PRDT6.PEA and PRDG6.REA on volume PROD04.

```
//  EXEC  PGM=CATALIST,REGION=14K,
//  PARM='TEST05(DEV05.F96),PROD04(PRDT6.PEA,PRDG6.REA)'
//SYSPRINT  DD  SYSOUT
```

5. REFERENCE

5.1 Cataloged procedure CATALIST in Division 3, Chapter 5, Section 1.

5.2 SOFTWARE-X-CHANGE (12-26-73) "CATALIST" Ref. #A-32
Author - E. Wasserman, Headquarters Data Center

5.3 Questions may be directed to J. R. Jackson, Dept. 9443, Columbus.

CATAUTIL - CATALOG MAINTENANCE UTILITY

1. GENERAL

1.1 CATAUTIL is a Western Electric utility whose purpose is to perform functions on the catalog that are not available through IEHPROGM.

1.2 CATAUTIL incorporates the following facilities:

1.21 Delete an index and all subordinate indexes and uncatalog any data sets that are subordinate to these indexes.

1.22 Change generation data group index specification without previously uncataloging all data sets and subsequently re-cataloging all data sets.

1.3 The return code will be the highest one encountered in the execution of the program.

2. CONTROL CARDS

2.1 The control card for deleting an index is as follows:

```
label    DLTINDEX    INDEX=name,CVOL=serial
```

where:

2.11 INDEX=name

Specifies the index level that is to be deleted and this includes deleting of all subordinate indexes and uncataloging all subordinate data sets.

2.12 CVOL=serial

Specifies the volume serial number of the control volume on which the search for the index is to begin. If CVOL is omitted the index should either reside on the system residence volume or on a volume that is appropriately connected to by the first level of the index.

CATAUTIL - CATALOG MAINTENANCE UTILITY (CONT.)

2.2 The control card for changing a Generation Data Group index specification is as follows:

```
label    CHNGBLDG  INDEX=name,ENTRIES=n,
                    OPTION=option,CVOL=serial
```

where:

2.21 INDEX=name

Specifies the name of the Generation Data Group index that is to be changed.

2.22 ENTRIES=n

Specifies the new number of entries for the Generation Data Group index which must not exceed 255.

2.23 OPTION=option (if omitted, the default is L)

Specifies one character option selection as follows:

OPTION	MEANING
L	Leave option as previously specified.
E	Empty option as stated for BLDG statement in the IBM utilities manual.
D	Delete option as stated for BLDG statement in the IBM utilities manual.
B	Both Empty and Delete
N	Neither Empty or Delete - just uncatalog

2.24 CVOL=serial

Specifies same as in paragraph 2.12.

CATAUTIL - CATALOG MAINTENANCE UTILITY (CONT.)

3. JCL STATEMENTS

3.1 In general the following JCL is required

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD * control cards follow
```

3.2 If the control volume is not a permanently resident volume then a DD statement should be placed in the JCL to insure the volume is mounted.

3.3 The basic module takes 24K to execute and requires additional 2K for each 30 data sets cataloged in a Generation Data Group that is being changed with a CHNGBLDG control card.

4. EXAMPLES OF USE

4.1 Example 1: There exists two Generation Data Groups below the NODE A.B both of which are no longer needed.

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DLTINDEX INDEX=A.B
```

4.2 Example 2: The number of entries for Generation Data Group index A.B.C is to be changed from 15 to 35 and the option is to be left as is.

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  CHNGBLDG INDEX=A.B.C,ENTRIES=35
```

4.3 Example 3: The option of delete is to be removed for Generation Data Group index A.B.D on control volume 222222.

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  CHNGBLDG INDEX=A.B.D,OPTION=N,
  CVOL=222222
```

CATAUTIL - CATALOG MAINTENANCE UTILITY (CONT.)

5. RESTRICTIONS

5.1 CATAUTIL may not be used for a generation data set which contains an entry which resides on more than five reels.

6. REFERENCE

6.1 Description of IEHPROGM in I.B.M. Operation System Utilities Manual (GC28-6586).

6.2 Cataloged procedure CATAUTIL in Division 3, Chapter 5, Section 1.

6.3 Questions may be directed to J. R. Jackson, Dept. 9443, Columbus.

DATA PROCESSING SYSTEM PROGRAMMING AIDS

UTILITY PROGRAMS

SECTION 3 - COMMERCIAL SOFTWARE PRODUCTS

1. INTRODUCTION

1.1 The programs described in this section are proprietary software systems that are leased from commercial software vendors. This material is presented for the purpose of informing the programmer of the general functions and services performed by these programs. It is not intended to be a tutorial on the use of these programs as reference is made to various Users Manuals and Reference Guides which contain a complete description of the product and an explanation of its use.

1.2 Cataloged procedures have been prepared to facilitate use of these programs, and are described in Division 3, Chapter 5, Cataloged Procedures.

1.3 Appendix A, Commercial Software Products Reference List, directs the user to brief descriptions of the general functions of the programs.

COMMERCIAL SOFTWARE PRODUCTS REFERENCE LIST

PROGRAM NAME	FUNCTION	APPENDIX	PAGE
AUTOFLOW	Produce flowcharts from program source language statements	B	3
DSF	Dump-restore any disk data set or physical disk segment to magnetic tape	K	19
IMS (DB & DB/DC)	OS based programs supporting user-written batch processing and teleprocessing applications	H	12
LIBRARIAN	Storage and retrieval system for developing and maintaining source programs and other card-image statements	C	4
METACOBOL	A pre-compile package which assists the programmer in writing, testing, debugging and improving ANS COBOL source programs	G	10
MITS (ATS)	An on-line system consisting of control and application programs that permits many different text-processing and data-handling activities to be carried on simultaneously through different communication terminals	I	14
OPTIMIZER II	Reduces program core requirements and execute time by optimizing ANS COBOL compiler produced object module	F	9
RE-ACT	A five program interpretive package for use in retrieving data from sequential or index sequential data files, producing formatted reports as output	J	17
SORT, SYNCSORT III	Sort or merge records	D	6
SMS/360 (PPE)	Study of application program performance	E	8

AUTOFLOW

1. INTRODUCTION

1.1 AUTOFLOW is a proprietary software system leased from Applied Data Research, Inc. The basic function of AUTOFLOW is to translate the source language of a program into a two-dimensional flowchart. Auxiliary listings are produced each time a flowchart is generated. These listings may be used as an aid in analyzing and evaluating the source program.

1.2 AUTOFLOW components included in the Warrenville Data Center package are:

ASSEMBLY
COBOL/CHART
COBOL/COMPRESS
COBOL/XDDA
FORTRAN
PL1
PL1/CHART

1.3 Source program listings produced by AUTOFLOW have been accepted as part of the final documentation package required by Headquarters.

1.4 AUTOFLOW accepts as input source programs written in COBOL, FORTRAN, ASSEMBLER, or PL/I programming languages. Additional information may be found in the following AUTOFLOW reference manuals:

a. Introductory Guide

b. AUTOFLOW Reference Guide - COBOL series

c. AUTOFLOW Reference Guide - FORTRAN series

d. AUTOFLOW Reference Guide - ASSEMBLER series

e. AUTOFLOW Reference Guide - PL/I series

f. AUTOFLOW Operation Guide - IBM System/360

1.5 A cataloged procedure has been prepared to facilitate use of the AUTOFLOW package. A description of the procedure and sample JCL for executing the procedure are described in Division 3, Chapter 5, Cataloged Procedures.

LIBRARIAN

1. INTRODUCTION

1.1 LIBRARIAN is a proprietary software system leased from Applied Data Research, Inc. It is a generalized data storage and retrieval system designed specifically to assist programmers in developing and maintaining source programs. Its usefulness, however, can be applied to storage of other card-image statements.

1.2 The LIBRARIAN source program preparation and maintenance facilities enable a programmer to:

a. Add, delete, and replace entire modules or specified card images within a module.

b. Change strings of characters in card images.

c. Search for strings of characters appearing in one or more modules.

d. Sequence the card numbers of new modules added to the master file.

e. Syntax check COBOL modules added to or updated on the master file.

f. Print and/or punch selected modules.

g. Store descriptive and historical commentary about a module.

h. Combine parts (or all) of several existing modules into a single new module.

i. Place commands in one module which specify inclusion of other modules.

Note: Master file and module are defined in LIBRARIAN reference manuals as:

"master file" - a pre-allocated area of space, disk or tape, in which data is stored in a compressed format.

"module" - can be a source program, a test data file, a collection of JCL statements, or any other grouping of card-oriented information.

1.3 A module can be updated and made available for processing by a subsequent job step. LIBRARIAN can also be directed to produce a complete job stream including the JCL necessary to compile, link-edit, and execute the module.

LIBRARIAN (CONT.)

1.4 To facilitate program checkout, the LIBRARIAN system permits temporary changes to be made to a module for testing purposes before they are permanently applied to the master file module. There are also provisions for simultaneously maintaining test and production versions of the same module.

1.5 Additional information may be found in the following LIBRARIAN Reference Manuals:

a. Introductory Guide

b. User Reference Manual

c. O/S System Reference Manual

d. Concepts and Facilities for EDP Managers

1.6 A cataloged procedure has been prepared to facilitate use of the LIBRARIAN package. A description of the procedure and sample JCL for executing the procedure are described in Division 3, Chapter 5, Cataloged Procedures.

SORT, SYNCSORT III

1. INTRODUCTION

1.1 Syncsort III is a generalized sort/merge program for IBM S/360 and S/370 computers which operates under OS or OS/VS and is a leased product from Whitlow Computer Services.

1.2 To execute the sort/merge program, the user must prepare two types of statements:

1.21 Job Control Language (JCL) statements, describing input and output data sets, which are processed by the operating system.

1.22 Program control statements, describing the users' input records and how they are to be sorted or merged, which are processed by the sort/merge program.

1.3 Syncsort III sort/merge program operates under the operating system control program. JCL statements must be supplied for the operating system to initiate and execute the sort/merge program and perform the desired sort/merge application. Data sets used by the program must be defined according to operating system standards (data definition statements) and must be placed in the input stream with the job step that initiates the sort/merge program. This includes defining input and output data sets and intermediate storage space used as work areas.

1.4 Program control statements describe the input data, provide information about the control fields in the input records which the program will sort or merge, and describe any user exit routines the user may include during program execution. The control statements also include whether the operating system's checkpoint/restart facility is to be utilized, and whether sorting should begin on a record other than the first record in the input data set.

2. SPECIAL FEATURES

2.1 The sort work file restrictions have been modified to significantly reduce the chances of a sort failure due to the lack of available disk space or under allocation by the user.

2.11 The sort work files no longer require contiguous space, therefore, it may be omitted from the DD cards or if supplied it will be ignored.

SORT, SYNCSORT III (CONT.)

2.12 Syncsort III provides an automatic correction for underallocation of sort work files by means of secondary allocation within the following limits. If space was allocated in:

1) cylinders, then Syncsort will acquire up to an additional 10 cylinders.

2) tracks, then Syncsort will acquire up to an additional 200 tracks.

3) blocks, then Syncsort will acquire up to an additional 100 blocks.

2.13 Syncsort III requires a minimum of one sort work file and requires less sort work space than previous sort packages (up to 50 per cent more data can be sorted within a given disk space).

2.2 Syncsort III supports the use of 2311, 2314, 3330, 3330-11, 3340, or any combination of these devices.

2.3 Sort Load Auditing System - Syncsort may be integrated with a special auditing system package and used to collect sort data, develop sort standards, and monitor the implementation of these standards. At the completion of each sort, Syncsort produces a statistical record which contains:

a. characteristics of the input file
b. resources allocated to the sort
c. performance of the sort
d. facilities used by the sort.

These records may be added to the SMF file. The Sort Load Auditing System may be used to select these records from SMF and obtain a detailed profile of the total sorting load.

2.4 Alternate PARM Feature - Syncsort allows the parameters that are passed to the sort by a calling program to be changed without recompiling the calling program. To modify the parameters passed to the sort the user provides the following in the JCL deck:

```
//$ØRTPARM  DD  *
 ØPERATIØN PARAMETER1,PARAMETER2,...,PARAMETER10
```

2.5 Sort Optimization Tools - SYNCSIM and HISTOGRM, which enable a sort user to optimize the performance of a sort, and SORTSTAT, a module which provides the user with detailed information about a sort are distributed with Syncsort.

SORT, SYNCSORT III (CONT.)

2.51 SYNCSIM is a short simulator which can be used to determine the optimal environment for a sort or class of sorts. Its output contains such information as elapsed time, number of I/O accesses, or CPU time which pertain to the performance of the sort.

2.52 HISTOGRM scans variable length records and provides information such as minimum/maximum record length, the modal length, and total number of bytes.

2.53 SORTSTAT provides the user with statistics about a sort such as the number of EXCP's to the sort work data sets, merge orders used, number of strings generated, etc.

2.6 The ddname for the sort message data set will be SYSOUT for normal execution of SORT and SORTPRNT for an internal sort. If the DD statement is omitted, all non-critical messages will be suppressed while critical messages will be directed to the system console. Assembler programs can respecify the SORTPRNT ddname.

2.7 Logical record lengths may be less than 18 bytes.

3. PARM DEFAULTS

3.1 CORE=MAX: This is the default value and recommended value for CORE. It will cause all core within the requested region to be used. In an internal sort 16K will be reserved for the use of the main program (e.g. opening files in an input or output procedure). The maximum amount of core that Syncsort will use in an OS/VS2 environment regardless of any CORE parameters or defaults is 192K.

3.2 MSG=AP: All messages are placed in the sort message data set (SYSOUT or SORTPRNT). Critical messages are directed to the system console.

3.3 LIST: Sort control statements from a normal execution of SORT will be printed.

3.4 Any of these PARMS may be respecified through the use of the $ORTPARM DD * control cards.

4. REFERENCE

4.1 Additional information may be found in the "SYNCSORT III PROGRAMMER'S GUIDE" from Whitlow Computer Services which is available from the "location coordinator".

4.2 A cataloged procedure has been prepared to facilitate use of the SYNCSORT III program. A description of the procedure and sample JCL for executing the procedure are described in Division 3, Chapter 5, Cataloged Procedures.

SMS/360 PROBLEM PROGRAM EFFICIENCY (PPE)

1. INTRODUCTION

1.1 Systems Measurement Software (SMS/360), Problem Program Efficiency (PPE) is a proprietary software system leased from Boole & Babbage, Inc. The PPE product measures the efficiency of problem programs and points out areas that would benefit most from the programmer's attention. The PPE system is composed of two programs, the Extractor and the Analyzer, which are described in the following paragraphs.

1.2 The Extractor program samples the problem program under study at a rate specified by the user for the given run. Extraction of data starts when the problem program begins executing, and terminates upon completion of the problem program under study. This data is recorded on tape or disk for subsequent analysis by the Analyzer program.

1.3 The Analyzer program, using data produced by the Extractor program, generates the reports used to evaluate the overall program efficiency. An Analyzer run can be made immediately following the completion of an Extractor run, or at a later time if the Extractor output data set is saved. The Analyzer can accept one or more extractor data sets for combination onto a single report. Depending on the options selected, the report may include:

a. A histogram showing the percentage of time spent in various sections of the user program.

b. A distribution of I/O wait time associated with each data set.

c. A module map listing all the modules used during execution of the job step and a percentage of time spent in execution of the module.

d. A percentage of time spent in executing SVC's.

1.4 Additional information may be found in the following SMS/360 Systems Measurement Software Reference Manual:

SMS/360 Users Guide for PPE

1.5 A cataloged procedure has been prepared to facilitate use of SMS/360 Problem Program Efficiency. A description of the procedure and sample JCL for executing the procedure are described in Division 3, Chapter 5, Cataloged Procedures.

CAPEX OS 360/370 COBOL OBJECT CODE OPTIMIZER II

1. INTRODUCTION

1.1 The Capex OS 360/370 COBOL OBJECT CODE OPTIMIZER II is a proprietary software product leased from the Capex Corp. It examines the procedure division object code generated by a COBOL ANS compile and reduces the core requirements for execution of the program (II, III and IV versions).

1.2 Core requirements reduction is achieved by substitution of efficient code for some of the inefficient code normally generated by the COBOL compiler. Core reduction is the result of fewer machine instruction in the optimized objective module, thereby reducing the running time of the optimized program as well.

2. WHEN TO USE

2.1 Execution of the OPTIMIZER II requires from 20% to 70% of compile time and therefore should be executed only as the final step in new system development or executed to improve the performance of current stable production programs.

2.2 Since the OPTIMIZER II only deals with the machine instructions, a program with a large PROCEDURE DIVISION is a better candidate than a program with a small PROCEDURE DIVISION. Programs which are CPU bound will achieve greater benefits than programs that are I/O bound.

3. INPUTS

3.1 Input to the OPTIMIZER II are the compiler produced object module and listing.

4. OUTPUTS

4.1 Output from the OPTIMIZER II are the optimized object module and OPTIMIZER II produced listing. This listing enables cross-referencing the original COBOL source module entries with the map of optimized program.

5. REFERENCE

5.1 Additional information may be found in the Capex Corp. 360/370 COBOL OPTIMIZER II USERS GUIDE.

5.2 Catalogued procedures have been prepared to facilitate the use of the Capex OPTIMIZER II program. A description of these procedures and sample JCL for executing the procedure are described in Division 3, Chapter 5, Cataloged Procedures.

METACOBOL

1. INTRODUCTION

1.1 MetaCOBOL is an Applied Data Research (ADR) proprietary software system which assists the programmer in writing, testing, debugging, and improving COBOL source programs. The basic MetaCOBOL System, which is operational on an IBM/360 or /370, provided facilities ranging from simple substitution to conditional generation of sets of standard COBOL statements. Three extensions to the basic system -- the Test Data Generator, Run-time Debugging Aid, and COBOL Performance Monitor -- facilitate COBOL program testing, debugging, and operational improvement respectively. These extensions are described in Chapters 6-8 of the ADR METACOBOL Users Guide.

1.2 MetaCOBOL utilizes macro procedures to search an input source program for words, word sequences, and syntax patterns which, when encountered, cause these words to be replaced according to rules defined within the macro procedures themselves. (For further details, consult MetaCOBOL Macro Writing, Volumes 1 & 2.) The primary output from MetaCOBOL is a standard COBOL source program. The basic Translator can be used to:

1.21 Permit abbreviations for required COBOL words and phrases or user data and procedure names.

1.22 Add entirely new verbs and functions to COBOL.

1.23 Generate COBOL source programs in standardized formats.

1.24 Syntax check generated COBOL output prior to compilation.

1.25 Perform various Utility functions on existing COBOL source -- such as conversion from one compiler level to another, installation standards enforcement, systems management, paragraph numbering, etc. (COBOL F conversion to ANS COBOL is included.)

1.3 The MetaCOBOL Test Data Generator (TDG) is used to create a COBOL source program which will generate test data files. This facility allows the programmer to specify the manner in which data is to be generated.

1.4 The Run-time Debugging Aid (RDA) allows the programmer to print, during program execution, the contents of selected portions of the DATA DIVISION in a form which clearly relates the data to the field names in the source program. If an abnormal termination should occur, the programmer may obtain a list of the names of the last "n" paragraphs, where "n" is determined by the programmer. Processing may then resume after the instruction that caused the abnormal termination. The names of any unexecuted paragraphs can also be listed.

METACOBOL (Cont.)

1.5 The COBOL Performance Monitor (CPM) is used to determine the relative "cost" of each paragraph in an operable program, thus isolating for improvement those paragraphs having the greatest effect on overall program performance.

1.6 METACOBOL macros and utilities available are stored on the SYS2.METACOB.SLIB and SYS2.METALIB partitioned data sets which are concatenated for the "SYSLIB" data set in the cataloged procedure "META" which is documented in Division 3, Chapter 5, Section 1, of this manual. Any locally developed macros should be stored on a separate PDS concaternated to the above SYSLIB PDSs. When full documentation of the macros functions is available, a copy furnished to the Warrenville operations department along with a request to add the macro to one of the above PDSs, will make the new macro available to all Warrenville Data Center users.

1.7 Additional information required for the use of METACOBOL is contained in the following ADR METACOBOL manuals.

A. User Guide
B. MACADR
C. Macro Facility Reference Manual
D. Macro Writing Volume 1 and 2

1.8 A catalogued procedure has been prepared to facilitate use of the METACOBOL package. A description of the procedure and sample JCL for executing this procedure are described in Division 3, Chapter 5, Cataloged Procedures.

INFORMATION MANAGEMENT SYSTEM/360,

VERSION 2 - IMS

1. INTRODUCTION

1.1 IMS is an OS based IBM program product which supports user-written batch processing and teleprocessing applications. It provides supervisory level services in two categories.

- Data base management services which support the implementation of multiple applications in a common data base environment.

- Data communications management services which support the implementation of multiple applications in a shared terminal environment.

1.2 The IMS package consists of two features.

- Data Base System (DB) providing data base management services.

- Data Communication System (DC) providing data communication management services.

DB is a prerequisite for the DC service, hence either the DB or DB/DC Systems can be utilized.

1.3 Applications which operate within the environments supported by DB or DB/DC Systems may be either batch-oriented or transaction-oriented. Batch and transaction-oriented applications may execute concurrently or separately.

1.4 The Interactive Query Facility (IQF) feature is available to users of IMS DB or DB/DC. The feature offers the capability for on-line retrieval and display of data maintained within IMS data bases.

1.5 For additional information refer to Division 1, Chapter 1 and 2 and to the materials referenced in paragraph 3 of this section.

2. PLANNING

2.1 IMS DB and DB/DC services are available to support new applications. Hardware resources to support the system are planned and allocated on the basis of approved projects. Proposal preparation and project planning are to be coordinated with Department 9411 - Computer Systems Software Development and Operations (Warrenville). Department 9411 will assist in the development and implementation of IMS DB and DB/DC oriented projects.

INFORMATION MANAGEMENT SYSTEM/360, VERSION 2 - IMS (CONT.)

3. REFERENCES

3.1 Information Management System/360, Version 2, General Information Manual. IBM order number GH20-0765

3.2 Information Management System/360, Version 2, System/Application Design Guide. IBM order number GH20-0910

3.3 Information Management System/360, Version 2, Application Programming Reference Manual. IBM order number GH20-0911

3.4 Information Management System/360, Version 2, Utilities Reference Manual. IBM order number GH20-0915

3.5 Information Management System/360, Version 2, Messages and Codes Reference Manual. IBM order number SH20-0914

3.6 Interactive Query Facility (IQF) for IMS/360 Version 2, General Information Manual. IBM order number GH20-1074

MANAGEMENT INFORMATION AND TEXT SYSTEM - MITS

1. INTRODUCTION

1.1 The Administrative Terminal System (ATS) designed and developed by IBM has been enhanced. The expanded version of ATS is marketed by H and B Computing of Placentia, California with the title Management Information and Text System (MITS). MITS consists of control and functional programs that permit many different data and text input, editing and formatting activities to be carried on simultaneously through typewriter terminals attached to an IBM SYSTEM/370 Computer. The system was designed for application as follows:

- Publication of large volume transcripts
- Publication and revision of technical manuals and specifications
- Preparation of proposals
- Report writing
- Source data entry at the point of origination.
- Immediate updating and verification of files.
- Hard copy message delivery between terminals.

2. CAPABILITIES

2.1 The basic ATS programming provides the MITS user the following capabilities.

2.11 Enter information into MITS via typewriter keyboard, card reader, or magnetic tape.

2.12 Store existing text and data files.

2.13 Management control and operational responsibility for his text and data files.

2.14 Time-share activities with other data processing activities on the same computer.

2.15 Produce output from rough draft input: including automatic page headings, footings, page numbering, and right-margin justification.

2.16 Intermix free-form and fixed format information.

2.17 Produce remote or central output: high or low speed, uppercase or upper - and lower case printing, magnetic tape disk or punched card.

2.18 Store, retrieve and update from a terminal any defined set of information and prohibit unauthorized retrieval of the information.

2.19 Transmit messages to any other MITS terminal, including the MITS master console.

MANAGEMENT INFORMATION AND TEXT SYSTEM - MITS (CONT.)

2.2 The H and B Computing enhancements of ATS provide the MITS user the following additional capabilities.

2.21 Extract, change, or report statistical information from the data or information entered.

2.22 A series of programmed operations can be stored for repeated execution.

2.23 On-line (MITS) sorting of up to eleven fields of any working storage information.

2.24 Segmentation of large documents

2.25 On-line formatting of data fields

2.26 Release JCL text from MITS to ASP.

2.27 Message transmission to OS master console.

2.28 For complete information on enhancements see the manuals listed in paragraph 5.

3. PLANNING

3.1 MITS is an on-line system. Hardware resources to support the system are planned and allocated on the basis of approved projects requiring MITS capabilities. Proposal preparation and project planning are to be coordinated with Department 9414 - Information Systems Development, Key Data Entry, Management Information Systems (Warrenville). Department 9414 will assist in the development and implementation of MITS projects and/or information systems using MITS as a sub-system.

4. HOURS OF OPERATION

MITS programs are initiated Monday through Friday of each week at 7:00 A.M. and terminated each day at 7:00 P.M. (Warrenville time)

5. REFERENCES

5.1 MITS/370 Users Manual

5.2 MITS/370 Terminal Operator's Manual

MANAGEMENT INFORMATION AND TEXT SYSTEM - MITS (CONT.)

5.3 System/360 Administrative Terminal System - Program Description Manual. IBM Order Number GH20-0582.

5.4 System/360 Administrative Terminal System - Terminal Operator's Manual. IBM Order Number GH20-0589.

5.5 System/360 Administrative Terminal System - Application Description Manual. IBM Order Number GH20-0297.

RE-ACT - DATA RETRIEVAL/REPORT FORMATTING SYSTEM

1. INTRODUCTION

1.1 RE-ACT is a proprietory software package made up of five BAL programs (REACT1, REACT2, REACT3, REACT4 and REACT6) which is procured by CASO from Boeing Computer Services. The system accepts conversational English phrases to retrieve information and generate desired reports. Although the complete system is designed to perform the four basic functions of Retrieval, File Maintenance, Matrix and Special Utilities, the Western Electric module contains only the Retrieval function. This module logically extracts data from users files, processes the information when necessary, and then formats and writes reports on an output medium (printer, magnetic tape or card punch).

1.2 RE-ACT requires an initial definition of the file (data base) in a data name table; these field names may consist of up to 20 characters, including special characters, allowing the requests to be self-documenting. In addition, an elementary security code disallows designated fields from being retrieved by unauthorized personnel. Currently existing files are accessable by RE-ACT as long as they have fixed length records and are sequential/indexed-sequential in organization.

1.21 Variable length records and/or multiple input files can be handled by utilizing user-written code (BAL) at READ time; this facility is the only user-exit in RE-ACT.

2. RE-ACT FUNCTIONS

2.1 Requests to the system are simple to generate because there are only nine (9) verbs and only one special character. Up to forty (40) reports can be produced in one pass of the file.

2.2 Users may sort on multiple fields (up to 50 characters) in ascending, descending or intermixed order. The descending sort can only be used for fields defined to RE-ACT as numeric or packed data. It is also possible to sort on computed fields.

2.3 Summary reports can be combined with detailed reporting or produced alone. RE-ACT provides up to five (5) levels of subtotals as well as grand totals. Summarized reports can be only one print line per output record whereas detailed reports may have up to ten (10) "wraparound" print lines.

2.4 Flexible formatting allows the user to generate multiple page and column headings, designate desired spacing, and provide standard edit masks on printed numeric fields (e.g. leading zero suppression, floating or fixed dollar sign, slashes, dashes, etc.). The user can cause frequently-used report formats to be automatically generated (AUTOFORM).

RE-ACT - DATA RETRIEVAL/REPORT FORMATTING SYSTEM (CONT.)

2.5 RE-ACT provides facilities for the user to set up internal counters and use COMPUTE/ANALYZE statements to obtain such values as might be desired (e.g. averages).

2.6 A re-retrieve capability that allows the user to retrieve records based on general criteria, perform various computations, and then retrieve again based upon the new computed criteria. This facility allows exception reporting and enables the user to report only the specific information desired.

3. REFERENCE

3.1 "RE-ACT System" manual.

4. CATALOGUED PROCEDURE

4.1 A catalogued procedure has been provided to facilitate use of the RE-ACT System. A description and listing of the procedure and sample JCL for executing the procedure are described in Division 3, Chapter 5, Catalogued Procedures.

DSF, DATA SET FUNCTIONS, DATA DUMP/RESTORE UTILITY

1. INTRODUCTION

1.1 DSF is a proprietary utility program of Innovation Data Processing which dumps any disk data set or physical disk segment to magnetic tape. It also restores data sets by name and physical disk segments from DSF generated tapes.

1.2 For sequential and partitioned data sets, DSF dumps and restores the used space, while for all other data sets it dumps and restores all allocated space.

2. ADVANTAGES

2.1 DSF is a faster method of dumping and restoring disk data sets than provided by OS standard utilities.

2.2 DSF is less expensive to run because of I/O, CPU, SIGMA, and MBS times which are lower than a comparable run of standard OS utility programs.

2.3 Control statements required for DSF are simple. JCL provides for device allocation and volume identification while the DSF control statement specifies the action, DUMP or RESTORE, and the data sets.

2.4 Stacking is simplified. Up to ten data sets may be dumped from the same volume as a single "dump data set" and yet restored selectively.

3. DISADVANTAGE

3.1 Data sets to be restored must be preallocated with sufficient primary space to contain the data.

3.2 Only a single disk volume may be referenced in an execution of DSF.

3.3 Data sets are restored with the original creation date, hence they cannot be safely restored to "storage class" packs (ie STORnn) which are subjected to a date dependant scratch utility.

4. SUGGESTED USES

4.1 DSF can be used to back up any disk data set on tape.

4.2 DSF can be used to facilitate off-line storage of disk data sets on tape. DSF will rapidly load and unload them to and from disk volumes. Up to ten data sets located on the same disk may be unloaded to the same tape.

DSF, DATA SET FUNCTIONS, DATA DUMP/RESTORE UTILITY (CONT.)

5. PERFORMANCE COMPARISON

The following information gives comparative figures for dumping and restoring a direct access file using the O.S. utility, IEHMOVE and DSF.

	IEHMOVE		DSF	
	Dump	Restore	Dump	Restore
EXCP	52,596	52,623	646	882
I/O	4 min/9 sec.	7 min/28 sec.	45 sec.	47 sec.
CPU	30.43 sec.	22.37 sec.	2.87 sec.	1.37 sec.
CORE	66K	62K	92K	76K
QCM COST	$117.45	$214.12	$22.78	$22.86

6. SAMPLE JCL

6.1 EXAMPLE 1. Dumping two data sets.

```
//DMP EXEC PGM=FDRDSF,REGIØN=94K,ADDRSPC=REAL
//SYSPRINT DD SYSØUT=A
//DISK1 DD UNIT=DISK,DISP=SHR,VØL=SER=PVØL63
//TAPE1 DD UNIT=TAPE,DSN=PRD4.DPM.DUMPDSF,
//         DISP=(,CATLG),LABEL=RETPD=60
 DUMP DSN=PRD4.DPM.MASTER1
 DUMP DSN=PRD4.DPM.TABLEX
```

6.2 EXAMPLE 2. Restoring one data set.

```
// EXEC PGM=FDRDSF,REGIØN=80K,ADDRSPC=REAL
//SYSPRINT DD SYSØUT=A
//DISK1 DD DSN=PRD4.DPM.MASTER1,UNIT=DISK,VØL=SER=PVØL63,
//         DISP=(,KEEP),SPACE=(5,1))
//TAPE1 DD DSN=PRD4.DPM.DUMPDSF,DISP=ØLD
  RESTØRE DSN=PRD4.DPM.MASTER1
```

6.3 When allocating a 3330 Model 11 it is necessary to identify the Model 11 by means of a DCB parameter, namely DCB=DEN=2.

7. REFERENCE

7.1 Additional information can be found in Innovation Data Processing, Fast Dump Restore and Data Set Functions User Documentation. A copy of this publication is available at each location served by Warrenville.

At Warrenville, two types of partitioned data sets are available for the efficient storage of small amounts of card image information. One type identified as PRDn.CONTROL is intended for holding information that remains relatively stable such as sort control statements or utility control statements. The second type identified as PRDn.PARMLIB is used for control information generated by an application program, perhaps to be passed to another job in the system or the next scheduled run. The distinction between the libraries is that PARMLIB's are open to update by programmers while CONTROL libraries can only be updated via the PLC system.

What these libraries have in common is that their members usually occupy a small amount of space, a fraction of a track. The advantage to our operation is that we are able to store up to 61 records per track. Consider the alternative: creating each member as a separate data set would result in using a full track for each equivalent member. For example, 61 single card members would occupy 61 tracks as separate data sets.

Should these libraries be used for storing larger amounts of data, perhaps in excess of 60 records? No, because the records in these libraries are unblocked and do not efficiently store multiple record members. The CONTROL and PARMLIB originated as places to store information that came from the job stream (//SYSIN DD *), consequently, they carry the record characteristics of instream data, 80 characters and unblocked. They must maintain these characteristics to remain compatible with instream data.

An application that needs more than a track of 3330 space for storage of control information should consider a more efficient method of storage than provided by these card image libraries. Alternatives might include separate sequential data sets or a private partitioned data set efficiently blocked. For example, a blocksize of 800 increases storage to 140 records per track, a blocksize of 12960 provides 162 records per track, as opposed to 61 records per track unblocked.

PARMLIB libraries should be considered for those applications that write one or two records on a track of space. There are some precautions to be observed however, since PARMLIB's are open to public update. Programmers should employ a technique that either updates inplace or issues a device RESERVE. The sub-routine CCRETREV/CCUPDATE provides the facility of updating inplace. The utility program ATTUPDTE, invoked in the procedure CHGPDS, issues a RESERVE. We recommend its use for adding a member to a PARMLIB initially.

We hesitate to define hard and fast rules for use of these libraries. We do request that users give some thought to storage of data ranging from one card to several hundred cards. Should the user decide to add data to our libraries that may exceed a track, we request a warning. The PARMLIB and CONTROL libraries currently range from 4 tracks to 5 cylinders, consequently, unexpected large amounts of input can create problems. A user could fill a PARMLIB, or disrupt the scheduled update of PLC controlled libraries. An advance notice will allow us to take appropriate action to handle the situation.

SWITCHING EQUIPMENT DIVISION

WARRENVILLE DATA CENTER

PROGRAMMING NOTES NO. 6 5-5-76

SUBJECT - USE OF CONTROL AND PARMLIB LIBRARIES

MANUAL REFERENCES - PROGRAMMING S & R MANUAL

PREPARED BY - E. W. RZYM, DEPT. 9412, WARRENVILLE CORNET 391-5245

APPROVED: R. J. Bonelli
R. J. BONELLI - 9412

DATA PROCESSING SYSTEM PROGRAMMING AIDS

SUBROUTINES

SECTION 1 - GENERAL

1. INTRODUCTION

1.1 A subroutine is a sequence of instructions that can be incorporated into a program to perform a specific operation. The subroutines presented in this section were developed by Western Electric personnel to simplify the writing of extensive coding of source programs to accomplish commonly required functions or to obtain commonly required data.

1.2 Subroutines described in this Section are primarily intended for the COBOL programming language. They reside as load modules in the automatic call library, SYS1.COBLIB, referenced in the Data Center COBOL procedures.

1.3 Appendix A, Subroutine Reference List, directs the user to an Appendix which contains a description of the subroutine and examples of usage.

SUBROUTINE REFERENCE LIST

SUBROUTINE NAME	FUNCTION	APPENDIX	PAGE
ADJUSTL	Remove leading blanks, field manipulation	B	3
CALENDAR	Convert calendar date to Julian date	N	28
CCRETREV/CCUPDATE	Control record retrieve and update in place	C	4
CMPDATE5	Apply an interval to a fiscal date	D	6
CMPDATE6	Apply an interval to a calendar date	E	8
CVDATE	Fiscal date conversion	F	10
CVDATEF6	Fiscal date conversion	G	11
CVDATE6F	Calendar date conversion	H	12
FISCALPD	Convert six digit fiscal date to Julian date	O	29
FISCALWK	Convert five digit fiscal date to Julian date	P	30
GETDATE	Date manipulation; PARM passing; set condition code	I	13
GETTMT	Unformatting transmission files	J	17
PARMEXCD	Obtain EXEC card PARM information	K	22
PUTTMT	Formatting files for transmission	L	23
RANDOM	A psuedo-random number generator	Q	31
WESNAP	Callable dump for debugging	M	27

ADJUSTL - REMOVE LEADING BLANKS, FIELD MANIPULATION

1. GENERAL

ADJUSTL is a subroutine which will remove leading blanks and left adjust an input field as it is moved to an output field. The output field size may be lengthened, shortened, or left unchanged. Field modification is accomplished by the user specifying a value corresponding to the number of bytes of the input record, beginning with the left-hand byte, that are to be "adjusted" and moved to the output record. The maximum size of the field to be "adjusted" is 256 characters.

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
77  INREC          PICTURE X (size of input field).
77  ØUTREC         PICTURE X (size of output field).
77  LGT            PICTURE S999 VALUE +nnn COMP SYNC.
```

where "nnn" is the maximum number of bytes to be moved including leading blanks. The value of "nnn" must be greater than zero and not greater than 256.

2.2 Procedure Division.

```
MØVE fieldname TØ INREC.
CALL 'ADJUSTL' USING INREC, ØUTREC, LGT.
```

3. EXAMPLES OF USE

Example 1: Edit and left justify an 8 byte field.
Input: '   123 '
Result: '123 uuu'

Example 2: Edit and left justify 6 bytes of an 8 byte field. The output field, defined by the user, is 10 bytes in length.
Input: ' 12 345'
Result: '12 3uuuuuu'

Example 3: Edit and left justify 6 bytes of an 8 byte field. The output field, defined by the user, is 10 bytes in length.
Input: '12345678'
Result: '123456uuuu'

Example 4: Edit and left justify 6 bytes of a 10 byte field. The output field, defined by the user, is 8 bytes in length.
Input: ' 12345678'
Result: '1234uuuu'

Note: In the above examples, "u" represents data that is unchanged from the previous use of OUTREC. The user may wish to initialize OUTREC prior to issuing the CALL statement.

CCRETREV/CCUPDATE - CONTROL RECORD RETRIEVE AND UPDATES IN PLACE

1. GENERAL

CCRETREV/CCUPDATE is a dual entry subroutine used to retrieve and update in place control type records which are passed from one run of a system to the next. It allows the user to do the following:

a. Retrieve from one to three 80 byte blocks of information from a member of a library (CCRETREV entry point).

b. Update the member in place within the same jobstep (CCUPDATE entry point).

The CCRETREV/CCUPDATE module is approximately 600 bytes in length.

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

The following should be coded and used as the argument in the CALL statement:

```
01  CØNTRØL-INFØ.
    03  REC1                        PICTURE X(80).
    03  REC2                        PICTURE X(80).
    03  REC3                        PICTURE X(80).
```

2.2 Procedure Division.

CCRETREV/CCUPDATE is executed through a CALL statement coded as follows:

```
CALL 'CCRETREV' USING CØNTRØL-INFØ.
CALL 'CCUPDATE' USING CØNTRØL-INFØ.
```

3. USING CCRETREV/CCUPDATE

3.1 CONTROL-INFO is an area large enough to contain the complete member being retrieved and updated. After execution of CCRETREV, CONTROL-INFO contains the information obtained from the member of the partitioned data set. Before execution of CCUPDATE, CONTROL-INFO should contain the information to be passed via the member to the next run of the system.

3.2 If the member retrieved by CCRETREV need not be changed, then it is not necessary to call CCUPDATE. However, a member may not be updated using CCUPDATE if a previous call to CCRETREV has not been issued in the same program.

CCRETREV/CCUPDATE - CONTROL RECORD RETRIEVE AND UPDATE,(CONT.)

3.3 Information may initially be stored in the library by using the W.E. utility program ATTUPDTE (Div. 3, Chap. 2, Sec. 2). Do not code DCB information for the library since it is already present in the DSCB.

3.4 An additional JCL DD card is required for use by the CCRETREV/CCUPDATE program. This card is named CNTRLREC and must specify the library and the member to be accessed. Coding for the library should be DISP=SHR.

3.5 To test a program which calls CCRETREV/CCUPDATE, the user should first create a temporary partitioned data set containing his member. One track and one directory block should be allocated for this data set. Its record format must be fixed, with a record length and blocksize of 80.

4. EXAMPLE OF USAGE

4.1 The following ANS COBOL program was used while testing the CCRETREV/CCUPDATE subroutine:

```
IDENTIFICATIØN DIVISIØN.
PRØGRAM-ID. UPDTTEST.
DATA DIVISIØN.
WØRKING-STØRAGE SECTIØN.
01  CØNTRØL-INFØ.
    03  REC1         PICTURE X(80).
    03  REC2         PICTURE X(80).
    03  REC3         PICTURE X(80).
PRØCEDURE DIVISIØN.
CALL 'CCRETREV' USING CØNTRØL-INFØ.
DISPLAY REC1.
DISPLAY REC2.
DISPLAY REC3.
MØVE 'CHANGE RECØRD 1' TØ REC1.
MØVE 'CHANGE RECØRD 2' TØ REC2.
MØVE 'CHANGE RECØRD 3' TØ REC3.
CALL 'CCUPDATE' USING CØNTRØL-INFØ.
CALL 'CCRETREV' USING CØNTRØL-INFØ.
DISPLAY REC1.
DISPLAY REC2.
DISPLAY REC3.
GØBACK.
```

The displayed results follow:

```
CONTROL RECORD 1
CONTROL RECORD 2
CONTROL RECORD 3
CHANGE RECORD 1
CHANGE RECORD 2
CHANGE RECORD 3
```

5. REFERENCE

Refer to discussion of Parameter Libraries in Division 2, Chapter 5, Section 1.

CMPDATE5 - APPLY INTERVAL TO A FISCAL DATE

1. GENERAL

CMPDATE5 is a subroutine used to apply an interval, earlier or later, to a five digit fiscal date (in the form of year, fiscal week, day of week). The interval is stated as work weeks and work days (based on a five day work week) with a maximum of 99 weeks and 5 days.

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
77  RTCØDE              PICTURE XX.
01  CØMP-DATE-IN-5.
    03  CI5-YEAR        PICTURE 99.
    03  CI5-WEEK        PICTURE 99.
    03  CI5-DAY         PICTURE 9.

01  COMP-DATE-ØUT-5.
    03  CØ5-YEAR        PICTURE 99.
    03  CØ5-WEEK        PICTURE 99.
    03  CØ5-DAY         PICTURE 9.

01  INTERVAL.
    03  INTVL-WEEKS     PICTURE 99.
    03  INTVL-DAYS      PICTURE 9.
    03  INTVL-SIGN      PICTURE X.
```

Note: INTVL-SIGN must have one of the following values:

"E" to subtract the interval from the date
"L" to add the interval to the date

2.2 Procedure Division.

CMPDATE5 is executed through a call statement coded as follows:

```
CALL 'CMPDATE5' USING CØMP-DATE-IN-5, CØMP-DATE-ØUT-5, INTERVAL, RTCØDE.
```

3. RETURN CODES

A return code is passed to the COBOL program upon execution of CMPDATE5. The return codes are:

00 - Successful
01 - Date not all numeric
02 - Invalid date (see Paragraph 4, RESTRICTIONS)
03 - Interval not numeric
04 - Invalid interval (INTVL-SIGN must be "E" or "L")

CMPDATE5 - APPLY INTERVAL TO A FISCAL DATE (CONT.)

4. RESTRICTIONS

4.1 The year to which the interval applies must be within the range of 73 through 78.

4.2 The interval to be applied must be stated as work weeks and work days (based on the five day work week).

CMPDATE6 - APPLY INTERVAL TO A CALENDAR DATE

1. GENERAL

CMPDATE6 is a subroutine used to apply an interval, earlier or later, to a six digit calendar date (in the form of year, month, day). The interval is stated as calendar weeks and calendar days (based on a seven day week) with a maximum of 99 weeks and 7 days.

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
77  RTCØDE              PICTURE XX.
01  CØMP-DATE-IN-6.
    03  CI6-YEAR        PICTURE 99.
    03  CI6-MØNTH       PICTURE 99.
    03  CI6-DAY         PICTURE 99.

01  CØMP-DATE-ØUT-6.
    03  CØ6-YEAR        PICTURE 99.
    03  CØ6-MØNTH       PICTURE 99.
    03  CØ6-DAY         PICTURE 99.

01  INTERVAL.
    03  INTVL-WEEKS     PICTURE 99.
    03  INTVL-DAYS      PICTURE 9.
    03  INTVL-SIGN      PICTURE X.
```

Note: INTVL-SIGN must have one of the following values:

"E" to subtract the interval from the date
"L" to add the interval to the date

2.2 Procedure Division.

CMPDATE6 is executed through a call statement coded as follows:

```
CALL 'CMPDATE6' USING CØMP-DATE-IN-6, CØMP-DATE-ØUT-6, INTERVAL, RTCØDE.
```

3. RETURN CODES

A return code is passed to the COBOL program upon execution of CMPDATE6. The return codes are:

00 - Successful
01 - Date not all numeric
02 - Invalid date (see Paragraph 4, RESTRICTIONS)
03 - Interval not numeric
04 - Invalid interval (INTVL-SIGN must be "E" or "L")

CMPDATE6 - APPLY INTERVAL TO A CALENDAR DATE (CONT.)

4. RESTRICTIONS

The interval to be applied must be stated as calendar weeks and calendar days (based on a seven day week).

CVDATE - FISCAL DATE CONVERSION

1. GENERAL

CVDATE is a subroutine used to convert a five digit fiscal date (in the form of year, fiscal week, day of week: 73225) to a six digit calendar date (in the form of year, month, day: 730601).

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
77  RTCØDE                PICTURE XX.
01  UNCØNVERTED-DATE.
    03  UCV-YEAR          PICTURE 99.
    03  UCV-WEEK          PICTURE 99.
    03  UCV-DAY           PICTURE 9.

01  CØNVERTED-DATE.
    03  CV-YEAR           PICTURE 99.
    03  CV-MØNTH          PICTURE 99.
    03  CV-DAY            PICTURE 99.
```

2.2 Procedure Division.

CVDATE is executed through a CALL statement coded as follows:

```
CALL 'CVDATE' USING UNCØNVERTED-DATE, CØNVERTED-DATE, RTCØDE.
```

3. RETURN CODES

A return code is passed to the COBOL program upon execution of CVDATE. The return codes are:

00 - Successful
01 - Date not all numeric
02 - Invalid date (see Paragraph 4, RESTRICTIONS)

4. RESTRICTIONS

The year being converted must be within the range of 71 through 80.

CVDATEF6 - FISCAL DATE CONVERSION

1. GENERAL

CVDATEF6 is a subroutine used to convert a six digit fiscal date (in the form of year, month, week of month, day of week: 730533) to a six digit calendar date (in the form of year, month, day: 730516).

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
77  RTCØDE              PICTURE XX.
01  FISCAL-DATE.
    03  F-YEAR          PICTURE 99.
    03  F-MØNTH         PICTURE 99.
    03  F-WEEK          PICTURE 9.
    03  F-DAY           PICTURE 9.

01  CALENDAR-DATE.
    03  CAL-YEAR        PICTURE 99.
    03  CAL-MØNTH       PICTURE 99.
    03  CAL-DAY         PICTURE 99.
```

2.2 Procedure Division.

CVDATEF6 is executed through a CALL statement coded as follows:

```
CALL 'CVDATEF6' USING FISCAL-DATE, CALENDAR DATE, RTCØDE.
```

3. RETURN CODES

A return code is passed to the COBOL program upon execution of CVDATEF6. The return codes are:

00 - Successful
01 - Date not all numeric
02 - Invalid date (see Paragraph 4, RESTRICTIONS)

4. RESTRICTIONS

The year being converted must be within the range of 71 through 80.

CVDATE6F - CALENDAR DATE CONVERSION

1. GENERAL

CVDATE6F is a subroutine used to convert a six digit calendar date (in the form of year, month, day: 730601) to a six digit fiscal date (in the form of year, month, week-of-month, day-of-week: 730615).

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
77  RTCØDE                PICTURE XX.
01  CALENDAR-DATE.
    03  CAL-YEAR          PICTURE 99.
    03  CAL-MØNTH         PICTURE 99.
    03  CAL-DAY           PICTURE 99.

01  FISCAL-DATE.
    03  F-YEAR            PICTURE 99.
    03  F-MØNTH           PICTURE 99.
    03  F-WEEK            PICTURE 9.
    03  F-DAY             PICTURE 9.
```

2.2 Procedure Division.

CVDATE6F is executed through a CALL statement coded as follows:

```
CALL 'CVDATE6F' USING CALENDAR-DATE, FISCAL-DATE, RTCØDE.
```

3. RETURN CODES

A return code is passed to the COBOL program upon execution of CVDATE6F. The return codes are:

00 - Successful
01 - Date not all numeric
02 - Invalid date (see Paragraph 4, RESTRICTIONS)

4. RESTRICTIONS

The year being converted must be within the range of 71 through 80.

GETDATE - DATE MANIPULATION, PARM PASSING, SET CONDITION CODE

1. GENERAL

GETDATE (GETDATE 3, Version 5) is a subroutine used to provide the following:

1.1 Contents of the PARM parameter of the EXEC card.

1.2 The following dates based on the Operating System date:

a. Julian date (YYDDD)
b. Calendar date (YYMMDD)
c. Fiscal week number (01 thru 52 or 53)
d. Fiscal period (YYMMW)
e. Fiscal day of week (1 thru 7)

1.3 The time indicator:

A for AM (0001 hours to 1200 hours)
P for PM (1201 hours to 2400 hours)

1.4 The standard constant to compare a line counter for carriage overflow.

1.5 The Works Identification Code:

1.51 This code is a two position location code obtained from the programmer name field on the JOB card at program execution time (see Div. 2, Chap. 4, Sect. 4). If the location code so obtained from the JOB card is one of the following POIS locations, the GETDATE Works Identification field will be a single character (left justified in the two character field) as shown below.

JOB Card Programmer Name Field Location Code	GETDATE Works Identification	POIS Location
'CB'	'J '	Columbus Works
'HW' or 'WN'	'H '	Hawthorne Works
'OC'	'Y '	Oklahoma City Works
'OH'	'U '	Omaha Works
'VC'	'C '	Columbia River Plant

1.52 All other JOB card location codes will be used as is for the GETDATE Works Identification field.

1.53 If an error is detected or this routine is unable to read this information from the JOB card, the GETDATE Works Identification field will contain spaces. Is is therefore necessary for programs dependent upon this to include a test for spaces.

1.6 The ability to pass a condition code to the next job step.

1.7 The ability to compute all dates listed in 1.2 above, from a user supplied Julian date (YYDDD).

CAUTION: If the Julian date provided by the calling program is invalid, i.e., the DDD portion is less than 001 or greater than 366, the value returned to the calling program will be only the Julian date that GETDATE received.

GETDATE - DATE MANIPULATION, PARM PASSING, SET CONDITION CODE (CONT.)

1. GENERAL (Cont.)

1.8 The ability to abnormally terminate the job step with a user code.

1.9 The ability to obtain the jobname and stepname of the job stream the calling program is running in.

2. SPECIAL CONSIDERATIONS WHEN USING GETDATE

2.1 The current version of GETDATE does not display any error messages. Note the CAUTION in par. 1.7 and also note par. 1.53.

2.2 The GETDATE routine must be called prior to the calling of any of the other routines (paragraphs 1.6 through 1.9).

2.3 There are no known problems in the current version of this subroutine which has been tested and validated through December 31, 1999. The following years have 53 fiscal weeks: 1971, 1976, 1982, 1988, 1993, and 1999. (Years having 53 fiscal weeks are defined in C.I. 95.183, Section 33, Appendix B).

2.4 This subroutine links to another program (GETPGMNM) which resides in SYS2.LINKLIB.

3. EXAMPLE OF USAGE

3.1 Obtaining the PARM field from the EXEC statement, control dates computed from the system date, a time indicator, a line constant, and a Works Identifier (see par. 1.51).

```
DATA DIVISION.
    .
WORKING-STORAGE SECTION.
    .
01  EXEC-PARM.
    03  FILLER             PICTURE XX.
    03  PARM-DET           PICTURE X(39).
01  CØNTRØL-DATES.
    03  JULIAN-DATE        PICTURE S9(5).
    03  CALENDAR-DATE      PICTURE S9(6).
    03  FISCAL-WEEK-NØ     PICTURE S99.
    03  FISCAL-PERIØD      PICTURE S9(5).
    03  LINE-CØNSTANT      PICTURE S99.
    03  DAY-ØF-WEEK        PICTURE S9.
    03  AM-PM-INDICATØR    PICTURE X.
    03  WØRKS-ID           PICTURE XX.
    .
PROCEDURE DIVISION.
    .
CALL 'GETDATE' USING EXEC-PARM, CØNTRØL-DATES.
```

GETDATE - DATE MANIPULATION, PARM PASSING, SET CONDITION CODE, (CONT.)

3.2 Passing a condition code to the next job step.

```
DATA DIVISION.
     .
WORKING-STORAGE SECTION.
     .
01  CØNTRØL-DATES.  (As shown in par. 3.1)
     .
01  CØNDITIØN-CØDES.
    03  STEP-CØDE         PICTURE S99            CØMPUTATIØNAL.
    03  GØ-CØDE           PICTURE S99  VALUE 00  CØMPUTATIØNAL.
    03  CØNTRØL-ERRØR     PICTURE S99  VALUE 04  CØMPUTATIØNAL.
    03  SEQUENCE-ERRØR    PICTURE S99  VALUE 08  CØMPUTATIØNAL.
    03  I-Ø-ERRØR         PICTURE S99  VALUE 12  CØMPUTATIØNAL.
     .
PROCEDURE DIVISION.
     .
CALL 'GETDATE'......(as shown in par. 3.1)
MØVE CØNTRØL-ERRØR TØ STEP-CØDE.
CALL 'SETCØDE' USING STEP-CØDE.
```

3.3 Computing the control dates from a user supplied Julian date.

```
DATA DIVISION.
     .
WORKING-STORAGE SECTION.
     .
01  CØNTRØL-DATES.  (As shown in par. 3.1)
     .
01  USER-DATE                PICTURE S9(6)  CØMPUTATIØNAL-3.
     .
PROCEDURE DIVISION.
     .
CALL 'GETDATE'......(as shown in par. 3.1)
MØVE  73269 TØ USER-DATE.
CALL 'MYDATE' USING USER-DATE, CØNTRØL-DATES.
```

GETDATE - DATE MANIPULATION, PARM PASSING, SET CONDITION CODE, (CONT.)

3.4 Abnormally terminate the jobstep with a user code.

```
DATA DIVISION.
      .
WORKING-STORAGE SECTION.
      .
01  CØNTRØL-DATES.  (As shown in par. 3.1)
      .
01  ABEND-USER-CØDE       PICTURE 9(4)  VALUE ZERØS.
      .
PROCEDURE DIVISION.
      .
CALL 'GETDATE'......(as shown in par. 3.1)
MØVE  0401 TØ ABEND-USER-CØDE.
CALL 'ABENDIT' USING ABEND-USER-CØDE.
```

3.5 Obtain the jobname and stepname that the user program is running in.

```
DATA DIVISION.
      .
WORKING-STORAGE SECTION.
      .
01  CØNTRØL-DATES.  (As shown in par. 3.1)
      .
01  ERRØR-MSG-TØ-PRINT.
    03  CØDED-MSG          PICTURE X(31)  VALUE SPACES.
    03  FILLER             PICTURE X(5)   VALUE ' JØB '.
    03  JØB-NAME           PICTURE X(8)   VALUE SPACES.
    03  FILLER             PICTURE X(6)   VALUE ' STEP '.
    03  STEP-NAME          PICTURE X(8)   VALUE SPACES.
      .
PROCEDURE DIVISION.
      .
CALL 'GETDATE'......(as shown in par. 3.1)
CALL 'JNAME' USING JØB-NAME, STEP-NAME.
```

4. REFERENCE

Responsible programmer, J. L. Halley, Oklahoma City.

GETTMT - UNFORMATTING TRANSMISSION FILES

1. GENERAL

GETTMT is a subroutine which accepts transmission input files for Fixed-Blocked, Fix-Unblocked, and Variable-Length records as specified in C.I. 95.183, Section 35.

The subroutine has three functions: Open, Read, and Close. The Open routine passes the header label (THD), the Read routine passes a logical record, and the Close routine passes the trailer label (TTLE). O.S. Opens and Closes are handled by the subroutine, as well as stacked data sets.

The subroutine, which requires 3K, should be linkage edited with the calling program.

2. USING GETTMT

2.1 Environment Division.

A select clause should not exist for input files processed by the subroutine.

2.2 Data Division.

File Section.

FD information should not exist for input files processed by the subroutine.

Working-Storage Section.

The following area should be coded and used as the arguments in the CALL statement.

```
01  PASSDATA.
    03  ACTCØDE                  PICTURE 99.
    03  REELCNT                  PICTURE S99 CØMP.
    03  BLKCNT                   PICTURE S99 CØMP.
    03  RECCNT                   PICTURE S99 CØMP.
    03  PBLKSIZE                 PICTURE 9(5).
    03  LRECSIZE                 PICTURE 9(5).
    03  CØMMAND                  PICTURE 9.
    03  ERØPT                    PICTURE XXX VALUE 'CØDE'.
    03  RCVAREA                  PICTURE X(NUM).
```

2.2 Data Division (Contd)

Description of Fields:

2.21 PASSDATA is the argument used in CALL statement.

2.22 ACTCODE will contain a two-digit numeric upon every return from the subprogram. This unsigned numeric will be indicative of the function that was performed by the subroutine. The following table summarizes the meaning and the recommended user-program logic for each ACTCODE.

	MEANING(S)	USER ACTION(S)
READ CODES	User called GETTMT with COMMAND=1, READ	
10	Successful Read and Complete.	Continue Processing.
12	EØD, EØF, TTLE encountered.	Call GETTMT with close COMMAND.
14	Invalid function, tried to read a file that was never opened.	Call GETTMT with open COMMAND.
16	Reel read to tape mark, no required trailer label (TTLE) found for the Label Data Set.	File considered closed. Open the next Label Data Set.
18	A 001 error was encountered by GETTMT and the user valued EROPT as POS. Current Label Data Set bypassed and now Considered Closed.	Issue an Open for the next Label Data Set.
OPEN CODES	User called GETTMT with COMMAND=2, OPEN	
20	Successful open and complete Header Label (THD) exists in RCVAREA.	Continue Processing.
22	No Header Label (THD) found as first record after a trailer label (TTLE).	Close the file. In effect the user is bypassing the processing of the file.
24	Invalid function, tried to OPEN a file, but previous file was not closed.	Call GETTMT with CLOSE COMMAND.
26	Invalid record format in header. GETTMT cannot process the file.	Must call GETTMT with a Close Command. In effect the user is bypassing the label data set.

2.2 Data Division (Contd)

CLOSE CODES	User called GETTMT with COMMAND=3, CLOSE	
30	Successful CLOSE and COMPLETE. Trailer label (TTLE) exists in RCVAREA. BLOCK and RECORD counts check with label.	Continue Processing.
32	Successful CLOSE and COMPLETE. Trailer exists in RCVAREA, but BLOCK and/or RECORD counts do not check with label.	Continue Processing.
34	Invalid function, tried to close a file that was never opened.	Call GETTMT with an OPEN COMMAND.
ANY COMMAND CODE		
00	Invalid command, not an unsigned 1, 2, or 3.	Call subroutine with desired command.

2.23 REELCNT, a computational field, indicates the number of input reels to be processed. This field must be initialized by the user. For example, if two reels are to be processed the REELCNT should be initialized to two and will equal zero at the end of the last label data set. The user should check the REELCNT immediately after all open calls to the subroutine. If the subroutine is called after the REELCNT equals zero, a USER 999 abend will be issued by GETTMT.

2.24 BLKCNT is a counter that the subroutine uses to sum the number of blocks read. Because this field is used by GETTMT in comparing the trailer label (TTLE) count, the user should never alter or change the contents of this field.

2.25 RECCNT (Same as BLKCNT description above).

2.26 PBLKSIZE - With FB or FU formatted records this field will contain the physical block size as indicated on the header label. This will be the block size that will be used to process the file. With VB formatted records and upon return from an Open, this field will contain the maximum physical block size of largest block on the file. Upon return from a read, it will contain the size of the current block.

General - This field should never be altered by the user.

GETTMT - UNFORMATTING TRANSMISSION FILES, (CONT.)

2.27 LRECSIZE - With FB or FU formatted records, this will be the logical record size that the subroutine will use to process the records.

With VB formatted records and upon a return from an Open, it will contain the size of the largest record on the file. Upon return from a Read, it will contain the record size of the current record passed.

General - This field should never be altered by the user.

2.28 COMMAND - This field must contain the numeric 1, 2, or 3. The subroutine uses this field as follows:

1 = READ A RECORD
2 = OPEN A FILE
3 = CLOSE A FILE

NOTE: The command must also be an unsigned 1, 2, or 3. Any other data in this field will cause an ACTCODE of "00" to be set. (See ACTCODE 2.22).

2.29 EROPT - Code must be initialized to one of the following:

'ACC' - Accept all data blks with 001 errors
'SKP' - Skip all data blks with 001 errors
'ABE' - Abend the step on a 001 error
'POS' - Position to the next label data set on all 001 errors. An ACTCODE of 18 will be returned on this condition. (See ACTCODE 2.22).

2.30 RCVAREA PIC X(NUM).
NUM must contain the maximum record length the user will process. In any condition, this number must not be greater than 1000 (the largest size the standard will allow) or less than 80 (the record length of the labels).

Upon successful GETTMT actions, RCVAREA will contain:

ØPEN - 80 character header of the opened file (THD)
READ - a logical record
CLØSE - 80 character trailer of the closed file (TTLE)

NOTE: All headers, trailers, and records will be left justified in RCVAREA.

GETTMT - UNFORMATTING TRANSMISSION FILES, (CONT.)

2.3 Procedure Division

The subroutine is invoked by the following statement:

```
CALL 'GETTMT' USING PASSDATA.
```

2.4 Permanent I/O Errors

GETTMT gives users full control on I/O errors through the EROPT argument. It also prints out a descriptive message concerning all I/O errors onto a sysout data set called GETOUT. (See JCL 2.52)

2.5 JCL

2.51 GETTMT expects all input files to be described by one DD statement with the DDNAME of TMTTAPE. For example:

```
//TMTTAPE DD DSN=ANY,LABEL=(,NL),DISP=ØLD,UNIT=TAPE,VØL=SER=NAME
```

Note: No DCB information should be coded.

2.52 GETTMT requires a sysout data set named GETOUT. It must be coded as follows:

```
//GETØUT DD SYSØUT=A
```

GETOUT will print all headers as they are encountered followed by descriptive 001 error messages should they occur.

PARMEXCD - OBTAIN EXEC CARD PARM INFORMATION

1. GENERAL

PARMEXCD is a subroutine that allows the user to:

a. Obtain up to 100 bytes of data from the PARM parameter of the EXEC card and place the data in the users program.

b. Place the length of the PARM in the users program.

2. CODING REQUIREMENTS

2.1 Data Division.

```
Working-Storage Section.
01  PARM-EXEC-CD.
    03  PARM-LENGTH                PICTURE 999 VALUE ZERØS.
    03  PARM-DATA                  PICTURE X (Range 1 to 100) VALUE SPACES.
```

2.2 Procedure Division.

```
        CALL ' PARMEXCD' USING PARM-EXEC-CD.
```

2.3 If no PARM information is passed, a length of 000 will be returned to the users work area.

PUTTMT - FORMATTING FILES FOR TRANSMISSION

1. GENERAL

PUTTMT is a subroutine which produces transmission output files for Fixed-Blocked and Fixed-Unblocked records as specified in C.I. 95.183, Section 35. No provision is made to process Variable-Length records.

The subroutine is broken down into three functions: Open, Write, and Close. The Open routine writes the standard header (THD), the Close routine writes the standard trailer (TTLE), and the Write routine writes a record. Blocking of records, padding the last block with nines, and performing O. S. Open and Close functions on the files are automatic to the user by supplying the proper subroutine command.

The subroutine, which requires 3K, should be linkage edited with the calling program.

2. USING PUTTMT

2.1 Environment Division.

A select clause should not exist for output files processed by the subroutine.

2.2 Data Division.

File Section.

FD information should not exist for output files processed by the subroutine.

Working-Storage Section.

The following area should be coded and used as the argument in the CALL statement.

```
01   DATAPASS.
     03  ACTIØN            PIC 99.
     03  REELCNT           PIC 99 CØMP VALUE 0.
     03  CØM               PIC 9.
     03  PASSDATA          PIC X(n).
```

Description of Fields:

2.21 DATAPASS is the argument used in the call statement.

2.22 ACTION will contain a two-digit numeric value upon every return from the subprogram. This unsigned numeric value will be indicative of the function that was performed by the subroutine. The following table summarizes the meanings and the recommended user-program logic for each ACTION.

PUTTMT - FORMATTING FILES FOR TRANSMISSION,(CONT.)

	MEANING(S)	USER ACTION(S)
Codes 00	Invalid COM, must be a 1,2,3,4, or 5 in DISPLAY.	Move desired command to COM and Call Subroutine again.
Open Codes 10	Header record (THD) put on file successfully.	Continue processing.
12	Required header does not have a "THD" in character positions 1-3 of PASSDATA.	Move proper header information to PASSDATA and call subroutine again.
14	Record factor is not a whole multiple of the block factor in the THD.	Move proper header information to PASSDATA and call subroutine again.
16	Invalid function, a "TTLE" was not put on previous file, or trying to put two headers for the same file.	Continue processing or put a "TTLE" on the file.
Write Codes 20	Action of put record complete and successful.	Continue processing.
26	Invalid function, a "THD" was not put on the file.	First move header information to PASSDATA and call subroutine to put header, then continue by calling subroutine to put the record.
Close Codes 30	Trailer record (TTLE) on file successfully.	Continue processing.
36	Invalid function a "THD" was not put on the file, or trying to put two trailers for the same file.	Continue processing or put a "THD" on the file.

PUTTMT - FORMATTING FILES FOR TRANSMISSION, (CONT.)

2.23 REELCNT, a computational field, indicates the number of output reels that have been processed or the current number of output reels. For example, if REELCNT=5 at the end of output processing, this indicates there are 5 output reels; whereas, a REELCNT=5 during processing, indicates that the fifth reel is accepting the output.

2.24 COM (command) is an unsigned numeric field valued 1 through 5. The subroutine interrogates this field to determine which of the following functions it should perform:

VALUE	FUNCTION
1	Put the header (THD) on the file
2	Write a record
3	Put the trailer (TTLE) on the file without volume switching. This implies stacked data sets per reel.
4	Put the trailer (TTLE) on the file with volume switching.
5	Close the last data set. This must be the final command issued to the subroutine.

Note: If the command is not one of the above unsigned-numerics a "00" in the ACTION field will be returned. (See table under ACTION 2.22)

2.25 PASSDATA PIC X(n) is the area where the header and all logical records must be moved. The subroutine writes from this area; thus, information which is relevant to the command being issued, must exist in this area.

The "PIC" size, "n", must be the size of the largest logical record with two restrictions. It must not be less than 80, which is the size of the labels; and secondly, it cannot be greater than 1000, which is the maximum size of any record as specified in C.I. 95.183.

When COM(command=1) is issued, the subroutine expects the user header to exist in this field; and when COM(command)=2, the subroutine expects a logical record in this field. On all other commands, the subroutine ignores the content of the field.

PUTTMT - FORMATTING FILES FOR TRANSMISSION, (CONT.)

2.3 Procedure Division.

The subroutine is invoked by the following statement:

```
CALL 'PUTTMT' USING DATAPASS.
```

2.4 JCL

PUTTMT expects all output files to be described by a DD statement with the DDNAME of TAPETMT.

For example:

```
//TAPETMT  DD  DSN=ANY,DISP=(NEW,KEEP),UNIT=TAPE,LABEL=(,NL),
//  VØL=SER=(VØL1,VØL2,VØL3,.........voln)
```

Where "n" represents the maximum number of output volumes.

WESNAP - CALLABLE DUMP FOR DEBUGGING

1. GENERAL

WESNAP is a subroutine which produces a "SYSUDUMP" type core dump each time it is called during the execution of a program. It is a useful debugging aid in that a programmer may place calls for the dump at logical points in a program to capture conditions at that time.

2. USING WESNAP

2.1 The routine is invoked in the procedure division of a COBOL program with a simple call statement.

```
CALL 'WESNAP'.
```

2.2 A dd statement is required in the step executing the COBOL program as follows:

```
//SNAP DD SYSØUT=A
```

CALENDAR - CONVERT CALENDAR DATE TO JULIAN DATE

1. GENERAL

CALENDAR is a subroutine of 286 bytes that may be used to convert a six digit calendar date (in the form of YYMMDD) to Julian date (YYDDD).

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
01  DATES.
    03  JULIAN-DATE             PIC S9(5).
    03  CALENDAR-DATE.
        05  CAL-DATE-YR         PIC 99.
        05  CAL-DATE-MØ         PIC 99.
        05  CAL-DATE-DA         PIC S99.

01  USER-DATE                   PIC S9(6)   CØMP-3.
```

2.2 Procedure Division.

CALENDAR is executed through a CALL statement coded as follows:

```
CALL 'CALENDAR' USING CALENDAR-DATE, USER-DATE.
```

3. REFERENCE

Responsible programmer, J. L. Halley - Oklahoma City.

CALENDAR - CONVERT CALENDAR DATE TO JULIAN DATE

1. GENERAL

CALENDAR is a subroutine of 286 bytes that may be used to convert a six digit calendar date (in the form of YYMMDD) to Julian date (YYDDD).

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
01  DATES.
    03  JULIAN-DATE             PIC S9(5).
    03  CALENDAR-DATE.
        05  CAL-DATE-YR         PIC 99.
        05  CAL-DATE-MØ         PIC 99.
        05  CAL-DATE-DA         PIC S99.

01  USER-DATE                   PIC S9(6)   CØMP-3.
```

2.2 Procedure Division.

CALENDAR is executed as follows:

```
MOVE (six digit calendar date (YYMMDD))  TO CALENDAR-DATE.
CALL 'CALENDAR' USING CALENDAR-DATE, USER-DATE.
MOVE USER-DATE TO JULIAN-DATE.
```

3. REFERENCE

Responsible programmer, J. L. Halley - Oklahoma City

FISCALPD - CONVERT SIX DIGIT FISCAL DATE TO JULIAN DATE

1. GENERAL

FISCALPD is a subroutine of 1884 bytes that may be used to convert a six digit fiscal date (in the form of YYMMWD) to Julian Date (YYDDD).

2. CODING REQUIREMENTS

2.1 Data Division.

Working-Storage Section.

```
01  DATES.
    03  JULIAN-DATE              PIC S9(5).
    03  CALENDAR-DATE            PIC S9(6).
    03  FISCAL-PERIØD.
        05  FIS-PERIØD-YR        PIC 99.
        05  FIS-PERIØD-MØ        PIC 99.
        05  FIS-PERIØD-WK        PIC 9.
        05  FIS-PERIØD-DA        PIC S9.

01  USER-DATE                    PIC S9(6)   CØMP-3.
```

2.2 Procedure Division.

FISCALPD is executed through a CALL statement coded as follows:

```
CALL 'FISCALPD' USING FISCAL-PERIØD, USER-DATE.
```

CAUTION: If the D portion of the fiscal date provided by the calling program is zero or blank, the value returned to the calling program will be the first day of that week.

3. REFERENCE

Responsible programmer, J. L. Halley - Oklahoma City.

FISCALWK - CONVERT FIVE DIGIT FISCAL DATE TO JULIAN DATE

1. GENERAL

FISCALWK is a subroutine of 1826 bytes that may be used to convert a five digit fiscal date (in the form of YYWWD) to Julian date (YYDDD).

2. CODING REQUIREMENTS

2.1 Date Division.

Working-Storage Section.

```
01   DATES.
     03   JULIAN-DATE           PIC S9(5).
     03   CALENDAR-DATE         PIC S9(6).
     03   FISCAL-PERIØD         PIC S9(6).
     03   FISCAL-WEEK.
          05   FIS-WEEK-YR      PIC 99.
          05   FIS-WEEK-WK      PIC 99.
          05   FIS-WEEK-DA      PIC S9.

01   USER-DATE                  PIC S9(6)   CØMP-3.
```

2.2 Procedure Division.

FISCALWK is executed through a CALL statement coded as follows:

```
CALL 'FISCALWK' USING FISCAL-WEEK, USER-DATE.
```

CAUTION: If the D portion of the fiscal date provided by the calling program is zero or blank, the value returned to the calling program will be the first day of that week.

3. REFERENCE

Responsible programmer, J. L. Halley - Oklahoma City.

RANDOM - A PSEUDO-RANDOM NUMBER GENERATOR

1. GENERAL

1.1 RANDOM is a multiple-entry subprogram which returns statistically random numbers to the calling program. The intended application area is in Monte-Carlo Simulations. The statistical properties are far superior to those of most commonly used generators, yet the execution time is generally about the same. It can be invoked from COBOL, FORTRAN, or PL/I programs, and will provide either uniform or normal random numbers.

1.2 Uniform numbers will be in the range from 0 to 1, and are returned in floating point format. Normal numbers have a mean of 0 and a standard deviation of 1. Normal numbers within the range of -6 sigma to +6 sigma are statistically sound. The starting value of the generator can be controlled by the user for debugging purposes, or the user can let the subprogram generate its own random starting value. The generator has a cycle length of 2.14 billion for uniform numbers, and a length of 44.7 million for normal numbers.

2. CODING REQUIREMENTS

2.1 COBOL CODING

```
Data Division.
Working-Storage Section.
77 RANDOM-NUMBER USAGE COMP-3.
              .
              .
Procedure Division.
MOVE X.XXX TO RANDOM-NUMBER.
CALL SEEDIT USING RANDOM-NUMBER.
              .
              .
CALL RANDOM USING RANDOM-NUMBER.
       and/or
CALL NORAND USING RANDOM-NUMBER.
```

RANDOM - A PSEUDO-RANDOM NUMBER GENERATOR (CONT.)

2.2 FORTRAN CODING

```
REAL *8 RANUMB ✓
RANUMB=X.XXX
CALL SEEDIT (RANUMB)
CALL RANDOM (RANUMB) ✓
   ~~or ANUMB=FRAND(RANUMB)~~
     and/or
CALL NORAND(RANUMB)
   or ANUMB=FNRAND(RANUMB)
```

2.3 PL/I CODING (Checkout & Optimizing Compiler Only)

```
DECLARE RANDOM_NUMBER DECIMAL FLOAT(a);

RANDOM_NUMBER=X.XXX
CALL SEEDIT (RANDOM_NUMBER)
CALL RANDOM (RANDOM_NUMBER)
    and/or
CALL NORAND (RANDOM_NUMBER)
```

2.4 Coding Notes

2.41 The variable used to pass data to the external seeding routine "SEEDIT" does not have to be the same as the argument of the other call statements. It was done in the examples simply for coding brevity.

2.42 The "FRAND" and "FNRAND" entries for FORTRAN do not have an argument. However, the compiler requires a dummy argument to identify it as a function subprogram.

2.43 The use of a double word (8 bytes) argument is optional. However, to get consistent results from program to program, the argument for entry "SEEDIT" must be a double word.

3. GENERATOR USAGE

3.1 Entry "SEEDIT" - This call statement is used to obtain a random number sequence starting with the user specified value. The argument, which is the starting value, should be a double word floating point number in order to ensure repeatability. It should be in the range of 0 to 1; if not an arbitrary constant is substituted and execution continues.

RANDOM - A PSEUDO-RANDOM NUMBER GENERATOR (CONT.)

3.2 Entry "RANDOM" - This call statement returns to the user a single word (4 byte) floating point uniform random number in the argument specified. The number is between 0 and 1.

3.3 Entry "FRAND" - This function subprogram entry returns a uniform random number to the FORTRAN user the same as "RANDOM". It should not be called from other languages.

3.4 Entry "NORAND" - This call statement returns to the user a single word (4 byte) floating point normal random number in the argument specified.

3.5 Entry "FNRAND" - This function subprogram entry returns a normal random number to the FORTRAN user the same as "NORAND". It should not be invoked from COBOL or PL/I.

4. REFERENCE

4.1 Please refer any questions to W. A. Bell - 9411, on CORNET 391-5257.

DATA PROCESSING SYSTEM PROGRAMMING AIDS

MACROS - WESTERN ELECTRIC

SECTION 1 - GENERAL

1. INTRODUCTION

1.1 A macro instruction provides a convenient way to generate a desired sequence of assembler language statements in one or more programs. The macro definition is written only once, and a single statement, a macro instruction statement, is written each time a programmer wants to generate the desired sequence of statements. The assembler generates a sequence of assembler language statements for each occurrence of the same macro instruction. The generated statements are then processed like other assembler language statements.

1.2 The macro instructions presented in this Section were developed by Western Electric personnel to simplify the coding of instructions which perform a specific function or special purpose. The macros reside in SYS1.MACLIB.

1.3 Appendix A, Macro Reference List, directs the user to an Appendix which contains a description of the macro and examples of usage.

1.4 For a complete description of assembler language macros and an explanation on how to use macros, refer to IBM System/360 Operating System Assembler Language Manual, Form C28-6514.

DATA PROCESSING SYSTEM PROGRAMMING AIDS

MACROS - WESTERN ELECTRIC

MACRO REFERENCE LIST

DEBUG MACRO

1. GENERAL

1.1 The DEBUG macro provides printouts of the general purpose registers, selected areas of core and an indication of the calling point in the using program without modifying the environment in which the DEBUG routine is operating.

2. FEATURES

2.1 The following features and capabilities are provided by the DEBUG macro:

2.11 Ability to specify a register for both the core location from which to start dumping and the length of the area.

2.12 Ability to specify a length on storage dumping up to 4095 bytes.

2.13 Translation of storage to EBCDIC.

2.14 Ability to turn printing on or off under program control.

2.15 The address is affixed to each line of storage printed.

2.16 The processing module is generated as a named CSECT so that only about 150 bytes of addressable storage are needed.

3. USAGE

3.1 Control of the DEBUG macro is accomplished through the use of up to four operands and one DD card;

```
DEBUG  message,from,length,DDNAME=
```

3.11 message - The first positional operand is a message to be printed with the register and core contents to indicate the calling point. The message may be up to 12 characters long and must be enclosed in apostrophes if embedded blanks are present. The apostrophe and ampersand characters must be doubled as in normal DCs. Messages longer than 12 characters will be truncated to the first 12. Shorter messages will be padded to the right with blanks. If this operand is omitted, the default used is the CSECT name concatenated with an integer indicating the relative position of the DEBUG macro.

3.12 from - The second positional operand is the label or a pointer in a register of the beginning address of the storage area to be printed. If a label is used, it may be any label based on any USING statement. This includes labels in DSECTs. If register notation is used, it must be of the form: DISPL(REG) where either or both of DISPL and REG may be equates or numbers. Examples: 4(6), ZERO(R8). If this address is not a doubleword it is rounded down to the nearest doubleword. This operand may be omitted if no storage dumping is desired.

3.13 length - The third positional operand need be specified only when the second operand, "from," is specified. The third operand specifies the number of bytes to be printed. It may be a fixed number or in register notation and must not exceed 4095 bytes. If register notation is used, the form is the same as described for the second operand in 3.12 above. The displacement and the contents of the register will be added together and used for a length. If the length is zero or negative, the default value of 128 bytes will be used. If the length is greater than 4095 bytes, it will be reset to 4095. If the length is omitted, and a second operand (from) has been supplied, 128 bytes will be assumed.

3.14 DDNAME=ddname - This keyword operand is required only on the first use of the DEBUG macro in an assembly (meaning the first physical use, not necessarily the first logical use). Specify the DDNAME of a DD card which DEBUG may use for output to be printed. This DDNAME must be unique in your JCL run deck. User must supply a DD card in his execution step specifying this DDNAME. If omitted, an MNOTE will be issued and the CSECT name will be assumed.

4. THE ON-OFF FEATURE

4.1 The single byte at the generated label 'ZZXØNØFF' is the key to program control of DEBUG printing. If its value is X'00' (as it is set initially), normal DEBUG printing will occur. Whenever the byte has a non-zero value, DEBUG printing for this module will be suppressed. The user is free to change this one byte as often as desired.

5. ADVANTAGES OF DEBUG OVER SNAP DUMPS

5.1 Easier to use

5.2 Does not skip a page on output

5.3 Does not change any registers

5.4 Does not change the condition code

5.5 Supplies its own processing module and DCB

5.6 OPENs its own file

5.7 Provides a message indicating macro location in program in S-CON form (i.e., base-displacement)

5.8 Prints CSECT name of program

5.9 All registers are printed automatically on every call

5.10 Ability to turn off and on under program control

6. DEBUG DISADVANTAGES AND RESTRICTIONS

6.1 The first physical use of DEBUG requires 156 bytes of main storage addressable by the user's base register and approximately 950 bytes in a separate CSECT. Subsequent use requires only 25 bytes of addressable storage. Four more bytes are required if a label for storage dumping is specified.

6.2 Must be used within a named CSECT (provided by ENTER macro operand CSECT=).

6.3 May not be used in read-only storage since it modifies itself.

6.4 The first use of the DEBUG macro must be addressable by other uses of it in an assembly. This is not a problem if basing for the module is set up and not changed.

6.5 The first 18 words of the save area based on register 13 must not be specifically used by the user except for use as a standard save area.

ENTER MACRO

1. GENERAL

The ENTER macro generates the instructions required at the beginning of an assembly language program.

2. USAGE

2.1 Simple form - The macro may be coded without any parameters:

```
ENTER
```

This produces an unnamed control section, saves all the general registers, establishes register 12 as a base register and creates a save-area to which register 13 points.

An entry point name is not generated. Consequently, this version of the macro is suitable only in a main program.

2.2 General form - Loc

```
Loc      ENTER  CSECT=csname,
                BASE=basereg,
                SAVNAME=savarea
```

All the parameters are optional. The operands are keyword parameters and may be coded in any order. Use commas where necessary to separate operands.

2.21 loc - A symbol coded in the name field of the ENTER macro defines the entry point to the program. An ENTRY is generated for the symbol, and the symbol is placed in the name field of the first program instruction.

If this parameter is omitted, an ENTRY statement is not generated and the name field of the first program instruction is left blank.

2.22 CSECT=csname - This parameter gives a name to the control section generated by the ENTER macro. If the parameter is omitted, the control section is considered unnamed.

2.23 SAVNAME=savarea - This parameter gives a name to the 72 byte save area. Unless it is necessary to refer to the save area by name, this parameter may be omitted since the address is loaded into register 13.

2.24 BASE=basereg - This parameter may be used to assign a base register. If it is omitted, register 12 is assumed. Registers 0 and 13 may not be specified.

```
2.3  Complete form - loc     ENTER  USE=RENT
                                    LV=length
                                    SP=pool
                                    CSECT=(csname,start)
                                    BASE=basereg
                                    REG=(r1,r2)
                                    ID=* or ID=(ident)
                                    SAVNAME=savarea
                                    SYMREG=Y
```

2.31 The parameters 'loc', BASE=basereg, and SAVNAME=savarea are the same as described under General Form, paragraph 2.2.

2.32 USE=RENT - This parameter makes the code generated by the ENTER macro reentrant. Instead of generating a 72 byte save-area, a GETMAIN supervisor call is issued to dynamically allocate a 72 byte save-area.

2.33 LV=length and SP=pool - LV and SP may be coded in conjunction with USE=RENT. These parameters correspond to the LV and SP parameters in the R type GETMAIN which specify the subpool number the length in bytes of the requested main storage area. LV=72 and SP=0 are assumed if not specified. No less than 72 bytes should be requested since the first 72 bytes are used for the register save area.

2.34 CSECT=(csname,start) - The parameter csname is the same as described under General Form, paragraph 2.2. The parameter 'start' may be used to specify an initial location counter value for the program. If coded, it must be a self-defining term. A START instruction rather than a CSECT will then be generated. This may only be done for the first (or only) control section for a program.

2.35 REG=(r,(,r2)) - Normally all the registers are saved. The REG parameter may be coded so that only certain registers are saved. The registers must be specified so that they are stored in the order 14, 15, 0, ..., 11, 12 when used in a STM instruction. Both r1 and r2 must be self-defining terms, and must not designate register 13. If USE=RENT is coded, all registers are saved regardless of the REG parameter.

2.36 ID=* or ID=(ident) - Some applications require an EBCDIC character string at the beginning of the program. The ID parameter specifies an identifier which will be assmbled into such a character string.

If ID=* is coded, the identifier is taken from the name field of the ENTER macro. If the name field is blank, the name of the control section will be used. If the control section is unnamed and the name field is blank, no character string will be assembled. If a different identifier is desired it may be explicitly coded. For instance,

```
ID=OTHER
```

This name may be up to 70 characters long. If the name includes blanks or commas, it should be enclosed in single quotes.

When ID is coded, the first machine instruction generated by the macro will be a branch around the assembled character string. Following the branch instruction and preceding the character string is a one-byte length field indicating the number of characters in the string.

For example:

```
ENTER   ID=OTHER
```

would generate

```
        .
        .
        .
        B     10(0,15)      BRANCH AROUND ID
        DC    AL1(5)        LENGTH OF IDENTIFIER
        DC    CL5'OTHER'    IDENTIFIER
```

In all cases, the length field and the character string are displaced 4 bytes and 5 bytes respectively from the program entry point.

2.37 SYMREG=Y - If SYMREG=Y is coded, symbolic register equates, R0, R1, R2, ..., R15, are generated.

3. CODING DETAILS

3.1 If USE=RENT is not coded, an in-line save-area is created such that the address of the save-area is also the base address.

Thus, additional base registers may be established as follows:

```
         ENTER              SAVNAME=BASE
         LM                 10,11,NEWBASE
         USING              BASE+4096,10,11
         B                  NEXT
NEWBASE  DC                 A(BASE+4096,BASE+8192)
NEXT     ...
```

Registers 10,11, and 12 are established as base registers.

It is not permitted to specify BASE=13 in the macro. Although it is not recommended, register 13 may still be used as a base register as follows:

```
ENTER         SAVNAME=BASE
USING         BASE,13
DROP          12
```

3.2 The ENTER macro assumes that the program was entered by a standard calling sequence. This requires that register 13 contain the address of the caller's save-area and that register 15 contains the address of the program entry point. This will always be true for a main program, which is called by the system, or for programs entered using system-linkage macros (i.e., CALL,LINK).

3.3 The ENTER macro does not alter the contents of the registers other than the base register and register 13, the save-area register.

3.4 An MNOTE error message is issued if either BASE=0 or BASE=13 is coded. An MNOTE warning message is issued if any of registers 1, 2, 14, or 15 is coded in the BASE parameter.

3.5 The ENTER macro is intended to be complementary to the EXIT macro. If USE=RENT is specified in either macro, it should be specified in both. If the REG parameter is used to save only certain registers, it should be used similarly in the EXIT macro to restore only those registers. The LV or SP operands when specified in the ENTER macro should be coded identically in the EXIT macro.

3.6 All combinations of the parameters are legal.

EXIT MACRO

1. GENERAL

The EXIT macro generates the instructions required to restore the general registers and return control to the calling program. For a main program, this amounts to stopping execution and returning control to the system.

2. USAGE

2.1 Simple form - The macro may be coded without any parameters.

```
                    EXIT
```

This will restore the general registers and return control to the calling program.

```
2.2 General form - loc    EXIT   RES=result,
                                 RC=retcode
```

All the parameters are optional. The operands are keyword parameters and may be coded in any order. Use commas where necessary to separate operands.

2.21 loc - This symbol is placed in the name field of the first instruction generated by the macro. When control is to be returned, a branch to this address may be executed. If this parameter is omitted, the name field of the first instruction is left blank.

2.22 RES=result - This parameter allows an integer value to be returned to the calling program in register 0. See paragraph 3 for valid forms of this parameter.

2.23 RC=retcode - This parameter allows a return code to be returned to the calling program in register 15. See paragraph 3 for valid forms of this parameter.

2.3 Complete form -

```
loc    EXIT    USE=RENT,
               LV=length,
               SP=pool,
               RES=result,
               RC=retcode,
               REG=(r1,r2)
```

2.31 The parameters 'loc', RES=result, and RC=retcode are the same as described under General form, paragraph 2.2.

2.32 USE=RENT - Coding this parameter generates a FREEMAIN supervisor call which releases the current 72 byte save-area.

2.33 LV=length and SP=pool - LV and SP may be coded in conjunction with USE=RENT. These parameters correspond to the LV and SP parameters in the R type FREEMAIN which specify the subpool number and the length in bytes of the main storage area to be released. LV=72 and SP=0 are assumed if not specified.

2.34 REG=(r1(,r2)) - Normally, all the registers are restored. The REG parameter may be coded so that only certain registers are restored. The registers must be specified so that they are restored in the order 14, 15, 0, ..., 11, 12 when used in a LM instruction. Both r1 and r2 must be self-defining terms and must not designate register 13. If USE=RENT is specified, all the registers are restored. Whenever the RC parameter is coded, register 15 will not be restored, but will be loaded with the designated value. Similarly if the RES parameter is coded, register 0 will not be restored, but will be loaded with its designated value. These conditions also apply when USE=RENT is specified.

3. VALID FORMS OF THE RC AND RES PARAMETERS

3.1 Absolute Form - This is written as a self-defining term whose value is less than 4096.

```
EXIT   RC=8         (or RES=8)
```

3.2 Register Form - if the value is in a general register, the register number should be enclosed in parenthesis.

```
EXIT   RC=(10)      (or RES=(10) )
```

3.3 Symbolic Form - A symbol defining a full-word or a half-word value may be designated.

```
EXIT   RC=CONST     (or RES=CONST)
```

Where CONST might be defined as:

```
CONST  DC  F'12'

       or

CONST  DC  H'12'
```

3.4 Literal Form - A full-word or a half-word literal may be designated:

```
EXIT    RC==F'16'          (or RES==F'16')

        or

EXIT    RC==H'20'          (or RES==H'20')
```

4. CODING DETAILS

4.1 It is not recommended that RC=(0) be coded when RES is used or that RES=(15) when RC is used. The resulting code is necessarily inefficient since a register must temporarily be stored in order to load the proper value.

4.2 The EXIT macro assumes that a 72 byte save-area has been provided for the program and that the address of this area is in register 13. This will be the case if an ENTER macro was executed at the beginning of the program.

4.3 If the REG parameter is specified so that register 14 will not be restored, then this register should not have been modified by the program, since the EXIT macro will use register 14 to return control.

4.4 The EXIT macro is intended to be complementary to the ENTER macro. If USE=RENT is specified in either macro, it should be specified in both. If the REG parameter is used to restore only certain registers, it should be used similarly in the ENTER macro so that these registers are saved. The LV or SP operands when specified in the ENTER macro should be coded identically in the EXIT macro.

4.5 All combinations of the parameters are legal.

DATA PROCESSING SYSTEM PROGRAMMING AIDS

CATALOGED PROCEDURES

SECTION 1 - GENERAL

1. INTRODUCTION

1.1 A cataloged procedure is a set of job control statements that has been placed in a partitioned data set called the procedure library. The procedure can be retrieved from the library by using its member name in an EXEC statement of a job step in the input stream. It can contain statements for the processing of an entire job, or it can contain statements to process one or more steps of a job, with the remaining steps defined by job control statements in the input stream. A job can use several cataloged procedures, each processing one or more of the job steps. A job can also call for execution of the same cataloged procedure in more than one job step.

1.2 In general, this material should enable the user to make effective use of these cataloged procedures. For those procedures which invoke IBM programs or other commercial software products, reference is made to the applicable vendors manual or reference guide where additional information is available to the user.

1.3 Appendix A, Cataloged Procedure Reference List, directs the user to an Appendix which contains a description of the cataloged procedure and examples of usage.

CATALOGED PROCEDURE REFERENCE LIST

PROCEDURE	FUNCTION	APPENDIX	PAGE
ALGOFC	ALGOL compile	B	6
ALGOFCG	ALGOL compile and execute	C	7
ALGOFCL	ALGOL compile and linkage edit	D	9
ALGOFCLG	ALGOL compile, linkage edit and execute	E	11
ASMFC	ASSEMBLER (F) compile	F	13
ASMFCG	ASSEMBLER (F) compile and execute	G	14
ASMFCL	ASSEMBLER (F) compile and linkage edit	H	16
ASMFCLG	ASSEMBLER (F) compile, linkage edit and execute	I	18
AUTOFLOW	Source program documentation	J	20
CATALIST	List catalog node points	BZ	123
CATAUTIL	Catalog maintenance	CA	125
CHGPDS	Adding, replacing, changing card image PDS members	K	23
COBLG	COBOL linkage edit and execute	L	25
COB4C	ANS COBOL compile	M	26
COB4CG	ANS COBOL compile and execute	N	27
COB4CL	ANS COBOL compile and linkage edit	O	29
COB4CLG	ANS COBOL compile, linkage edit and execute	P	31
COB4COL	ANS COBOL compile, optimize, and link-edit	BA	85
DADUMP	Direct access file dump listing	AT	75

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
DASORT	Direct access utility sort	Q	33
DASPACE	Calculate direct access space usage for a logical record of given length	BC	88
DPMPSIL	List contents of the PSI table	BF	93
FORTGC/ FORTHC	FORTRAN (G & H) compile	R	35
FORTGCL/ FORTHCL	FORTRAN (G & H) compile and linkage edit	S	36
FORTGCLG/ FORTHCLG	FORTRAN (G & H) compile, linkage edit and execute	T	38
FORTLG	FORTRAN linkage edit and execute	U	40
LIBP	LIBRARIAN, PDS output	V	41
LIBS	LIBRARIAN, sequential output	W	43
LIBU	LIBRARIAN utility procedure	X	45
LIST	Listing VTOC, PDS directories catalogs	Y	47
LISTDISK	Listing contents of a direct access volume and attributes of direct access data sets	Z	48
LISTOUT	Listing a member of a card image partitioned data set	AA	49
LISTPDS	Listing a partitioned data set	AB	50
LKED	Link edit and produce a load module	AC	51
LKEDG	Link edit, produce a load module, and execute	AD	52
META	METACOBOL pre-compiler processor (includes COBOL F to ANS COBOL Conversion)	BB	86

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
MOD	Catalog operations, renaming, scratching	AE	53
MOVE	Copying direct access files, catalogs, PDS members	AF	54
OPRPNCH	Obtain copy of console JCL decks	AG	56
PLCPREP	Preparation for update of production libraries	AH	57
PLICKC	PL/I checkout compile	BM	107
PLICKCG	PL/I checkout compile, load, and go	BN	108
PLICKCL	PL/I checkout compile and link	BO	110
PLICKCLG	PL/I checkout compile, link-edit and go	BP	111
PLICKG	PL/I checkout load and go	BQ	113
PLICKI	PL/I checkout and go	BR	114
PLICKLG	PL/I checkout load and go	BS	115
PLICKR	PL/I checkout compile	BT	117
PLIXC	PL/I optimizing compile	BU	118
PLIXCG	PL/I optimizing compile and go	BV	119
PLIXCL	PL/I optimizing compile and link-edit	BW	120
PLIXCLG	PL/I optimizing compile, link-edit and go	BX	121
PLIXLG	PL/I optimizing link-edit and go	BY	122
PL1LFC	PL/I compile	AI	59
PL1LFCL	PL/I compile and linkage edit	AJ	60

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
PL1LFCLG	PL/I compile, linkage edit, and execute	AK	62
PL1LFLG	PL/I linkage edit and execute	AL	64
PPANAL	Problem Program Efficiency-Analyzer	AV	79
PPEXT	Problem Program Efficiency-EXTRACTOR	AW	81
REACT	Data retrieval system to retrieve data from sequential and index sequential files producing up to 40 formatted output reports	BD	89
RPGEC	RPG compile	AM	65
RPGECL	RPG compile and linkage edit	AN	66
RPGECLG	RPG compile, linkage edit, and execute	AO	68
SORT	Sort/merge with routines that require link editing	AX	83
SORTD	Sort/Merge with no routines or routines that do not require link editing	AY	84
SORTSPAC	Calculate disk sort work space	AP	70
TLSBINL	List contents of Tape Library System bin table	BE	92
TPDUMP	Tape dump listing	AU	77
TRMDUMMY	Create a "dummy" disk generation data set	BI	99
TRMHIST1	Extract backup volume information from Mini-WEDGE History files	BJ	101
TRMRECVR	Transmission file recovery	BK	103

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
TRMSTKAA	Copy application system outgoing transmission data set to STACK disk pack	BL	105
VSBASIC	Translate and run a BASIC source module	CB	127
WECDSK	Card input to disk	BG	94
WECT	Card input to standard label tape with cataloging	AQ	71
WELIST	Listing VTOC, PDS directories, catalogs	AR	73
WEMOD	Catalog operations, renaming, scratching	AS	74
WEOUTPUT	Summary listing of production system output which requires further processing	Div. 2, Ch. 4, Sect. 5	
WETPCOPY	Copy an entire tape to the double end-of-file mark	BH	97

TRMDUMMY - CREATE A "DUMMY" DISK GENERATION DATA SET

1. GENERAL

1.1 TRMDUMMY is a cataloged procedure developed for application systems interfacing with Mini-WEDGE. The function of this cataloged procedure is to create a "dummy" transmission data set on disk and make an entry in the system catalog for the "dummy" data set to avoid an empty GDG condition (no cataloged generations) which would cause job failure.

2. SYMBOLIC PARAMETERS

2.1 TRMDUMMY requires one symbolic parameter and may have three optional parameters, which are:

GDS, a required parameter, specifies the generation data group index (or base).

REG, an optional parameter, specifies the maximum region. The default is 10K. Ordinarily this default will not need to be changed.

VOLSER, an optional parameter, specifies the volume serial of the output disk pack. The default is the UNSTACK disk pack TRM002.

BLKSIZE, an optional parameter, specifies the DCB BLKSIZE value. The default is 1000.

3. ASP CONTROL CARD

3.1 All jobs using TRMDUMMY for systems interfacing with Mini-WEDGE must include an ASP //*MAIN statement which includes the CLASS=UNSTACK and FAILURE=CANCEL parameters.

4. USING TRMDUMMY

4.1 TRMDUMMY is invoked by an EXEC statement

```
// EXEC TRMDUMMY,GDS='GDSindex'
```

(Contined on next page)

TRMDUMMY (CONTD.)

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMDUMMY
//TRMDUMMY PROC GDS='GDS.DSNAME.BASE',                                  X00000100
//               VOLSER=TRM002,BLKSIZE=1000,                            X00000200
//               REG=10K                                                 00000300
//*              GENERATE A DUMMY GDS(+1) IF A GDS(0) DOES NOT ALREADY   00000400
//*              EXIST                                                   00000500
//*                                                                      00000600
//*              SYMBOLIC PARAMETER &GDS MUST SPECIFY THE GDG DATA SET   00000700
//*              NAME BASE                                               00000800
//STEP1   EXEC   PGM=TRMUST06,PARM='&GDS',REGION=&REG                    00000900
//STEPLIB   DD   DSNAME=PRD2.LOADLIB,DISP=SHR                            00001000
//STEP2   EXEC   PGM=WECOPY,COND=(0,EQ,STEP1),REGION=&REG                00001100
//SYSPRINT  DD   SYSOUT=A                                                00001200
//SYSUT1    DD   DUMMY,DCB=(RECFM=U,BLKSIZE=1000,LRECL=1000)             00001300
//SYSUT2    DD   DSNAME=&GDS.(+1),DISP=(NEW,CATLG,DELETE),SPACE=(TRK,1), X00001400
//               VOLUME=SER=&VOLSER,UNIT=DISK,                          X00001500
//               DCB=(SYS2.DSCB,RECFM=U,BLKSIZE=&BLKSIZE)                00001600
//LASTSTEP EXEC PGM=IEFBR14,REGION=4K                                    00001700
```

6. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

TRMHIST1 - EXTRACT BACKUP VOLUME INFORMATION FROM MINI-WEDGE HISTORY FILES

1. GENERAL

1.1 TRMHIST1 is a cataloged procedure developed for application systems interfacing with Mini-WEDGE. The function of this cataloged procedure is to search the STACK and UNSTACK History files for backup volume serial information pertaining to previous transmissions in the WEDGE system. This information may be used to recover transmission files if the need arises.

2. USING TRMHIST1

2.1 TRMHIST1 is invoked by an EXEC statement and an override card as follows:

```
//  EXEC  TRMHIST1
//STEPO.SØRTIN  DD  *
  (Data Cards - See Div. 2, Chap. 8, Sect. 2, Par. 5.2)
```

2.2 If no input data is submitted, an ABEND U0404 will occur with the message 'NO INPUT SUBMITTED'.

2.3 If the //STEPO.SØRTIN DD * override card is not used, an ABEND U0016 will occur with the message 'IGH043A - DATA SET ATTRIBUTES ERROR'.

3. EXAMPLE OF USE

3.1 Locate backup volume serial numbers from the STACK and UNSTACK History files.

```
//  EXEC  TRMHIST1
//STEPO.SØRTIN  DD  *
SØCSPSSUMHWTMHWPH740127
UØCRØCHDRARAMØCØC740066
```

(continued on next page)

TRMHIST1 (CONTD.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMHIST1
//TRMHIST1 PROC CURCHG=7400174001,LOC=PRDG2,R160=6400,
//        SWA=0500,SRTCTL=PRD2
//STEPO    EXEC PGM=SORT,REGION=110K
//SYSUDUMP  DD SYSOUT=A
//SORTLIB   DD  DSNAME=SYS1.SORTLIB,DISP=SHR
//SYSOUT    DD SYSOUT=A
//*
//*  INPUT IS SUBMITTED VIA AN OVERRIDE     //STEPO.SORTIN DD *
//*  TYPICAL TRANSACTIONS ARE SHOWN BELOW
//*  COL. 1 S OR U   S=STACK   U=UNSTACK
//*  COL. 2-3  LOCATION ID  EXAMPLE SHOWS  *OC*
//*  COL. 4-13   FILE-ID
//*  COL. 14-23  SERIAL NUMBER
//*  SOCSPSSUMHWTMHWPH740127
//*  UOCRDCHDRARAMOCOC740066
//*
//SORTIN    DD DUMMY
//SORTOUT   DD UNIT=DISK,DSN=&&FINDERS,DISP=(NEW,PASS,DELETE),
//             SPACE=(1600,(200,010),RLSE),
//             DCB=(RECFM=FB,BLKSIZE=1600,LRECL=080)
//SORTWK01  DD UNIT=DISK,SPACE=(TRK,(&SWA,),,CONTIG)
//SORTWK02  DD UNIT=DISK,SPACE=(TRK,(&SWA,),,CONTIG)
//SORTWK03  DD UNIT=DISK,SPACE=(TRK,(&SWA,),,CONTIG)
//SYSIN     DD DISP=SHR,DSN=&SRTCTL..CONTROL(TRMHIST1)
//*  SORT FIELDS=(1,23,CH,A)
//STEP1    EXEC PGM=TRMHST01,REGION=110K,PARM='RM01&CURCHG.',TIME=10
//*
//*  THIS PROG. IS USED TO EXTRACT BACKUP VOLUME SERIAL-S FROM
//*       THE STACK AND UNSTACK HISTORY FILES.
//*
//SYSOUT    DD SYSOUT=A
//FINDERS   DD UNIT=DISK,DSN=&&FINDERS,DISP=(OLD,DELETE,DELETE),
//             DCB=(RECFM=FB,BLKSIZE=1600,LRECL=080)
//STACKMST  DD UNIT=TAPE,
//             DSN=&LOC..TRM.STACK.HISTORY(+0),DISP=OLD,
//             DCB=(SYS2.DSCB,RECFM=FB,BLKSIZE=&R160,LRECL=160)
//UNSTAMST  DD UNIT=TAPE,
//             DSN=&LOC..TRM.UNSTACK.HISTORY(+0),DISP=OLD,
//             DCB=(SYS2.DSCB,RECFM=FB,BLKSIZE=&R160,LRECL=160)
//TGEXTFB   DD SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//CBEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//CSEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//DJEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//HMEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//HWEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//IHEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//LEEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//MGEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//OCEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//OHEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//VCEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)
//WNEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133;RECFM=FB)
//WORKFILE  DD UNIT=DISK,DSN=&WORKFILE,
//             SPACE=(&R160,(200,010),RLSE),
//             DCB=(RECFM=FB,BLKSIZE=&R160,LRECL=160)
```

5. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

TRMRECVR - TRANSMISSION FILE RECOVERY

1. GENERAL

1.1 TRMRECVR is a one-step cataloged procedure that may be used to recover selected transmission files from any data set containing multiple transmission files that conform to the standards of CI 95.183, Section 35. The backup and unmatched data sets of the WDC Mini-WEDGE System are examples of data sets that could be used.

2. USING TRMRECVR

2.1 TRMRECVR is invoked by an EXEC statement coded:

```
// EXEC TRMRECVR,DSN='transmission.dsname'
```

2.2 The following DD statements must be included to execute the procedure:

```
//STEP1.SYSUT2  DD  (parameters describing the recovered data set)
//STEP1.SYSIN  DD  *
   (Select cards describing transmission files to be recovered -
    See Div. 2, Chap. 8, Sect. 2, Par. 6)
```

3. EXAMPLE OF USE

3.1

```
// EXEC TRMRECVR,DSN='PRDG2.TRM.UNSTACK.BACKUP.G0086V00'
//STEP1.SYSUT2  DD  (appropriate parameters)
//STEP1.SYSIN  DD  *
    Select cards
```

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMRECVR
//TRMRECVR PROC  DSN='ERROR'                                           00000100
//*              RECOVERS TRANSMISSION FILES FROM BACKUP OR             00000200
//*              UNMATCHED DATA SETS. THE USER MUST SPECIFY             00000300
//*              THE DSNAME OF THE INPUT FILE IN THE DSN                00000400
//*              PARAMETER OF THE EXEC CARD AND SUPPLY DD               00000500
//*              OVERRIDE CARDS - SYSUT2-FOR THE OUTPUT FILE            00000600
//*                             - SYSIN-FOR CONTROL CARDS IN            00000700
//*                 EXAMPLE-                                            00000800
//*              // EXEC TRMRECVR,DSN='PRDG2.TRM.UNSTACK.BACKUP.G0086V00' 00000900
//*              //STEP1.SYSUT2 DD DSN=PRD2.TRM.RECOVERD,DISP=....     00001000
//*              //STEP1.SYSIN  DD  *                                   00001100
//*                                                                     00001200
//*              EXAMPLE DSNAME OF THE FILE TO BE RECOVERED FROM (INPUT)- 00001300
//*                 PRDG2.TRM.UNSTACK.BACKUP                            00001400
//*                 PRDG2.TRM.UNSTACK.MERGED.BACKUP                     00001500
//*                 PRDG2.TRM.UNSTACK.UNMATCH                           00001600
//*                 PRDC2.TRM.STACK.CUMINPUT                            00001700
//*                                                                     00001800
//STEP1  EXEC   PGM=TRMUST05,REGION=40K                                 00001900
//STEPLIB   DD  DSNAME=PRD2.LOADLIB,DISP=SHR                            00001920
//          DD  DSNAME=DEV06.LOADLIB,DISP=SHR                           00001940
//          DD  DSNAME=DEV08.LOADLIB,DISP=SHR                           00001960
//SYSUDUMP  DD  SYSOUT=A                                                00002000
//RCVRYRPT  DD  SYSOUT=A                                                00002100
//SEARCHIN  DD  DDNAME=SYSUT1                                           00002200
//SYSUT1    DD  DSNAME=&DSN,DISP=SHR                                    00002300
//RCVRYOUT  DD  DDNAME=SYSUT2                                           00002400
//CNTLCRDS  DD  DDNAME=SYSIN                                            00002500
```

TRMRECVR (CONTD.)

5. **REFERENCE**

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

TRMSTKAA - COPY APPLICATION SYSTEM OUTGOING TRANSMISSION DATA SET TO STACK DISK PACK

1. GENERAL

1.1 TRMSTKAA is a cataloged procedure developed for application systems interfacing with Mini-WEDGE. The function of this cataloged procedure is to move outgoing transmission data files from an application system data set onto the STACK disk pack TRM001.

2. SYMBOLIC PARAMETERS

2.1 TRMSTKAA requires two symbolic parameters and may have three additional parameters, which are:

LOC, a required parameter, specifies the source location, for example HW, OC, OH, etc.

INPUT, a required parameter, specifies the data set name of the input data set. The parameter must be enclosed in apostrophes if the name is indexed (contains periods). This data set is assumed to be cataloged and on disk or standard label tape. If these assumptions are not true, appropriate overrides must be entered via a //STEP1.SYSUT1 DD card.

REG, an optional parameter, specifies the maximum REGION size. The default is 20K. Ordinarily this default will not need to be changed.

SPACE, an optional parameter, specifies an estimate of the number of 3330 tracks necessary to hold the data being transferred. The default is 10 tracks, which has a capacity of approximately 110 blocks, 1000 characters in length in the initial allocation. The value specified for SPACE is also used for secondary allocations.

BLKSIZE, an optional parameter, specifies the maximum blocksize of the transmission file being copied. The default is 1000. This parameter should specify the actual maximum blocksize if the file being copied is less than 1000.

3. ASP CONTROL CARD

3.1 All jobs which use TRMSTKAA must use the CLASS parameter on the ASP //*MAIN statement. This parameter must specify the transmission class assigned to the remote location and is of the form CLASS=xxWEDGE where xx is the source location, for example HW, OC, OH, etc.

TRMSTKAA (CONTD.)

4. USING TRMSTKAA

4.1 TRMSTKAA is invoked by an EXEC statement

```
// EXEC TRMSTKAA,LØC=HW,INPUT='IN.DSNAME'
```

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME   TRMSTKAA
//TRMSTKAA PROC REG=20K,                                                X00000100
//              LOC=XX,                 SOURCE LOCATION (SEE LIST BELOW)X00000200
//              SPACE=10,               OUTPUT SIZE ESTIMATE (TRKS)     X00000300
//              BLKSIZE=1000,           MAXIMUM BLKSIZE                 X00000400
//              INPUT='INPUT.DSNAME'    DSNAME OF INPUT DATA SET         00000500
//*             COPY TRANSMISSION FILE FROM APPLICATION DATA SET TO      00000600
//*             LOCATION GDS FOR STACK INPUT                             00000700
//STEP1   EXEC  PGM=WECOPY,REGION=&REG                                   00000800
//SYSPRINT DD   SYSOUT=A                                                 00000900
//SYSUT1    DD  DSNAME=&INPUT,DISP=(OLD,KEEP)                            00001000
//SYSUT2    DD  DSNAME=PROG2.TRM.STACK.INPUT.&LOC.(+1),                 X00001100
//              SPACE=(TRK,(&SPACE,&SPACE),RLSE),                       X00001200
//              UNIT=DISK,VOLUME=SER=TRM001,DISP=(NEW,CATLG,DELETE),    X00001300
//              DCB=(DSORG=PS,RECFM=U,BLKSIZE=&BLKSIZE)                  00001400
//*                                                                      00001500
//*             SAMPLE EXECUTE CARD -                                    00001600
//*             // EXEC TRMSTKAA,LOC=CH,INPUT='PROG19.XMM.TRANTAPE(0)'   00001700
//*                                                                      00001800
//*             INPUT IS ASSUMED TO BE CATALOGED                         00001900
//*             INPUT IS ASSUMED TO BE ON DISK OR STANDARD LABEL TAPE    00002000
//*             IF INPUT ASSUMPTIONS ARE NOT TRUE, APPROPRIATE           00002100
//*             OVERRIDES MUST BE ENTERED VIA A //STEP1.SYSUT1 DD CARD   00002200
//*                                                                      00002300
//*             VALID LOCATIONS -- CB, DJ, HW, LE, MG, OC, OH, & VC      00002400
```

6. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

CATALOGED PROCEDURE REFERENCE LIST

PROCEDURE	FUNCTION	APPENDIX	PAGE
ALGOFC	ALGOL compile	B	6
ALGOFCG	ALGOL compile and execute	C	7
ALGOFCL	ALGOL compile and linkage edit	D	9
ALGOFCLG	ALGOL compile, linkage edit and execute	E	11
ASMFC	ASSEMBLER (F) compile	F	13
ASMFCG	ASSEMBLER (F) compile and execute	G	14
ASMFCL	ASSEMBLER (F) compile and linkage edit	H	16
ASMFCLG	ASSEMBLER (F) compile, linkage edit and execute	I	18
AUTOFLOW	Source program documentation	J	20
CHGPDS	Adding, replacing, changing card image PDS members	K	23
COBLG	COBOL linkage edit and execute	L	25
COBOPTC	ANS COBOL compile and optimize object module	BA	85
COBUC	ANS COBOL compile	M	26
COBUCG	ANS COBOL compile and execute	N	27
COBUCL	ANS COBOL compile and linkage edit	O	29
COBUCLG	ANS COBOL compile, linkage edit and execute	P	31
COBUOPT	See COBOPTC		
DADUMP	Direct access file dump listing	AT	75

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
DASORT	Direct access utility sort	Q	33
DASPACE	Calculate direct access space usage for a logical record of given length	BC	88
DPMPSIL	List contents of the PSI table	BF	93
FORTGC/ FORTHC	FORTRAN (G & H) compile	R	35
FORTGCL/ FORTHCL	FORTRAN (G & H) compile and linkage edit	S	36
FORTGCLG/ FORTHCLG	FORTRAN (G & H) compile, linkage edit and execute	T	38
FORTLG	FORTRAN linkage edit and execute	U	40
LIBP	LIBRARIAN, PDS output	V	41
LIBS	LIBRARIAN, sequential output	W	43
LIBU	LIBRARIAN utility procedure	X	45
LIST	Listing VTOC, PDS directories catalogs	Y	47
LISTDISK	Listing contents of a direct access volume and attributes of direct access data sets	Z	48
LISTOUT	Listing a member of a card image partitioned data set	AA	49
LISTPDS	Listing a partitioned data set	AB	50
LKED	Link edit and produce a load module	AC	51
LKEDG	Link edit, produce a load module, and execute	AD	52

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
META	METACOBOL pre-compiler processor (includes COBOL F to ANS COBOL Conversion)	BB	86
MOD	Catalog operations, renaming, scratching	AE	53
MOVE	Copying direct access files, catalogs, PDS members	AF	54
OPRPNCH	Obtain copy of console JCL decks	AG	56
PLCPREP	Preparation for update of production libraries	AH	57
PL1LFC	PL/I compile	AI	59
PL1LFCL	PL/I compile and linkage edit	AJ	60
PL1LFCLG	PL/I compile, linkage edit, and execute	AK	62
PL1LFLG	PL/I linkage edit and execute	AL	64
PPANAL	Problem Program Efficiency-Analyzer	AV	79
PPEXT	Problem Program Efficiency-Extractor	AW	81
REACT	Data retrieval system to retrieve data from sequential and index sequential files producing up to 40 formatted output reports	BD	89
RPGEC	RPG compile	AM	65
RPGECL	RPG compile and linkage edit	AN	66
RPGECLG	RPG compile, linkage edit, and execute	AO	68
SORT	Sort/merge with routines that require link editing	AX	83

CATALOGED PROCEDURE REFERENCE LIST (CONT.)

PROCEDURE	FUNCTION	APPENDIX	PAGE
SORTD	Sort/Merge with no routines or routines that do not require link editing	AY	84
SORTSPAC	Calculate disk sort work space	AP	70
TLSBINL	List contents of Tape Library System bin table	BE	92
TPDUMP	Tape dump listing	AU	77
TRMDUMMY	Create a "dummy" disk generation data set	BI	99
TRMHIST1	Extract backup volume information from Mini-WEDGE History files	BJ	101
TRMRECVR	Transmission file recovery	BK	103
TRMSTKAA	Copy application system outgoing transmission data set to STACK disk pack	BL	105
WECDSK	Card input to disk	BG	94
WECT	Card input to standard label tape with cataloging	AO	71
WELIST	Listing VTOC, PDS directories, catalogs	AR	73
WEMOD	Catalog operations, renaming, scratching	AS	74
WEOUTPUT	Summary listing of production system output which requires further processing	Div. 2, Ch. 4, Sect. 5	
WETPCOPY	Copy an entire tape to the double end-of-file mark	BH	97

ALGOFC - ALGOL COMPILE

1. GENERAL

ALGOFC is a one-step catalogued procedure to compile an ALGOL source program.

2. USING ALGOFC

2.1 ALGOFC may be invoked by an EXEC statement.

```
// EXEC ALGØFC
```

2.2 The following DD statement must be added to the procedure.

```
//ALGØL.SYSIN  DD  *
  (ALGOL source deck)
       or
//ALGØL.SYSIN  DD (appropriate parameters defining a
                   source input data set)
```

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  ALGOFC
//ALGOL  EXEC PGM=ALGOL,REGION=48K                                         00020000
//SYSPRINT CD SYSOUT=A                                                     00040000
//SYSPUNCH  DD  SYSOUT=P                                                   00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SPACE=(3600,(10,4)),                  *00080000
//           DISP=(MOD,PASS)                                               00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(50,10))                    00120C00
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))         0C140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                    00160000
```

4. REFERENCE

IBM 360 OS ALGOL Programmers' Guide, GC33-4000

ALGOFCG - ALGOL COMPILE AND EXECUTE

1. GENERAL

ALGOFCG is a two-step catalogued procedure to compile and execute an ALGOL source program using the IBM LOADER. Note that there is no linkage edit step, and a load module is not produced.

2. USING ALGOFCG

2.1 ALGOFCG may be invoked by an EXEC statement.

```
// EXEC ALGØFCG
```

2.2 The following DD statement must be added to the procedure.

```
//ALGØL.SYSIN  DD  *
  (ALGOL source deck)
         or
//ALGØL.SYSIN  DD (appropriate parameters defining a
                   source input data set)
```

2.3 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //ALGØL. DD statements and any overriding //GØ. DD statements.

2.4 Programs which require a REGION of more than 56K must increment their requirements by 28K when using the LOADER. The increased requirement should be specified with a region override:

```
// EXEC ALGØFCG,GØ.REGIØN=90K
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  ALGOFCG
//ALGOL EXEC PGM=ALGOL,REGION=48K                                      00020000
//SYSPRINT DD SYSOUT=A                                                 00040000
//SYSPUNCH  DD  SYSOUT=P                                               00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SPACE=(3600,(10,4)),              *00080000
//           DISP=(MOD,PASS)                                           00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(50,10))                00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))     00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                00160000
//GO EXEC PGM=LOADER,PARM=(MAP,LET,PRINT),COND=(5,LT,ALGOL)            00180000
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                             00200000
//SYSLIB DD DSN=SYS1.ALGLIB,DISP=SHR                                   00220000
//SYSLOUT DD SYSOUT=A                                                  00240000
//SYSPRINT DD SYSOUT=A                                                 00260000
//ALGLDD01 DD SYSOUT=A                                                 00280000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(20,10))                00300000
```

ALGOFCG - ALGOL COMPILE AND EXECUTE (CONT.)

4. REFERENCE

IBM 360 OS ALGOL Programmers' Guide, GC33-4000

ALGOFCL - ALGOL COMPILE AND LINKAGE EDIT

1. GENERAL

ALGOFCL is a two-step cataloged procedure to compile and linkage edit an ALGOL source program. Programmers will generally use this procedure when they wish to retain a load module in the load library specified by the SYSLMOD statement in the linkage edit step.

2. USING ALGOFCL

2.1 The ALGOFCL procedure is invoked with an EXEC statement coded:

```
//    EXEC   ALGØFCL
```

2.2 The following DD statements must be added to the procedure.

2.21 A ALGØL.SYSIN DD statement defining the input.

```
//ALGØL.SYSIN  DD  *
  (ALGOL source deck)
        or
//ALGØL.SYSIN  DD  (parameters defining a source input
                     data set)
```

2.22 A LKED.SYSLMØD DD statement to define a storage place for the load module.

```
//LKED.SYSLMØD  DD  (parameters describing the load module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //ALGØL. DD statements.

3. EXAMPLE OF USE

```
//   EXEC  ALGØFCL
//ALGØL.SYSIN  DD  *
  ALGOL source deck
//LKED.SYSLMØD  DD DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
/*
```

(Continued on Next Page)

ALGOFCL - ALGOL COMPILE AND LINKAGE EDIT (CONT.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  ALGOFCL

//ALGOL EXEC PGM=ALGOL,REGION=48K                                          00020000
//SYSPRINT DD SYSOUT=A,DCB=(DSORG=PS)                                      00040000
//SYSPUNCH  DD  SYSOUT=P                                                   00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SPACE=(3600,(10,4)),                  *00080000
//           DISP=(MOD,PASS)                                               00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(50,10))                    00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))         00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                    00160000
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST,LET',COND=(5,LT,ALGOL),            00180000
//               REGION=96K                                                00190000
//SYSPRINT DD SYSOUT=A                                                     00200000
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                                 00220000
//        DD DDNAME=SYSIN                                                  00240000
//SYSLIB DD DSN=SYS1.ALGLIB,DISP=SHR                                       00260000
//SYSLMOD DD DSN=&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),                   *00280000
//  SPACE=(TRK,(9,4,1))                                                    00300000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SEP=(SYSLIB,SYSLMOD),                  *00320000
//  SPACE=(TRK,(14,5))                                                     00340000
```

5. REFERENCE

IBM 360 OS ALGOL Programmers' Guide, GC33-4000

ALGOFCLG - ALGOL COMPILE, LINKAGE EDIT, AND EXECUTE

1. GENERAL

ALGOFCLG is a three-step catalogued procedure to compile, linkage edit, and execute an ALGOL source program.

2. USING ALGOFCLG

2.1 ALGOFCLG may be invoked by an EXEC statement.

```
// EXEC ALGØFCLG
```

2.2 The following DD statement must be added to the procedure.

```
//ALGØL.SYSIN  DD  *
  (ALGOL source deck)
             or
//ALGØL.SYSIN  DD  (appropriate parameters defining
                    a source input data set)
```

2.3 In order to supply additional input to the linkage editor, the following DD statement may be added.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //ALGØL. DD statements.

2.4 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //ALGØL. DD statements, all //LKED. DD statements, and any overriding //GØ. DD statements.

(Continued on Next Page)

ALGOFCLG - ALGOL COMPILE, LINKAGE EDIT, AND EXECUTE (CONT.)

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  ALGOFCLG
//ALGOL EXEC PGM=ALGOL,REGION=48K                                       00020000
//SYSPRINT DD SYSOUT=A                                                  00040000
//SYSPUNCH  DD  SYSOUT=P                                                00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SPACE=(3600,(10,4)),               *00080000
//          DISP=(MOD,PASS)                                             00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(50,10))                 00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))      00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                 00160000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET',COND=(5,LT,ALGOL),REGION=96K  00180000
//SYSPRINT DD SYSOUT=A                                                  00200000
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                              00220000
//       DD DDNAME=SYSIN                                                00240000
//SYSLIB DD DSN=SYS1.ALGLIB,DISP=SHR                                    00260000
//SYSLMOD DD DSN=&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),                *00280000
//  SPACE=(TRK,(9,4,1))                                                 00300000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SEP=(SYSLIB,SYSLMOD),               *00320000
//  SPACE=(TRK,(14,5))                                                  00340000
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,ALGOL),(5,LT,LKED))            00360000
//ALGLDD01 DD SYSOUT=A                                                  00380000
//SYSPRINT DD SYSOUT=A                                                  00400000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(20,10))                 00420000
```

4. REFERENCE

IBM 360 OS ALGOL Programmers' Guide, GC 33-4000

ASMFC - ASSEMBLER(F) COMPILE

1. GENERAL

ASMFC is a one-step catalogued procedure to compile an ASSEMBLER(F) source program. It does not normally produce an object module.

2. USING ASMFC

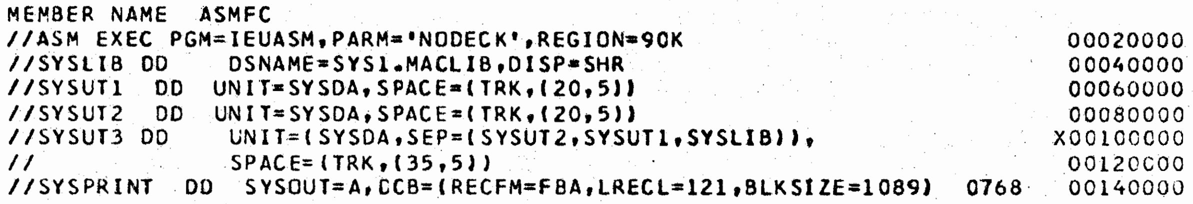
2.1 ASMFC may be invoked by an EXEC statement.

```
// EXEC ASMFC
```

2.2 The following DD statement must be added to the procedure.

```
//ASM.SYSIN  DD  *
 ASSEMBLER source deck
        or
//ASM.SYSIN  DD  (appropriate parameters defining a
                  source input data set)
```

3. CATALOGUED PROCEDURE LISTING



```
MEMBER NAME  ASMFC
//ASM EXEC PGM=IEUASM,PARM='NODECK',REGION=90K                              00020000
//SYSLIB DD     DSNAME=SYS1.MACLIB,DISP=SHR                                 00040000
//SYSUT1  DD  UNIT=SYSDA,SPACE=(TRK,(20,5))                                 00060000
//SYSUT2  DD  UNIT=SYSDA,SPACE=(TRK,(20,5))                                 00080000
//SYSUT3 DD     UNIT=(SYSDA,SEP=(SYSUT2,SYSUT1,SYSLIB)),                   X00100000
//              SPACE=(TRK,(35,5))                                          00120000
//SYSPRINT  DD  SYSOUT=A,CCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)  0768       00140000
```

4. REFERENCE

IBM 360 OS ASSEMBLER(F) Programmers' Guide, GC26-3756

ASMFCG - ASSEMBLER(F) COMPILE AND EXECUTE

1. GENERAL

ASMFCG is a two-step catalogued procedure to compile and execute an ASSEMBLER(F) source program using the IBM LOADER. Note that there is no linkage edit step, and a load module is not produced.

2. USING ASMFCG

2.1 ASMFCG may be invoked by an EXEC statement.

```
// EXEC ASMFCG
```

2.2 The following DD statement must be added to the procedure.

```
//ASM.SYSIN  DD  *
 ASSEMBLER source deck
          or
//ASM.SYSIN  DD  (appropriate parameters defining a
                  source input data set)
```

2.3 A CPU time limit of 2 minutes is assumed for execution. A small portion of this time will be used by the LOADER prior to execution. If additional time is needed, the TIME parameter may be coded on the EXEC statement.

```
// EXEC ASMFCG,TIME.GØ=5
```

2.4 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //ASM. DD statements and any overriding //GØ. DD statements. Please note that no SYSUDUMP or SYSABEND DD statement is supplied in the GO step. The user must provide one if he wishes to cover ABEND conditions.

2.5 Programs which require a REGION of more than 56K must increment their requirements by 28K when using the LOADER. The incremented requirement should be specified as follows:

```
// EXEC ASMFCG,LDREG=90K
```

This will automatically increment both the REGION and the LOADER's SIZE.

ASMFCG - ASSEMBLER(F) COMPILE AND EXECUTE (CONT.)

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  ASMFCG
//ASMFCG  PROC LDREG=90K,PARAM=                            LAJ 10/20/71 00000100
//ASM      EXEC PGM=IEUASM,PARM=(NODECK,LOAD),REGION=90K                00000200
//SYSLIB    DD DSN=SYS1.MACLIB,DISP=SHR                                 00000300
//SYSUT1    DD UNIT=SYSDA,SPACE=(TRK,(20,5))                            00000400
//SYSUT2    DD UNIT=SYSDA,SPACE=(TRK,(20,5))                            00000500
//SYSUT3    DD UNIT=(SYSDA,SEP=(SYSUT1,SYSUT2,SYSLIB)),                C00000600
//             SPACE=(TRK,(35,5))                                       00000700
//SYSPRINT  DD SYSOUT=A                                                 00000800
//SYSGO     DD DSN=&&LOADSET,UNIT=SYSDA,SPACE=(400,(50,10)),           C00000900
//             DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)      00001000
//GO       EXEC PGM=LOADER,PARM='MAP,LET,SIZE=&LDREG,NCAL/&PARAM',     C00001100
//             REGION=&LDREG,TIME=2,COND=(5,LT,ASM)                     00001200
//* FOR AN ABEND DUMP INCLUDE A SYSUDUMP DD CARD                         00001300
//SYSLOUT   DD SYSOUT=A                                                 00001400
//SYSLIN    DD DSN=&&LOADSET,DISP=(OLD,DELETE)                          00001500
```

4. REFERENCE

IBM 360 OS ASSEMBLER(F) Programmers' Guide, GC26-3756

ASMFCL - ASSEMBLER(F) COMPILE AND LINKAGE EDIT

1. GENERAL

ASMFCL is a two-step cataloged procedure to compile and linkage edit an ASSEMBLER(F) source program. Programmers will generally use this procedure when they wish to retain a load module in the load library specified by the SYSLMOD statement in the linkage edit step.

2. USING ASMFCL

2.1 The ASMFCL procedure is invoked with an EXEC statement coded:

```
//  EXEC  ASMFCL
```

2.2 The following DD statements must be added to the procedure.

2.21 A ASM.SYSIN DD statement defining the input.

```
//ASM.SYSIN  DD  *
 ASSEMBLER source deck
      or
//ASM.SYSIN  DD  (parameters defining a source input
                  data set)
```

2.22 A LKED.SYSLMØD DD statement to define a storage place for the load module.

```
//LKED.SYSLMØD  DD  (parameters describing the load module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //ASM. DD statements.

3. EXAMPLE OF USE

```
//  EXEC  ASMFCL
//ASM.SYSIN  DD  *
 ASSEMBLER source deck
//LKED.SYSLMØD  DD DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
/*
```

(Continued on Next Page)

ASMFCL - ASSEMBLER(F) COMPILE AND LINKAGE EDIT (CONT.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  ASMFCL
//ASM EXEC PGM=IEUASM,PARM='NODECK,LOAD',REGION=90K                          00000010
//SYSLIB DD    DSNAME=SYS1.MACLIB,DISP=SHR                                   00000020
//SYSUT1  DD  UNIT=SYSDA,SPACE=(TRK,(20,5))                                  00000030
//SYSUT2  DD  UNIT=SYSDA,SPACE=(TRK,(20,5))                                  00000040
//SYSUT3 DD    UNIT=(SYSDA,SEP=(SYSUT2,SYSUT1,SYSLIB)),                     X00000050
//             SPACE=(TRK,(35,5))                                            00000060
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)  0768        00000070
//SYSGO     DD  DSN=&LOADSET,UNIT=SYSDA,SPACE=(400,(50,10)),                 00000080
//             DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)           00000090
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST,LET',REGION=90K,                     00000100
//            COND=(8,LT,ASM)                                                00000110
//SYSLIN DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)                             00000120
//        DD    DDNAME=SYSIN                                                 00000130
//SYSUT1  DD  UNIT=(SYSDA,SEP=SYSLIN),SPACE=(TRK,(14,15))                    00000140
//SYSPRINT DD SYSOUT=A                                                       00000150
```

5. REFERENCE

IBM 360 OS ASSEMBLER(F) Programmers' Guide, GC26-3756

ASMFCLG - ASSEMBLER(F) COMPILE, LINKAGE EDIT, AND EXECUTE

1. GENERAL

ASMFCLG is a three-step catalogued procedure to assemble, linkage edit, and execute an ASSEMBLER(F) source program.

2. USING ASMFCLG

2.1 ASMFCLG may be invoked by an EXEC statement.

```
// EXEC ASMFCLG
```

2.2 The following DD statement must be added to the procedure.

```
//ASM.SYSIN  DD  *
 ASSEMBLER source deck
          or
//ASM.SYSIN  DD  (appropriate parameters defining
                  a source input data set)
```

2.3 In order to supply additional input to the linkage editor, the following DD statement may be added.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //ASM. DD statements.

2.4 Additional data sets required by the source program may be specified on DD statements.

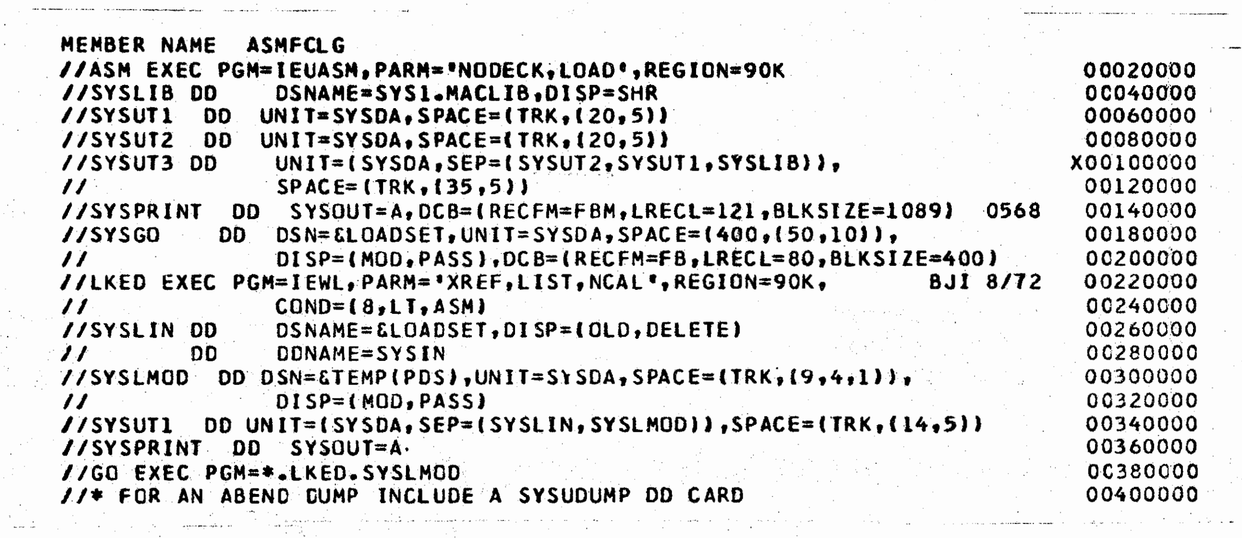
```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //ASM. DD statements, all //LKED. DD statements, and any overriding //GØ. DD statements. Please note that no SYSUDUMP or SYSABEND DD statement is supplied in the GO step. The user must provide one if he wishes to cover ABEND conditions.

(Continued on Next Page)

ASMFCLG - ASSEMBLER(F) COMPILE, LINKAGE EDIT, AND EXECUTE (CONT.)

3. CATALOGUED PROCEDURE LISTING



```
MEMBER NAME  ASMFCLG
//ASM EXEC PGM=IEUASM,PARM='NODECK,LOAD',REGION=90K                     00020000
//SYSLIB DD    DSNAME=SYS1.MACLIB,DISP=SHR                              00040000
//SYSUT1  DD  UNIT=SYSDA,SPACE=(TRK,(20,5))                             00060000
//SYSUT2  DD  UNIT=SYSDA,SPACE=(TRK,(20,5))                             00080000
//SYSUT3 DD    UNIT=(SYSDA,SEP=(SYSUT2,SYSUT1,SYSLIB)),                X00100000
//            SPACE=(TRK,(35,5))                                        00120000
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=1089)  0568    00140000
//SYSGO     DD   DSN=&LOADSET,UNIT=SYSDA,SPACE=(400,(50,10)),            00180000
//            DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)       00200000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,NCAL',REGION=90K,        BJI 8/72   00220000
//            COND=(8,LT,ASM)                                           00240000
//SYSLIN DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)                        00260000
//        DD    DDNAME=SYSIN                                            00280000
//SYSLMOD  DD DSN=&TEMP(PDS),UNIT=SYSDA,SPACE=(TRK,(9,4,1)),            00300000
//            DISP=(MOD,PASS)                                           00320000
//SYSUT1  DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(TRK,(14,5))       00340000
//SYSPRINT  DD  SYSOUT=A                                                00360000
//GO EXEC PGM=*.LKED.SYSLMOD                                            00380000
//* FOR AN ABEND DUMP INCLUDE A SYSUDUMP DD CARD                        00400000
```

4. REFERENCE

IBM 360 OS ASSEMBLER(F) Programmers Guide, GC26-3756

AUTOFLOW - SOURCE PROGRAM DOCUMENTATION

1. GENERAL

AUTOFLOW is a cataloged procedure which facilitates use of the leased utility program, AUTOFLOW, for generating a flowchart, source and cross-reference listings from program source statements. These facilities are useful in job maintenance functions and are acceptable as part of the final documentation package required by Headquarters.

2. USING AUTOFLOW

2.1 The AUTOFLOW procedure is invoked with an EXEC statement coded:

```
//  EXEC AUTOFLOW,NAME=PROGRAM
 control cards
```

2.11 The NAME parameter equates to the programmer supplied member name. This name appears on each page of the printed listings.

2.12 Control card statements are required to select desired source language routines. The material referenced in Paragraph 8 contains a description of the control cards and other information for using AUTOFLOW.

2.2 The AUTOFLOW procedure uses a REGION parameter of 120K.

3. ASP CODING REQUIREMENTS

3.1 AUTOFLOW output is printed with eight lines per inch spacing. This spacing requires an ASP FORMAT card as follows:

```
//*FORMAT PR,DDNAME=SYSPRINT,CARRIAGE=E025
```

All SYSOUT output will be under the control of the above FORMAT statement.

AUTOFLOW - SOURCE PROGRAM DOCUMENTATION (CONT.)

4. EXAMPLES OF USE

4.1 Single Card Module

```
//FLOW.SYSIN DD *
 COBOL (LIST,DXREF)    ⎫  Specify one control card
 ASSEMBLY (LIST)       ⎬  statement to select desired
 FORTRAN (LIST)        ⎪  routine.
 PL/I (LIST)           ⎭
 source deck (ALL JCL REMOVED)
./ MODEND
/*
```

4.2 Multiple Card Modules

```
//FLOW.SYSIN  DD  *
 COBOL (LIST,DXREF)
  source deck
./ MODEND
 ASSEMBLY (LIST)
  source deck
./ MODEND
 FORTRAN (LIST)
  source deck
./ MODEND
/*
```

4.3 Multiple Mixed Media Modules

```
//FLOW.DISK   DD  DSN=SRCPDS,DISP=SHR
//FLOW.SYSIN DD  *
 FORTRAN (LIST)
 source deck
DISK(MEMBER) COBOL (LIST,DXREF)
/*
```

4.4 Librarian Source Modules

```
//FLOW.MASTER DD  DSN=DEV.SOURCE
//FLOW.SYSIN DD  *
 COBOL (LIST)
$LIBR(MEMBER) COBOL (LIST,DXREF)
/*
```

AUTOFLOW - SOURCE PROGRAM DOCUMENTATION (CONT.)

5. ANS COBOL MODULES

5.1 An additional control statement is required to successfully chart an ANS COBOL source module. It is the OPTION statement and it must follow the COBOL statement. The form of the OPTION statement is:

```
OPTION ANSI=YES
```

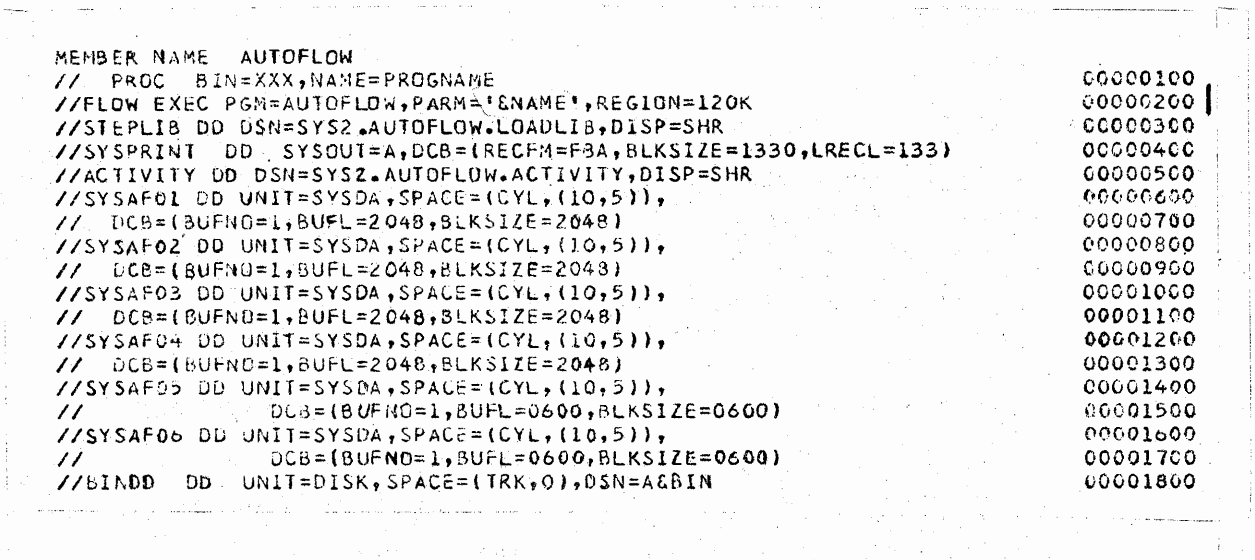
5.2 If the ANS COBOL source module is not on cards, an additional parameter must be specified in the COBOL statement in order that the OPTION card be processed. The additional parameter is named PARM. In the example in paragraph 4.3, if the module in SRCPDS is ANS COBOL, the following should be coded:

```
DISK(MEMBER) COBOL (LIST,DXREF,PARM)
OPTION ANSI=YES
```

6. AUTOFLOW OUTPUT

Output media from AUTOFLOW is directed to SYSOUT under the ddname SYSPRINT.

7. CATALOGED PROCEDURE LISTING



```
MEMBER NAME   AUTOFLOW
//   PROC   BIN=XXX,NAME=PROGNAME                                   00000100
//FLOW EXEC PGM=AUTOFLOW,PARM='&NAME',REGION=120K                   00000200
//STEPLIB DD DSN=SYS2.AUTOFLOW.LOADLIB,DISP=SHR                     00000300
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,BLKSIZE=1330,LRECL=133)     00000400
//ACTIVITY DD DSN=SYS2.AUTOFLOW.ACTIVITY,DISP=SHR                   00000500
//SYSAF01 DD UNIT=SYSDA,SPACE=(CYL,(10,5)),                         00000600
//   DCB=(BUFNO=1,BUFL=2048,BLKSIZE=2048)                           00000700
//SYSAF02 DD UNIT=SYSDA,SPACE=(CYL,(10,5)),                         00000800
//   DCB=(BUFNO=1,BUFL=2048,BLKSIZE=2048)                           00000900
//SYSAF03 DD UNIT=SYSDA,SPACE=(CYL,(10,5)),                         00001000
//   DCB=(BUFNO=1,BUFL=2048,BLKSIZE=2048)                           00001100
//SYSAF04 DD UNIT=SYSDA,SPACE=(CYL,(10,5)),                         00001200
//   DCB=(BUFNO=1,BUFL=2048,BLKSIZE=2048)                           00001300
//SYSAF05 DD UNIT=SYSDA,SPACE=(CYL,(10,5)),                         00001400
//             DCB=(BUFNO=1,BUFL=0600,BLKSIZE=0600)                 00001500
//SYSAF06 DD UNIT=SYSDA,SPACE=(CYL,(10,5)),                         00001600
//             DCB=(BUFNO=1,BUFL=0600,BLKSIZE=0600)                 00001700
//BINDD   DD  UNIT=DISK,SPACE=(TRK,0),DSN=A&BIN                     00001800
```

8. REFERENCE

Division 3, Chapter 2, Section 3, AUTOFLOW

CHGPDS - ADDING, REPLACING, CHANGING CARD IMAGE PDS MEMBERS

1. GENERAL

CHGPDS is a cataloged procedure which facilitates use of the IBM Utility Program IEBUPDTE for updating an existing card image partitioned data set. The data set must be cataloged.

2. SYMBOLIC PARAMETER

CHGPDS requires a DSN parameter which supplies the library name. The parameter must be enclosed in apostrophes if the name is indexed (contains periods).

3. USING CHGPDS

3.1 The CHGPDS procedure is invoked with an EXEC statement which includes the symbolic parameter, DSN.

```
//   EXEC CHGPDS,DSN='user.library'
```

3.2 The user must include a SYSIN DD statement and appropriate IEBUPDTE control statements

```
//SYSIN  DD *  or  //SYSIN  DD  DATA
  control statements
```

4. EXAMPLE OF USE

```
//   EXEC CHGPDS,DSN='PLC.PRØCLIB'
//SYSIN  DD  DATA
./ ADD NAME=NEWPRØC
./ NUMBER NEW1=10,INCR=10
new procedure JCL cards
/*
```

5. RESTRICTION

The Library must be cataloged and have LRECL=80.

6. SUGGESTED USE

CHGPDS is recommended for updating libraries such as procedure or control libraries from card input.

(Continued on Next Page)

CHGPDS - ADDING, REPLACING, CHANGING CARD IMAGE PDS MEMBERS (CONT.)

7. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  CHGPDS
//  PROC  DSN='SYS1.PROCLIB'                                               00000100
//CHG  EXEC  PGM=ATTUPDTE,DPRTY=(13,1)                                     00000200
//SYSPRINT  DD  SYSOUT=A                                                   00000300
//SYSUT1  DD  DISP=SHR,DSN=&DSN                                            00000400
//SYSUT2  DD  DISP=SHR,DSN=*.SYSUT1                                        00000500
```

8. REFERENCE

IBM OS Utilities GC28-6586, IEBUPDTE

COBLG - COBOL LINKAGE EDIT AND EXECUTE

1. GENERAL

COBLG is a two step cataloged procedure to linkage edit an object or load module to produce a load module and execute it. The linkage edit step provides a reference to the automatic call library SYS1.COBLIB. This will be effective for any COBOL compiler supported by the Data Center.

2. USING COBLG

2.1 COBLG may be invoked by an EXEC statement.

```
//    EXEC   CØBLG
```

2.2 The following DD statements may be added to supply additional input to the linkage editor.

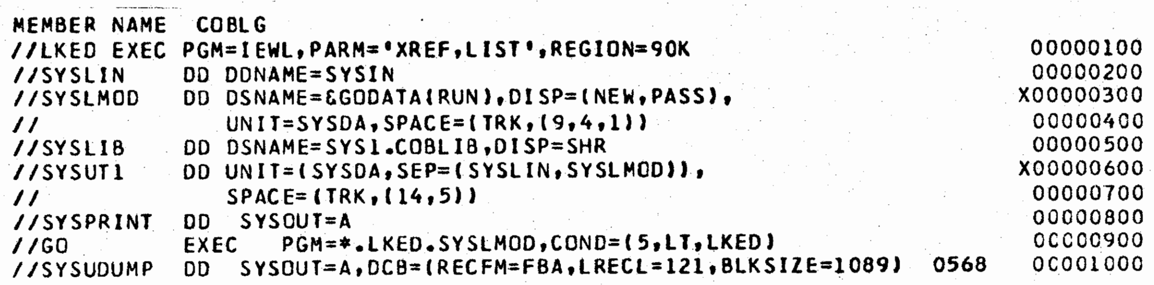
```
//LKED.SYSIN    DD  (parameters describing linkage editor control input
                     and/or object module input)
     or
//LKED.ddname   DD  (parameters describing an additional call library)
```

2.3 Additional data sets required by the users program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //LKED. DD statements.

3. CATALOGED PROCEDURE LISTING



```
MEMBER NAME  COBLG
//LKED EXEC PGM=IEWL,PARM='XREF,LIST',REGION=90K                            00000100
//SYSLIN    DD DDNAME=SYSIN                                                 00000200
//SYSLMOD   DD DSNAME=&GODATA(RUN),DISP=(NEW,PASS),                        X00000300
//               UNIT=SYSDA,SPACE=(TRK,(9,4,1))                             00000400
//SYSLIB    DD DSNAME=SYS1.COBLIB,DISP=SHR                                  00000500
//SYSUT1    DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),                          X00000600
//               SPACE=(TRK,(14,5))                                         00000700
//SYSPRINT  DD  SYSOUT=A                                                    00000800
//GO        EXEC   PGM=*.LKED.SYSLMOD,COND=(5,LT,LKED)                      0C000900
//SYSUDUMP  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)  0568       0C001000
```

4. REFERENCE

IBM ANS COBOL version 4 Programmers' Guide, SC28-6456
IBM 370 OS/VS Linkage Editor and Loader, GC26-3813
IBM 370 OS/VS Message Library: Linkage Editor and Loader Messages, GC38-1007

COB4C - ANS COBOL COMPILE

1. GENERAL

COB4C is a one-step catalogued procedure to compile an ANS COBOL source program. It does not normally produce an object module.

2. USING COB4C

2.1 COB4C may be invoked by an EXEC statement.

```
// EXEC CØB4C
```

2.2 The following DD statement must be added to the procedure.

```
//CØB.SYSIN  DD  *
 COBOL source deck
       or
//CØB.SYSIN  DD  (appropriate parameters defining a
                  source input data set)
```

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  COB4C
//COB4C PROC LINES=55                                                   00000005
//COB EXEC PGM=IKFCBL00,REGION=128K,                                    00000010
// PARM='NOLOAD,NODECK,LINECNT=&LINES,BUFSIZE=16K,SIZE=120K'            00000012
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR                              00000015
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)           00000030
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(12,5))                            00000040
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(12,5))                            00000050
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(9,5))                             00000060
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(6,5))                             00000070
```

4. REFERENCE

IBM ANS COBOL version 4 Programmers' Guide, SC28-6456

COB4CG - ANS COBOL COMPILE AND EXECUTE

1. GENERAL

COB4CG is a two-step catalogued procedure to compile and execute an ANS COBOL source program using the IBM LOADER. Note that there is no linkage edit step, and a load module is not produced.

2. USING COB4CG

2.1 COB4CG may be invoked by an EXEC statement.

```
// EXEC CØB4CG
```

2.2 The following DD statement must be added to the procedure.

```
//CØB.SYSIN  DD  *
 COBOL source deck
      or
//CØB.SYSIN  DD  (appropriate parameters defining a
                  source input data set)
```

2.3 A CPU time limit of 10 minutes is assumed for execution. A small portion of this time will be used by the LOADER prior to execution. If additional time is needed, the TIME parameter may be coded on the EXEC statement.

```
// EXEC CØB4CG,TIME.GØ=12
```

2.4 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //CØB. DD statements and any overriding //GØ. DD statements.

2.5 Programs which require a REGION of more than 56K must increment their requirements by 28K when using the LOADER. The incremented requirement should be specified as follows:

```
// EXEC CØB4CG,LDREG=90K
```

This will automatically increment both the REGION and the LOADER's SIZE.

(Continued on Next Page)

COB4CG - ANS COBOL COMPILE AND EXECUTE (CONT'D)

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  COB4CG
//COB4CG PROC LDREG=84K,LINES=55                                        00000010
//COB EXEC PGM=IKFCBL00,REGION=128K,                                    00000020
// PARM='LINECNT=&LINES,BUFSIZE=16K,SIZE=120K'                          00000022
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR                              00000025
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)         00000040
//SYSUT1  DD  UNIT=SYSDA,SPACE=(TRK,(12,5))                             00000050
//SYSUT2  DD  UNIT=SYSDA,SPACE=(TRK,(12,5))                             00000060
//SYSUT3  DD  UNIT=SYSDA,SPACE=(TRK,(9,5))                              00000070
//SYSUT4  DD  UNIT=SYSDA,SPACE=(TRK,(6,5))                              00000080
//SYSLIN  DD  DSNAME=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,              00000090
// SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)   1/71       00000100
//GO  EXEC  PGM=LOADER,PARM='MAP,LET,SIZE=&LDREG',REGION=&LDREG,        00000110
//     COND=(5,LT,COB),TIME=10                                          00000120
//SYSLIN  DD  DSN=&&LOADSET,DISP=(OLD,DELETE)                           00000130
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR                                00000140
//SYSLOUT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=121)           00000150
//SYSUDUMP DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)  1/71     00000160
//SYSOUT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=120)            00000170
```

4. REFERENCE

IBM ANS COBOL version 4 Programmers' Guide SC28-6456

COB4CL - ANS COBOL COMPILE AND LINKAGE EDIT

1. GENERAL

COB4CL is a two-step cataloged procedure to compile and linkage edit an ANS COBOL source program. Programmers will generally use this procedure when they wish to retain a load module in the load library specified by the SYSLMOD statement in the linkage edit step.

2. USING COB4CL

2.1 The COB4CL procedure is invoked with an EXEC statement coded:

```
//   EXEC  CØB4CL
```

2.2 The following DD statements must be added to the procedure.

2.21 A CØB.SYSIN DD statement defining the input.

```
//CØB.SYSIN  DD  *
 (COBOL source deck)
       or
//CØB.SYSIN  DD  (parameters defining a source input
                  data set)
```

2.22 A LKED.SYSLMØD DD statement to define a permanent storage place for the load module.

```
//LKED.SYSLMØD  DD  (parameters describing the load module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //CØB. DD statements.

3. EXAMPLE OF USE

```
//   EXEC  CØB4CL
//CØB.SYSIN  DD  *
  COBOL source deck
//LKED.SYSLMØD  DD DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
/*
```

(Continued on Next Page)

COB4CL - ANS COBOL COMPILE AND LINKAGE EDIT (CONT.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  COB4CL
//COB4CL PROC LINES=55                                                    00000005
//COB EXEC PGM=IKFCBL00,REGION=128K,                                      00000010
// PARM='LINECNT=&LINES,BUF=16K,SIZE=120K'                                00000012
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR                                00000015
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)           00000030
//SYSUT1  DD  UNIT=SYSDA,SPACE=(TRK,(12,5))                               00000040
//SYSUT2  DD  UNIT=SYSDA,SPACE=(TRK,(12,5))                               00000050
//SYSUT3  DD  UNIT=SYSDA,SPACE=(TRK,(9,5))                                00000060
//SYSUT4  DD  UNIT=SYSDA,SPACE=(TRK,(6,5))                                00000070
//SYSLIN  DD  DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                 00000080
// SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)   1/71         00000090
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST,LET',COND=(5,LT,COB),             00000100
//              REGION=128K                                               00000110
//SYSLIN  DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                           00000120
//          DD  DDNAME=SYSIN                                              00000130
//SYSLMOD  DD DSN=&GODATA(RUN),DISP=(NEW,PASS),                           00000133
//            UNIT=SYSDA,SPACE=(TRK,(9,,1))                               00000137
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR                                  00000140
//SYSUT1  DD  UNIT=SYSDA,SPACE=(TRK,(14,5))                               00000150
//SYSPRINT  DD  SYSOUT=A                                                  00000160
```

5. REFERENCE

IBM ANS COBOL version 4 Programmers' Guide, SC28-6456.

COB4CLG ANS COBOL COMPILE, LINKAGE EDIT, AND EXECUTE

1. GENERAL

COB4CLG is a three-step catalogued procedure to compile, linkage edit, and execute an ANS COBOL source program.

2. USING COB4CLG

2.1 COB4CLG may be invoked by an EXEC statement.

```
// EXEC CØB4CLG
```

2.2 The following DD statement must be added to the procedure.

```
//CØB.SYSIN  DD  *
 COBOL source deck
          or
//CØB.SYSIN  DD  (appropriate parameters defining
                  a source input data set)
```

2.3 In order to supply additional input to the linkage editor, the following DD statement may be added.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //CØB. DD statements.

2.4 A CPU time limit of 10 minutes is assumed for the execution step. If additional time is needed, the TIME parameter may be coded on the EXEC statement.

```
// EXEC  CØB4CLG,TIME.GØ=12
```

2.5 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //CØB. DD statements, all //LKED. DD statements, and any overriding //GØ. DD statements.

(Continued on Next Page)

COB4CLG - ANS COBOL COMPILE, LINKAGE EDIT, AND EXECUTE (CONT.)

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  COB4CLG
//COB4CLG PROC LINES=55                                                 00000100
//COB       EXEC PGM=IKFCBL00,PARM='LINECNT=&LINES',REGION=128K         00000200
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR                              00000300
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)           00000400
//SYSUDUMP DD SYSOUT=A                                                  00000500
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(12,5))                               00000600
//SYSUT2 DD UNIT=SYSDA,SPACE=(TRK,(12,5))                               00000700
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(9,5))                                00000800
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(6,5))                                00000900
//SYSLIN DD DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                    00001000
// SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)              00001100
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST,LET',COND=(5,LT,COB),           00001200
//            REGION=128K                                               00001250
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                              00001300
//       DD DDNAME=SYSIN                                                00001400
//SYSLMOD DD DSN=&GODATA(RUN),DISP=(NEW,PASS),UNIT=SYSDA,               00001500
// SPACE=(TRK,(9,,1))                                                   00001600
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR                                00001700
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(TRK,(14,5))        00001800
//SYSPRINT DD SYSOUT=A,SPACE=(TRK,(5,5))                                00001900
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,COB),(5,LT,LKED)),TIME=10      00002100
//SYSUDUMP DD SYSOUT=A                                                  00002200
//SYSOUT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=120)              00002300
```

4. REFERENCE

IBM ANS COBOL version 4 Programmers' Guide, SC28-6456

DASORT - DIRECT ACCESS UTILITY SORT

1. GENERAL

DASORT is a cataloged procedure which facilitates use of Syncsort III, a sort program with direct access devices designated for sort work media. The user may vary the unit of work space, the amount of work space, the region and the direct access device type by supplying values for a set of symbolic parameters. If the default values are accepted, the procedure will supply a region of 60K and 6 cylinders of 3330 work space.

2. SYMBOLIC PARAMETERS

SPACE - This is used to designate the unit of space to be allocated. It may be TRK or CYL. The default is CYL.

AMT - This is the number of units of space to be allocated for each of the six sort work areas. The default is 1.

REG - This is used to supply a value for the region parameter for the step. The default is 60 to yield a region of 60K.

UNIT - This is used to designate the sort work device type. The default is DISK.

SEC - This is the number of units of secondary space allocation. The default is 1.

3. USING DASORT

The procedure is invoked with an EXEC statement which may include the symbolic parameters described in paragraph 2. The user must provide DD statements describing input under the ddname SORTIN, the output file under the ddname SORTOUT, and the sort control card input under the ddname SYSIN.

```
// EXEC DASØRT,SPACE=xxx,AMT=yyy,REG=zzz
//SØRTIN DD input file parameters
//SØRTØUT DD output file parameters
//SYSIN DD *
  sort control statement
```

4. EXAMPLES OF USE

4.1 Using default values:

```
// EXEC DASØRT
//SØRTIN DD DSN=&TESTDATA,DISP=(ØLD,DELETE)
//SØRTØUT DD DSN=&SØRTDATA,DISP=(,PASS),UNIT=DISK,
//  DCB=(RECFM=FB,LRECL=50,BLKSIZE=1000),SPACE=(CYL,(4,1))
//SYSIN DD *
 SØRT FIELDS=(32,8,CH,A),SIZE=E15000
```

DASORT - DIRECT ACCESS UTILITY SORT (CONT.)

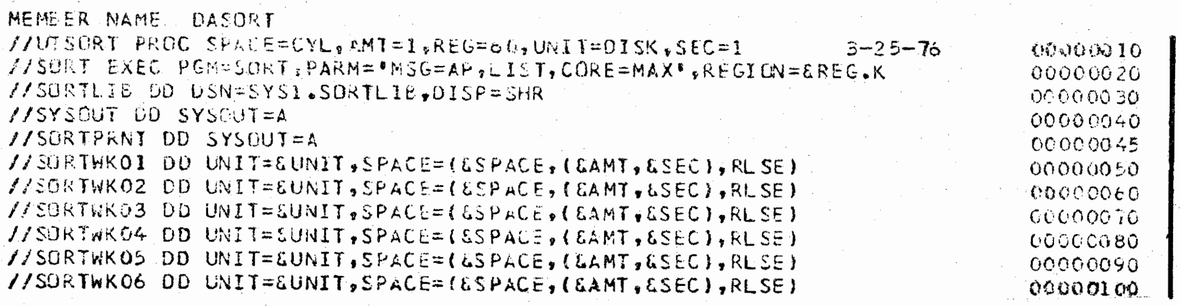
4.2 A small sort: (Note: minimum AMT when SPACE=TRK is 5.)

```
// EXEC  DASØRT,SPACE=TRK,AMT=5
//SØRTIN  DD  input file description
//SØRTØUT DD  output file description
//SYSIN   DD  *
 SØRT FIELDS=(14,10,CH,A),SIZE=E500
```

4.3 A moderately large sort:

```
// EXEC  DASØRT,SPACE=CYL,AMT=4,REG=90
//SØRTIN  DD  input file description
//SØRTØUT DD  output file description
//SYSIN   DD  *
 SØRT FIELDS=(22,6,CH.A),SIZE=E75000
```

5. CATALOGED PROCEDURE LISTING



```
MEMEER NAME  DASORT
//UTSORT PROC SPACE=CYL,AMT=1,REG=60,UNIT=DISK,SEC=1       3-25-76      00000010
//SORT EXEC PGM=SORT,PARM='MSG=AP,LIST,CORE=MAX',REGION=&REG.K          00000020
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR                                  00000030
//SYSOUT DD SYSOUT=A                                                    00000040
//SORTPRNT DD SYSOUT=A                                                  00000045
//SORTWK01 DD UNIT=&UNIT,SPACE=(&SPACE,(&AMT,&SEC),RLSE)                00000050
//SORTWK02 DD UNIT=&UNIT,SPACE=(&SPACE,(&AMT,&SEC),RLSE)                00000060
//SORTWK03 DD UNIT=&UNIT,SPACE=(&SPACE,(&AMT,&SEC),RLSE)                00000070
//SORTWK04 DD UNIT=&UNIT,SPACE=(&SPACE,(&AMT,&SEC),RLSE)                00000080
//SORTWK05 DD UNIT=&UNIT,SPACE=(&SPACE,(&AMT,&SEC),RLSE)                00000090
//SORTWK06 DD UNIT=&UNIT,SPACE=(&SPACE,(&AMT,&SEC),RLSE)                00000100
```

6. REFERENCE

6.1 Syncsort III Programmers Guide, Whitlow Computer Systems, Inc.
6.2 Division 3, Chapter 2, Section 2, SORTSPAC
6.3 Division 3, Chapter 2, Section 3, SORT, SYNCSORT III

FORTGC/FORTHC - FORTRAN COMPILE

1. GENERAL

FORTGC and FORTHC are one-step catalogued procedures to compile FORTRAN-G or FORTRAN-H source programs. The procedures do not normally produce an object module.

2. USING FORTGC or FORTHC

2.1 The procedures may be invoked be an EXEC statement.

```
// EXEC FØRTGC     or     // EXEC FØRTHC
```

2.2 The following DD statement must be added to the procedures.

```
//FØRT.SYSIN  DD  *
  (FORTRAN source deck)
        or
//FØRT.SYSIN  DD  (appropriate parameters defining a
                   source input data set)
```

3. CATALOGUED PROCEDURE LISTINGS

3.1 FORTGC

```
MEMBER NAME  FORTGC
//*   ALIAS NAME FOR FORTGC IS FORTEC.                                       00010000
//FORT     EXEC PGM=IEYFORT,REGION=100K,DPRTY=2                              00020000
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=1080)  0768       00040000
//SYSLIN  DD  DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                    00080000
//              SPACE=(TRK,(5,2))                                            00100000
```

3.2 FORTHC

```
MEMBER NAME  FORTHC
//FORT  EXEC  PGM=IEKAA00,REGION=150K                                        10000000
//SYSPRINT DD  SYSOUT=A                                                      20000000
//SYSPUNCH DD  SYSOUT=P                                                      30000000
//SYSLIN   DD  DSNAME=&LOADSET,UNIT=SYSSQ,DISP=(MOD,PASS),                  *40000000
//             SPACE=(400,(200,50),RLSE)                                     50000000
//SYSUT1     DD UNIT=SYSDA,SPACE=(TRK,(15,8))                                60000000
//SYSUT2     DD UNIT=SYSDA,SPACE=(TRK,(3,3))                                 70000000
```

4. REFERENCE

IBM 360 OS FORTRAN IV Programmers Guide, GC28-6817

FORTGCL/FORTHCL - FORTRAN COMPILE AND LINKAGE EDIT

1. GENERAL

FORTGCL and FORTHCL are two-step cataloged procedures to compile and linkage edit FORTRAN-G or FORTRAN-H source programs. Programmers will generally use these procedures when they wish to retain a load module in the load library specified by the SYSLMOD statement in the linkage edit step.

2. USING FORTGCL or FORTHCL

2.1 The procedures may be invoked with an EXEC statement coded:

```
//    EXEC   FØRTGCL      or      //    EXEC   FØRTHCL
```

2.2 The following DD statements must be added to the procedures.

2.21 A FØRT.SYSIN DD statement defining the input.

```
//FØRT.SYSIN  DD  *
  (FORTRAN source deck)
         or
//FØRT.SYSIN  DD  (parameters defining a source input
                   data set)
```

2.22 A LKED.SYSLMØD DD statement to define a storage place for the load module.

```
//LKED.SYSLMØD  DD  (parameters describing the load module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //FØRT. DD statements.

3. EXAMPLE OF USE

```
//   EXEC  FØRTGCL     or     //   EXEC  FØRTHCL
//FØRT.SYSIN  DD  *
  FØRTRAN source deck
//LKED.SYSLMØD  DD DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
/*
```

(Continued on Next Page)

FORTGCL/FORTHCL - FORTRAN COMPILE AND LINKAGE EDIT (CONT.)

4. CATALOGED PROCEDURE LISTINGS

4.1 FORTGCL

```
MEMBER NAME  FORTGCL
//*  ALIAS NAME FOR FORTGCL IS FORTECL.                                     00000010
//FORT    EXEC  PGM=IEYFORT,REGION=100K,DPRTY=2                             00000020
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=1080)  0768       00000030
//SYSLIN  DD  DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                      00000040
//             SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)      00000050
//LKED  EXEC  PGM=ATTIEWL,PARM='XREF,LET,LIST',REGION=90K,                  00000060
//         COND=(4,LT,FORT)                                                 00000061
//SYSLIB    DD  DSNAME=SYS1.FORTLIB,DISP=SHR                                00000070
//SYSPRINT DD SYSOUT=A                                                      00000080
//SYSUT1  DD UNIT=SYSDA,SPACE=(TRK,(4,2),RLSE)                              00000090
//SYSLIN    DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                           00000100
//          DD  DDNAME=SYSIN                                                00000110
```

4.2 FORTHCL

```
MEMBER NAME  FORTHCL
//FORT    EXEC  PGM=IEKAA00,REGION=150K                                     07000014
//SYSPRINT DD  SYSOUT=A                                                     14000014
//SYSPUNCH DD   SYSOUT=P                                                    21000014
//SYSLIN    DD  DSNAME=&LOADSET,UNIT=SYSSQ,DISP=(MOD,PASS),                *28000014
//              SPACE=(400,(200,50),RLSE)                                   35000014
//SYSUT1      DD UNIT=SYSDA,SPACE=(TRK,(15,8))                              36000014
//SYSUT2      DD UNIT=SYSDA,SPACE=(TRK,(3,3))                               37000014
//LKED EXEC PGM=ATTIEWL,REGION=90K,PARM='MAP,LET,LIST',                     42000014
//  COND=(4,LT,FORT)                                                        42000015
//SYSLIB    DD  DSNAME=SYS1.FORTLIB,DISP=SHR                                49000014
//SYSPRINT DD  SYSOUT=A                                                     56000014
//SYSLIN    DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                           77000014
//          DD  DDNAME=SYSIN                                                84000014
//SYSUT1   DD  DSN=&SYSUT1,UNIT=SYSDA,SPACE=(TRK,(14,5))                    91000018
```

5. REFERENCE

IBM 360 OS FORTRAN IV Programmers Guide, GC28-6817

FORTGCLG/FORTHCLG - FORTRAN COMPILE, LINKAGE EDIT, AND EXECUTE

1. GENERAL

FORTGCLG and FORTHCLG are three-step cataloged procedures to compile, linkage edit, and execute FORTRAN-G or FORTRAN-H source programs.

2. USING FORTGCLG or FORTHCLG

2.1 The procedures may be invoked by an EXEC statement.

```
// EXEC FØRTGCLG   or   // EXEC FØRTHCLG
```

2.2 The following DD statement must be added to the procedure.

```
//FØRT.SYSIN  DD  *
  (FORTRAN source deck)
            or
//FØRT.SYSIN  DD  (appropriate parameters defining
                   a source input data set)
```

2.3 In order to supply additional input to the linkage editor, the following DD statement may be added.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //FØRT. DD statements.

2.4 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //FØRT. DD statements, all //LKED. DD statements, and any overriding //GØ. DD statements.

3. CATALOGED PROCEDURE LISTING

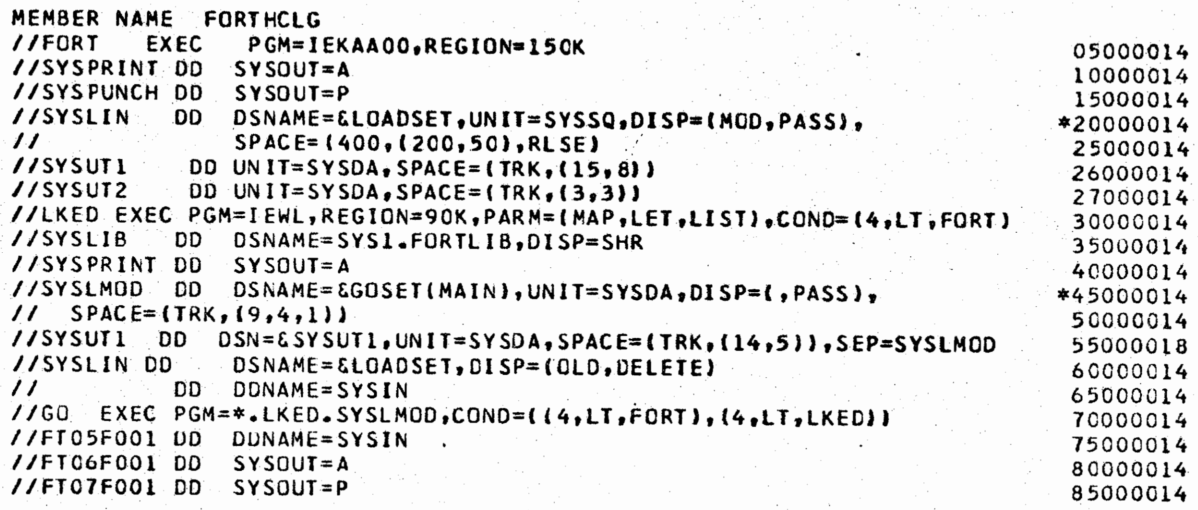
3.1 FORTGCLG

```
MEMBER NAME  FORTGCLG
//*  ALIAS NAME FOR FORTGCLG IS FORTECLG.                                   00000005
//FORT   EXEC  PGM=IEYFORT,REGION=100K,DPRTY=2                               00000010
//SYSPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=1080)  0568       00000020
//SYSLIN DD DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                    X00000030
//             SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)     00000040
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET',COND=(4,LT,FORT),REGION=90K       00000050
//SYSLIB    DD  DSNAME=SYS1.FORTLIB,DISP=SHR                                00000060
//SYSLMOD   DD  DSNAME=&GOSET(MAIN),DISP=(NEW,PASS),UNIT=SYSDA,            X00000070
//             SPACE=(TRK,(4,2,1))                                          00000080
//SYSPRINT DD SYSOUT=A                                                      00000090
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(4,2),RLSE)                             00000100
//SYSLIN    DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                           00000110
//          DD  DDNAME=SYSIN                                                00000120
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((4,LT,FORT),(4,LT,LKED)),DPRTY=2         00000130
//FT05F001   DD  DDNAME=SYSIN                                               00000140
//FT06F001   DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)            00000150
//FT07F001   DD  SYSOUT=P                                                   00000160
```

FORTGCLG/FORTHCLG - FORTRAN COMPILE, LINKAGE EDIT, AND EXECUTE (CONT.)

3. CATALOGUED PROCEDURE LISTING (Cont.)

3.2 FORTHCLG



```
MEMBER NAME  FORTHCLG
//FORT    EXEC   PGM=IEKAA00,REGION=150K                                            05000014
//SYSPRINT DD   SYSOUT=A                                                            10000014
//SYSPUNCH DD   SYSOUT=P                                                            15000014
//SYSLIN    DD  DSNAME=&LOADSET,UNIT=SYSSQ,DISP=(MOD,PASS),                        *20000014
//              SPACE=(400,(200,50),RLSE)                                           25000014
//SYSUT1     DD UNIT=SYSDA,SPACE=(TRK,(15,8))                                       26000014
//SYSUT2     DD UNIT=SYSDA,SPACE=(TRK,(3,3))                                        27000014
//LKED EXEC PGM=IEWL,REGION=90K,PARM=(MAP,LET,LIST),COND=(4,LT,FORT)                30000014
//SYSLIB    DD  DSNAME=SYS1.FORTLIB,DISP=SHR                                        35000014
//SYSPRINT  DD  SYSOUT=A                                                            40000014
//SYSLMOD   DD  DSNAME=&GOSET(MAIN),UNIT=SYSDA,DISP=(,PASS),                       *45000014
//  SPACE=(TRK,(9,4,1))                                                             50000014
//SYSUT1  DD  DSN=&SYSUT1,UNIT=SYSDA,SPACE=(TRK,(14,5)),SEP=SYSLMOD                 55000018
//SYSLIN DD     DSNAME=&LOADSET,DISP=(OLD,DELETE)                                   60000014
//          DD  DDNAME=SYSIN                                                        65000014
//GO  EXEC PGM=*.LKED.SYSLMOD,COND=((4,LT,FORT),(4,LT,LKED))                        70000014
//FT05F001 DD   DDNAME=SYSIN                                                        75000014
//FT06F001 DD   SYSOUT=A                                                            80000014
//FT07F001 DD   SYSOUT=P                                                            85000014
```

4. REFERENCE

IBM 360 OS FORTRAN IV Programmers Guide, GC28-6817

FORTLG - FORTRAN LINKAGE EDIT AND EXECUTE

1. GENERAL

FORTLG is a two step cataloged procedure to linkage edit an object or load module to produce a load module and execute it. The linkage edit step provides a reference to the automatic call library SYS1.FORTLIB. This will be effective for any FORTRAN compiler supported by the Data Center.

2. USING FORTLG

2.1 FORTLG may be invoked by an EXEC statement.

```
//   EXEC  FØRTLG
```

2.2 The following DD statements may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN   DD  (parameters describing linkage editor control input
                    and/or object module input)
        or

//LKED.ddname  DD  (parameters describing an additional call library)
```

2.3 Additional data sets required by the users program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //LKED.   DD statements.

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  FORTLG
//* ALIAS NAMES FOR FORTGLG ARE FORTELG AND FORTLG                        00010000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET',REGION=90K         BJI 8/72    00020000
//SYSLIB   DD  DSNAME=SYS1.FORTLIB,DISP=SHR                               00040000
//SYSLMOD  DD  DSNAME=&GOSET(MAIN),DISP=(NEW,PASS),UNIT=SYSDA,           X00060000
//             SPACE=(TRK,(4,2,1))                                        0C080000
//SYSPRINT DD SYSOUT=A                                                    00100000
//SYSUT1  DD UNIT=SYSDA,SPACE=(TRK,(4,2),RLSE)                            00120000
//SYSLIN   DD  DDNAME=SYSIN                                               00140000
//GO  EXEC  PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED),DPRTY=2                   00160000
//FT05F001  DD  DDNAME=SYSIN                                              00180000
//FT06F001  DD  SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=136)                       00200000
//FT07F001  DD  SYSOUT=P                                                  00220000
```

4. REFERENCE

IBM 360 OS FORTRAN IV Programmers Guide, GC28-6817
IBM 360 OS Linkage Editor and Loader, GC28-6538

LIBP - LIBRARIAN, PDS OUTPUT

1. GENERAL

This cataloged procedure invokes the Applied Data Research program LIBRARIN. If the EXEC option is specified, output is directed to a partitioned data set named &&PDS under the member name of the selected module. This permits the programmer to work with several modules in one execution of Librarian and yet maintain individual accessibility to output modules. Successive calls of the procedure within the job should not be necessary unless different masters are being referenced, however, should this be required, the same PDS, &&PDS, will be used for output.

2. SYMBOLIC PARAMETERS

MASTER This parameter supplies the dsname of the Librarian master file.

CYL This parameter controls the primary space allocation of the output file. It defaults to 1 to yield 1 cylinder. The file is a partitioned data set.

3. USING LIBP

3.1 LIBP is invoked with an EXEC statement.

```
// EXEC LIBP,MASTER='xxxxxx'
```

where xxxxxx is any valid dsname of a Librarian type master file.

3.2 The programmer places input statements, both Librarian control and data, after the EXEC card. If JCL statements are being stored, input must be preceded by //SYSIN DD DATA and followed by a /* statement.

4. EXAMPLE

In this example two modules are selected for compilation.

```
// EXEC LIBP,MASTER='DEV.SØURCE'
-ØPT
-SEL PRØGX,ZFLG,EXEC
-EMØD
-SEL PRØGY,MNLK,EXEC
-DEL 50,100
-EMØD
-END
// EXEC CØBUC
//SYSIN DD DSN=&&PDS(PRØGX),DISP=(ØLD,PASS)
// EXEC CØBUC
//SYSIN DD DSN=&&PDS(PRØGY),DISP=(ØLD,DELETE)
/*
```

LIBP - LIBRARIAN, PDS OUTPUT (CONT.)

5. CATALOGED PROCEDURE LISTING

```
//LIBP PROC MASTER=ERROR,CYL=1                                          00000100
//LIBP EXEC PGM=LIBRARIN,PARM='NRJS,NJTA',REGION=60K                    00000200
//MASTER DD DSN=&MASTER,DISP=SHR                                        00000300
//INDEX    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00000400
//LIST     DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00000500
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00000600
//OSJOB DD DSN=&PDS,DCB=BLKSIZE=960,UNIT=DISK,DISP=(MOD,PASS),          00000700
//          SPACE=(CYL,(&CYL,1,5))                                      00000800
//BACKUP DD DUMMY                                                       00000900
//SYSPUNCH DD SYSOUT=P,DCB=BLKSIZE=80                                   00001000
```

6. REFERENCE

6.1 Applied Data Research, The Librarian, User Reference Manual.
6.2 Division 3, Chapter 2, Section 3, LIBRARIAN.

LIBS - LIBRARIAN, SEQUENTIAL OUTPUT

1. GENERAL

This cataloged procedure invokes the Applied Data Research program, LIBRARIN. If the EXEC or UTILITY option is specified, the output is directed to a sequential file named &&SEQ. Successive calls of the procedure within the job should not be necessary unless different masters are being referenced, however, should this be required, the same sequential data set, &&SEQ, will be used for output with a disposition of MOD•

2. SYMBOLIC PARAMETERS

MASTER This parameter supplies the dsname of the Librarian master file.

CYL This parameter controls the primary space allocation of the output file. It defaults to 1 to yield 1 cylinder. The file is a sequential data set.

3. USING LIBS

3.1 LIBS is invoked with an EXEC statement.

```
// EXEC LIBS,MASTER='xxxxxx'
```

where xxxxxx is any valid dsname of a Librarian type master file.

3.2 The programmer places input statements, both Librarian control and data, after the EXEC card. If JCL statements are being stored, input must be preceded by //SYSIN DD DATA and followed by a /* statement.

4. EXAMPLE

In this example a single module is selected and updated for compilation.

```
// EXEC LIBS,MASTER='PRD5.SØURCE'
-ØPT
-SEL PRØGZ,ZXFL,EXEC,TEMP
-REP 60
    CØBØL statement
-EMØD
-END
// EXEC CØBUC
//SYSIN DD DSN=&&SEQ,DISP=(ØLD,DELETE)
/*
```

(Continued on next page)

LIBS - LIBRARIAN, SEQUENTIAL OUTPUT (CONT.)

5. CATALOGED PROCEDURE LISTING

```
//LIBS PROC MASTER=ERROR,CYL=1                                                  00000100
//LIBS EXEC PGM=LIBRARIN,PARM='NRJS,NJTA',REGION=60K                            00000200
//MASTER DD DSN=&MASTER,DISP=SHR                                                00000300
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)                    00000400
//LIST     DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)                    00000500
//INDEX    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)                    00000600
//OSJOB DD DSN=&SEQ,DCB=BLKSIZE=960,UNIT=DISK,DISP=(MOD,PASS),                  00000700
//           SPACE=(CYL,(&CYL,1))                                               00000800
//BACKUP DD DUMMY                                                               00000900
//SYSPUNCH DD SYSOUT=P,DCB=BLKSIZE=80                                           00001000
```

6. REFERENCE

6.1 Applied Data Research, The Librarian, User Reference Manual.

6.2 Division 3, Chapter 2, Section 3, LIBRARIAN.

LIBU - LIBRARIAN UTILITY PROCEDURE

1. GENERAL

This is a two-step Librarian cataloged procedure designed for transferring data between Librarian masters. The output, resulting from the UTILITY option, of the first step is used as input to the second step. The output of the second step is a partitioned data set named &PDS.

2. SYMBOLIC PARAMETERS

MASTER1	This parameter supplies the dsname of the Librarian master referenced in the first step.
CYL1	This parameter controls the primary space allocation of the output file of the first step. It defaults to 1 to yield 1 cylinder. The file is a sequential data set.
MASTER2	The parameter supplies the dsname of the Librarian master referenced in the second step.
CYL2	This parameter controls the primary space allocation of the output file of the second step. It defaults to 1 to yield 1 cylinder. The file is a partitioned data set.

3. USING LIBU

3.1 The procedure is invoked with an EXEC statement.

```
//    EXEC  LIBU,MASTER1='xxxxxx',MASTER2='yyyyyy'
```

where xxxxxx and yyyyyy are valid dsnames of Librarian type master files.

3.2 The following DD statement must be added to the procedure:

```
//UTIL.SYSIN  DD  *
  Librarian control statements and data
/*
```

(Continued on next page)

LIBU - LIBRARIAN UTILITY PROCEDURE (CONT.)

4. EXAMPLE

The following is a simple transfer of two source modules from one master to another.

```
// EXEC LIBU,MASTER1='DEV.LIBR1',MASTER2='PRD5.SØURCE'
//UTIL.SYSIN DD *
-ØPT UTILITY
-ØPT EXEC                        *
-ADD PRØGX,SEQ=CØBØL             *
-INC PRØGX
-EMØD                            *
-ADD PRØGY,SEQ=CØBØL             *
-INC PRØGY
-EMØD                            *
-END                             *
/*
// EXEC CØBUC
//SYSIN DD DSN=&&PDS(PRØGX),DISP=(ØLD,PASS)
// EXEC CØBUC
//SYSIN DD DSN=&&PDS(PRØGY),DISP=(ØLD,DELETE)
```

The -INC statements will be replaced by Librarian with the source statements of PROGX and PROGY. These source statements along with the starred (*) statements will form input to the second Librarian step and result in addition of the programs to the second master.

5. CATALOGED PROCEDURE LISTING

```
//LIBU PROC MASTER1=FRROR,MASTER2=ERROR,CYL1=1,CYL2=1                   00000100
//UTIL EXEC PGM=LIBRARIN,PARM='NRJS,NJTA',REGION=60K                    00000200
//MASTER DD DSN=&MASTER1,DISP=SHR                                       00000300
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00000400
//LIST     DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00000500
//INDEX    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00000600
//OSJOB DD DSN=&UTIL,DCB=BLKSIZE=960,UNIT=DISK,DISP=(,PASS),            00000700
//         SPACE=(CYL,(&CYL1,1))                                        00000800
//LIBP EXEC PGM=LIBRARIN,PARM='NRJS,NJTA',REGION=60K,COND=(0,NE,UTIL)   00000900
//MASTER DD DSN=&MASTER2,DISP=SHR                                       00001000
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRFCL=133,BLKSIZE=931)            00001100
//LIST     DD SYSOUT=A,DCE=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00001200
//INDEX    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=931)            00001300
//OSJOB    DD DSN=&PDS,DCB=BLKSIZE=960,UNIT=DISK,DISP=(MOD,PASS),       00001400
//            SPACE=(CYL,(&CYL2,1,5))                                   00001500
//SYSIN    DD DSN=&UTIL,DISP=(OLD,DELETE)                               00001600
//BACKUP   DD DUMMY                                                     00001700
//SYSPUNCH DD SYSOUT=P,DCB=BLKSIZE=80                                   00001800
```

6. REFERENCE

6.1 Applied Data Research, The Librarian, User Reference Manual.

6.2 Division 3, Chapter 2, Section 3, LIBRARIAN.

LIST - LISTING VTOC, PDS DIRECTORIES, CATALOGS

1. GENERAL

LIST is a cataloged procedure which facilitates use of the IBM Utility Program IEHLIST for listing catalogs, partitioned data set directories, or volume table of contents of the system residence volume. LIST provides DD statements for SYSPRINT and the system residence volume.

2. USING LIST

2.1 The LIST procedure is invoked with an EXEC statement coded:

```
//   EXEC LIST
```

2.2 The user must include a SYSIN DD statement and appropriate IEHLIST control statements.

```
//SYSIN  DD  *
  control cards
```

3. EXAMPLE OF USE

```
//   EXEC LIST
//SYSIN  DD  *
  LISTCTLG
/*
```

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  LIST
//LIST    EXEC   PGM=IEHLIST,REGION=44K                                       00020000
//DDSRV   DD     VOLUME=REF=SYS1.SVCLIB,DISP=OLD                              00040000
//SYSPRINT DD SYSOUT=A                                                        00060000
```

5. REFERENCE

IBM OS Utilities GC28-6586, IEHLIST

LISTDISK - LISTING CONTENTS OF A DIRECT ACCESS VOLUME AND ATTRIBUTES OF DIRECT ACCESS DATA SETS

1. GENERAL

LISTDISK is a cataloged procedure which facilitates use of a Western Electric Utility Program,SLIST,to list contents of a direct access volume or to provide information about the attributes of all or selected disk data sets on a selected direct access volume.

2. SYMBOLIC PARAMETERS

LISTDISK requires a VOLID parameter which supplies the volume serial number of the disk pack to be selected.

3. USING LISTDISK

3.1 The LISTDISK procedure is invoked with an EXEC statement which includes the symbolic parameter, VOLID.

```
//    EXEC LISTDISK,VØLID=serial
```

3.2 A group of data sets may be selected by specifying a four character PARM value representing the first 4 characters of the DSNAME's to be listed.

```
//    EXEC LISTDISK,VØLID=serial,PARM=index
```

4. EXAMPLE OF USE

Example 1: // EXEC LISTDISK,VØLID=STØR31
Result: All data sets on STØR31 will be listed

Example 2: // EXEC LISTDISK,VØLID=SYSRES,PARM=SYS1
Result: All data sets on a volume identified as SYSRES and having SYS1 as the first 4 characters of DSNAME will be listed.

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  LISTDISK
//IEFPROC EXEC PGM=SLIST,REGION=30K                                         00000100
//* OPERATOR START = 'S LISTDISK,,VOLID'  VOLID=DISK PACK NAME              00000200
//SYSPRINT DD SYSOUT=A                                          BJI 9/72    00000300
//DD1       DD DDNAME=IEFRDER                                               00000400
//IEFRDER DD UNIT=SYSDA,DISP=OLD,VOL=SER=&VOLID                             00000500
```

6. REFERENCE

Division 3, Chapter 2, Section 2, SLIST Utility Program

LISTOUT - LISTING A MEMBER OF A CARD IMAGE PARTITIONED DATA SET

1. GENERAL

LISTOUT is a cataloged procedure which invokes the Western Electric Utility Program WECOPY to list a member of a card image partitioned data set. The library must be a cataloged data set.

2. SYMBOLIC PARAMETERS

LISTOUT requires two symbolic parameters and may have two optional parameters, which are:

PDS, a required parameter, supplies the library name. Enclose in apostrophes if the name is indexed (contains periods).
MEMBER, a required parameter, supplies the name of the member to be printed.
MAX, an optional parameter, supplies to the maximum number of records to be printed.
SKIP, an optional parameter, supplies the maximum number of records to be skipped before printing begins.

3. USING LISTOUT

The LISTOUT procedure is invoked with an EXEC statement which includes the two required symbolic parameters, PDS and MEMBER, and the optional parameters, MAX and SKIP, if used.

```
//   EXEC LISTØUT,PDS='library',MEMBER=prog,[MAX=nnn],[SKIP=nnn]
```

4. EXAMPLE OF USE

```
//   EXEC LISTØUT,PDS='SYS1.PRØCLIB',MEMBER=CØBUCLG
```

5. SUGGESTED USE

LISTOUT is recommended for listing procedure, control and "parmlib" members.

6. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  LISTOUT
//           PROC  MAX=9999,SKIP=0                                      00000010
//LIST       EXEC  PGM=WECOPY,PARM=(&MAX,&SKIP)                         00000020
//SYSPRINT DD SYSOUT=A                                                  00000030
//SYSUT1  DD  DSN=&PDS.(&MEMBER),DISP=SHR                               00000040
//SYSUT2 DD SYSOUT=A                                                    00000050
```

7. REFERENCE

Division 3, Chapter 2, Section 2, WECOPY Utility Program

LISTPDS - LISTING A PARTITIONED DATA SET

1. GENERAL

LISTPDS is a cataloged procedure which facilitates use of the IBM Utility Program IEBPTPCH for listing the content of all members of a partitioned data set.

2. SYMBOLIC PARAMETER

LISTPDS requires a DSN parameter which supplies the library name. The parameter must be enclosed in apostrophes if the name is indexed (contains periods).

3. USING LISTPDS

LISTPDS is invoked with an EXEC statement which includes the symbolic parameter, DSN.

```
//    EXEC LISTPDS,DSN='dataset'
```

4. EXAMPLE OF USE

```
//    EXEC LISTPDS,DSN='PRD7.PARMLIB'
```

5. RESTRICTION

The library must be cataloged and have LRECL=80.

6. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  LISTPDS
//IEFPROC EXEC PGM=IEBPTPCH,REGION=60K                                00000100
//SYSPRINT DD   SYSOUT=A                                              00000200
//SYSUT1    DD DDNAME=IEFRDER                                         00000300
//IEFRDER   DD DSN=&DSN,DISP=SHR                                      00000400
//SYSUT2   DD   SYSOUT=A                                              00000500
//SYSIN     DD DSN=SYS1.CONTROL(LISTPDS),DISP=SHR                     00000600
```

7. REFERENCE

IBM OS Utilities GC28-6586, IEBPTPCH

LKED - LINK EDIT AND PRODUCE A LOAD MODULE

1. GENERAL

LKED is a one-step cataloged procedure that link edits an input data set, produces a load module, and passes the load module to another step in the same job.

2. USING LKED

2.1 LKED may be invoked by an EXEC statement.

```
//    EXEC  LKED
```

2.2 The following DD statement must be added to the procedure.

```
//LKED.SYSIN  DD  *
  (Object module deck(s) and/or control statements)
         or
//LKED.SYSIN  DD  (appropriate parameters defining a source
                   input data set)
```

2.3 Reference to the automatic call library must be provided by the user through SYSLIB DD statements, e.g. SYS1.COBLIB, SYS1.FORTLIB, etc.

```
//SYSLIB  DD  DSN=SYS1.library,DISP=SHR
```

2.4 Additional DD statements may be added to supply input to the linkage editor.

```
//LKED.ddname  DD  (appropriate parameters)
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  LKED
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LET,LIST',REGION=90K                00020000
//SYSPRINT  DD  SYSOUT=A                                                00040000
//SYSLIN    DD DDNAME=SYSIN                                             00060000
//SYSLMOD DD DSN=&GOSET(GO),SPACE=(TRK,(15,,1)),                        00080000
//            UNIT=SYSDA,DISP=(MOD,PASS)                                00100000
//SYSUT1    DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),                      X00120000
//            SPACE=(TRK,(14,5))                                        00140000
```

4. REFERENCE

IBM 360 OS Linkage Editor and Loader, GC28-6538

LKEDG - LINK EDIT, PRODUCE A LOAD MODULE, AND EXECUTE

1. GENERAL

LKEDG is a two-step cataloged procedure that link edits an input data set, produces a load module, and executes that load module.

2. USING LKEDG

2.1 LKEDG may be invoked by an EXEC statement.

```
//    EXEC  LKEDG
```

2.2 The following DD statement must be added to the procedure.

```
//LKED.SYSIN  DD  *
  (Object module decks and/or control statements)
        or
//LKED.SYSIN  DD  (appropriate parameters defining a source
                   input data set)
```

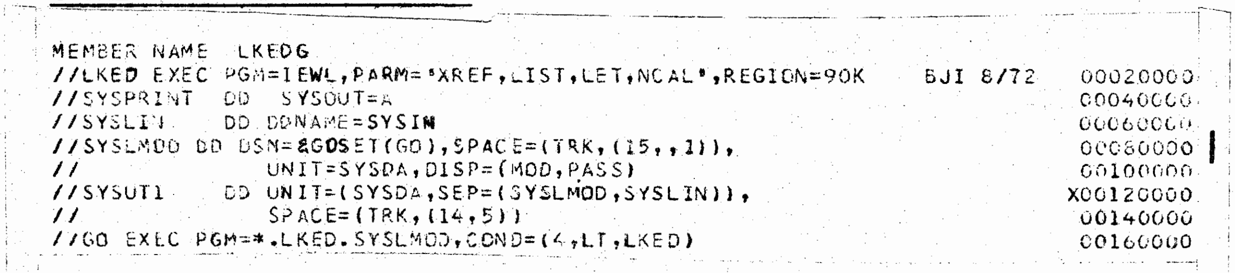
2.3 Additional DD statements may be added to supply input to the linkage editor.

```
//LKED.ddname  DD  (appropriate parameters)
```

2.4 Additional DD statements may be added for the GO step.

```
//GØ.ddname   DD  (appropriate parameters)
```

3. CATALOGED PROCEDURE LISTING



```
MEMBER NAME  LKEDG
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET,NCAL',REGION=90K    BJI 8/72    00020000
//SYSPRINT  DD  SYSOUT=A                                                 00040000
//SYSLIN    DD DDNAME=SYSIN                                              00060000
//SYSLMOD DD DSN=&GOSET(GO),SPACE=(TRK,(15,,1)),                         00080000
//             UNIT=SYSDA,DISP=(MOD,PASS)                                00100000
//SYSUT1    DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),                       X00120000
//             SPACE=(TRK,(14,5))                                        00140000
//GO EXEC PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED)                            00160000
```

4. REFERENCE

IBM 360 OS Linkage Editor and Loader, GC28-6538

MOD - CATALOG OPERATIONS, RENAMING, SCRATCHING

1. GENERAL

MOD is a cataloged procedure which facilitates use of the IBM Utility Program IEHPROGM for scratching, renaming, and performing catalog operations on direct access volumes. MOD provides DD statements for SYSPRINT and the resident catalog pack.

2. USING MOD

The MOD procedure is invoked with an EXEC statement, a DD statement defining the volume, and IEHPROGM control statements.

```
//  EXEC  MOD
//ddname  DD  UNIT=DISK,VOL=SER=------,DISP=SHR
  control statements
```

3. EXAMPLE OF USE

```
//  EXEC  MOD
//DD1  DD  UNIT=DISK,VOL=SER=TEST05,DISP=SHR
  SCRATCH DSNAME=USERS.DATASET,VOL=3330=TEST05
/*
```

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  MOD
//MOD  EXEC  PGM=IEHPROGM,REGION=44K                                     00020000
//DDSRV  DD  VOL=REF=SYS1.SVCLIB,DISP=SHR                                00040000
//SYSPRINT DD SYSOUT=A,SPACE=(TRK,(5,5))                    69160-EWR    00060000
```

5. REFERENCE

IBM OS Utilities GC28-6586, IEHPROGM

MOVE - COPYING DIRECT ACCESS FILES, CATALOGS, PDS MEMBERS

1. GENERAL

MOVE is a cataloged procedure which facilitates use of the IBM Utility Program IEHMOVE for performing disk to disk operations such as copying data sets or catalogs. MOVE provides DD statements for SYSPRINT and three disk packs.

2. SYMBOLIC PARAMETERS

MOVE requires two symbolic parameters and may have an optional parameter, which are:

VOL1, a required parameter, supplies the source volume serial number.
VOL2, a required parameter, supplies the receiving volume serial number.
VOL3, an optional parameter, supplies a work volume serial number.
The default is UTIL31.

3. USING MOVE

The MOVE procedure is invoked with an EXEC DD statement which includes the symbolic parameters, VOL1, VOL2, and optionally, VOL3, followed by IEHMOVE control statements.

```
//    EXEC MØVE,VØL1=serial,VØL2=serial,[VØL3=serial]
  control statements
```

4. EXAMPLE OF USE

```
//  EXEC MØVE,VØL1=TEST05,VØL2=TEST50
  CØPY DSNAME=DEV05.ABC.MASTR,FRØM=3330=TEST05,TØ=3330=TEST50
/*
```

5. PRECAUTION

The user should be aware of the differences between a move operation and a copy operation. A move operation will result in the deletion of the source data set or library even though only selected members are moved while a copy operation leaves source data set intact. In addition, for cataloged data sets, a move operation updates the catalog to refer to the moved version (unless otherwise specified), while a copy operation leaves the catalog unchanged. COPY is the preferred method of data set transfer or member transfer.

MOVE - COPYING DIRECT ACCESS FILES, CATALOGS, PDS MEMBERS (CONT.)

6. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  MOVE
//   PROC   VOL3=UTIL31                                              00000010
//IEFPROC EXEC PGM=IEHMOVE                                           00000020
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=121,BLKSIZE=968)          00000030
//SYSUT1 DD UNIT=SYSDA,DISP=OLD,VOL=SER=&VOL3                        00000040
//VOL1 DD UNIT=SYSDA,DISP=OLD,VOL=SER=&VOL1                          00000050
//VOL2 DD UNIT=SYSDA,DISP=OLD,VOL=SER=&VOL2                          00000060
```

7. REFERENCE

IBM OS Utilities C28-6586, IEHMOVE
Also see PLCPREP cataloged procedure.

Reason for reissue: As indicated and former paragraph 6 deleted.

OPRPNCH - OBTAIN COPY OF CONSOLE JCL DECKS

1. GENERAL

OPRPNCH is a cataloged procedure which may be used to obtain a copy of JCL console decks that reside in PROD.JCLLIB and are accessed by the internal reader.

2. USING OPRPNCH

The OPRPNCH procedure is invoked with an EXEC statement coded:

```
//  EXEC ØPRPNCH,JØB=XXXXXXXX
```

There must be one EXEC card for each job JCL deck desired.

3. ASP REQUIREMENTS

The following ASP control card must be supplied by the user:

```
//*MAIN CARDS=n
```

where n represents the number of cards, in hundreds, to be punched.

4. CATALOGED PROCEDURE LISTING

```
//OPRPNCH PROC DSN='PROD.JCLLIB'                                       00000010
//PUNCHIT  EXEC PGM=WECOPY                                              00000020
//SYSPRINT  DD  SYSOUT=A                                                00000030
//SYSUT1  DD  DSN=&DSN(&JOB),DISP=SHR                                   00000040
//SYSUTP DD SYSOUT=P,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)                  00000050
```

5. REFERENCE

Division 2, Chapter 5, Section 1

Reason for Reissue: As indicated and reference to ASP SETUP card deleted.

PLCPREP - PREPARATION FOR UPDATE OF PRODUCTION LIBRARIES

1. GENERAL

PLCPREP is a cataloged procedure which allows the user to obtain a "copy" of a production library member, make modifications as needed, and test the updated version without disturbing the production library. It invokes the IBM Utility Program IEBUPDTE. The data sets referenced must be cataloged.

2. SYMBOLIC PARAMETER

PLCPREP requires two symbolic parameters, "FROM" and "TO", which supply the library names. The parameters must be enclosed in apostrophes if the names are indexed (contain periods).

3. USING PLCPREP

3.1 The PLCPREP procedure is invoked with an EXEC statement which includes the symbolic parameters, FROM and TO.

```
//   EXEC PLCPREP,FRØM='from.lib',TØ='to.lib'
```

where: FROM = the library from which the data is extracted and then updated and written on the data set specified by the TO = parameter.

3.2 The user must include a SYSIN DD statement and appropriate IEBUPDTE control statements.

```
//SYSIN  DD  DATA
   control statements
```

4. EXAMPLE OF USE

```
//   EXEC PLCPREP,FRØM='PRØD.JCLLIB',TØ='PLC.JCLLIB'
//SYSIN  DD  DATA
./ CHANGE NAME=TLSLIST,LIST=ALL
   user JCL changes
/*
```

5. RESTRICTION

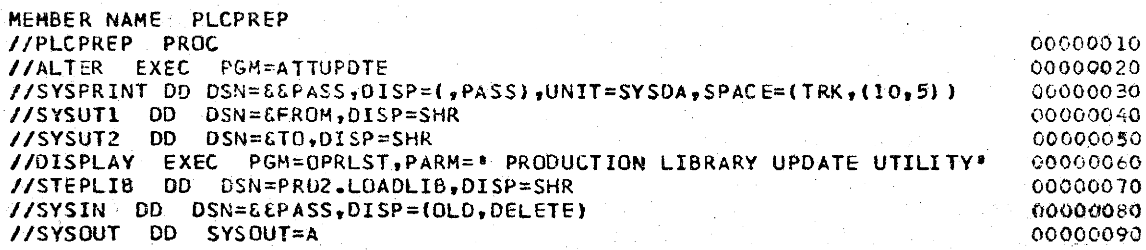
The libraries must be cataloged and have LRECL=80.

6. SUGGESTED USE

PLCPREP is recommended for updating libraries such as procedure, JCL or control libraries from card input.

PLCPREP - PREPARATION FOR UPDATE OF PRODUCTION LIBRARIES (CONT.)

7. CATALOGED PROCEDURE LISTING



```
MEMBER NAME  PLCPREP
//PLCPREP  PROC                                                          00000010
//ALTER   EXEC  PGM=ATTUPDTE                                             00000020
//SYSPRINT DD DSN=&&PASS,DISP=(,PASS),UNIT=SYSDA,SPACE=(TRK,(10,5))      00000030
//SYSUT1   DD   DSN=&FROM,DISP=SHR                                       00000040
//SYSUT2   DD   DSN=&TO,DISP=SHR                                         00000050
//DISPLAY  EXEC  PGM=OPRLST,PARM=' PRODUCTION LIBRARY UPDATE UTILITY'    00000060
//STEPLIB  DD  DSN=PRD2.LOADLIB,DISP=SHR                                 00000070
//SYSIN  DD  DSN=&&PASS,DISP=(OLD,DELETE)                                00000080
//SYSOUT   DD   SYSOUT=A                                                 00000090
```

8. REFERENCE

IBM OS Utilities GC28-6586, IEBUPDTE
CHGPDS Cataloged Procedure
Division 2, Chapter 5, Section 2

PL1LFC - PL/I COMPILE

1. GENERAL

PL1LFC is a one-step catalogued procedure to compile a PL/I source program. It does not normally produce an object module.

2. USING PL1LFC

2.1 PL1LFC may be invoked by an EXEC statement.

```
// EXEC PL1LFC
```

2.2 The following DD statement must be added to the procedure.

```
//PL1L.SYSIN  DD  *
  (PL/I source deck)
         or
//PL1L.SYSIN  DD  (appropriate parameters defining a
                   source input data set)
```

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  PL1LFC
//PL1L EXEC PGM=IEMAA,PARM='NCLOAD,NODECK',REGION=90K                    10000000
//SYSPRINT DD DCNAME=PL1OUT                                               20000000
//PL1CUT DD SYSOUT=A,DCB=(RECFM=VB,LRECL=125,BLKSIZE=1254)                20000005
//SYSLIN    DD   DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,             *30000000
//               SPACE=(80,(250,100)),DCB=BLKSIZE=80                      40000000
//SYSUT3    DD   UNIT=SYSDA,SPACE=(80,(250,250)),SEP=SYSPRINT,           *50000000
//               DCB=BLKSIZE=80                                           6C000000
//SYSUT1    DD   UNIT=SYSDA,SPACE=(1024,(60,60),,CONTIG),                *7C000000
//               SEP=(SYSUT3,SYSPRINT,SYSLIN),DCB=BLKSIZE=1024            80000000
```

4. REFERENCE

IBM 360 OS PL/I(F) Programmers Guide, GC28-6594

PL1LFCL - PL/I COMPILE AND LINKAGE EDIT

1. GENERAL

PL1LFCL is a two-step cataloged procedure to compile and linkage edit a PL/I source program. Programmers will generally use this procedure when they wish to retain a load module in the load library specified by the SYSLMOD statement in the linkage edit step.

2. USING PL1LFCL

2.1 The PL1LFCL procedure is invoked with an EXEC statement coded:

```
//    EXEC   PL1LFCL
```

2.2 The following DD statements must be added to the procedure.

2.21 A PL1L.SYSIN DD statement defining the input.

```
//PL1L.SYSIN  DD  *
  (PL/I source deck)
         or
//PL1L.SYSIN  DD  (parameters defining a source input
                   data set)
```

2.22 A LKED.SYSLMØD DD statement to define a storage place for the load module.

```
//LKED.SYSLMØD  DD  (parameters describing the load module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //PL1L. DD statements.

3. EXAMPLE OF USE

```
//   EXEC  PL1LFCL
//PL1L.SYSIN  DD  *
  PL/I source deck
//LKED.SYSLMØD  DD DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
/*
```

(Continued on Next Page)

PL1LFCL - PL/I COMPILE AND LINKAGE EDIT (CONT.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PL1LFCL
//PL1L      EXEC PGM=IEMAA,PARM='LOAD,NODECK',REGION=90K                      00000010
//SYSPRINT DD DDNAME=PL1OUT                                                   00000020
//PL1OUT DD SYSOUT=A,DCB=(RECFM=VB,LRECL=125,BLKSIZE=1254)                    00000025
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,                    00000030
//              SPACE=(400,(100,50)),DCB=BLKSIZE=400                          00000040
//SYSUT3    DD  UNIT=SYSDA,SPACE=(80,(250,250)),SEP=SYSPRINT,                *00000050
//              DCB=BLKSIZE=80                                                00000060
//SYSUT1    DD  UNIT=SYSDA,SPACE=(1024,(60,60),,CONTIG),                     *00000070
//              SEP=(SYSUT3,SYSPRINT,SYSLIN),DCB=BLKSIZE=1024                 00000080
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST',COND=(16,EQ,PL1L),                   00000090
//           REGION=90K                                                       00000100
//SYSLIB    DD  DSNAME=SYS1.PL1LIB,DISP=SHR                                   00000110
//SYSUT1   DD  UNIT=SYSDA,SEP=SYSLIB,                                         00000120
//  SPACE=(TRK,(14,5))                                                        00000130
//SYSPRINT DD  SYSOUT=A                                                       00000140
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                              00000150
//          DD  DDNAME=SYSIN                                                  00000160
```

5. REFERENCE

IBM 360 OS PL/I(F) Programmers' Guide, GC28-6594

PL1LFCLG - PL/I COMPILE, LINKAGE EDIT, AND EXECUTE

1. GENERAL

PL1LFCLG is a three-step catalogued procedure to compile, linkage edit, and execute a PL/I source program.

2. USING PL1LFCLG

2.1 PL1LFCLG may be invoked by an EXEC statement.

```
// EXEC PL1LFCLG
```

2.2 The following DD statement must be added to the procedure.

```
//PL1L.SYSIN  DD  *
  (PL/I source deck)
        or
//PL1L.SYSIN  DD  (appropriate parameters defining
                   a source input data set)
```

2.3 In order to supply additional input to the linkage editor, the following DD statement may be added.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //PL1L. DD statements.

2.4 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //PL1L. DD statements, all //LKED. DD statements, and any overriding //GØ. DD statements.

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PL1LFCLG
//PL1L      EXEC PGM=IEMAA,PARM='LOAD,NODECK',REGION=90K                  04000000
//SYSPRINT DD DDNAME=PL1OUT                                               08000000
//PL1OUT DD SYSOUT=A,DCB=(RECFM=VB,LRECL=125,BLKSIZE=1254)                08000005
//SYSLIN   DD   DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,               12000000
//               SPACE=(400,(100,50)),DCB=BLKSIZE=400                     16000000
//SYSUT3     DD  UNIT=SYSDA,SPACE=(80,(250,250)),SEP=SYSPRINT,           *20000000
//               DCB=BLKSIZE=80                                           24000000
//SYSUT1     DD  UNIT=SYSDA,SPACE=(1024,(60,60),,CONTIG),                *28000000
//               SEP=(SYSUT3,SYSPRINT,SYSLIN),DCB=BLKSIZE=1024            32000000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST',COND=(16,EQ,PL1L),                  36000000
// REGION=90K                                                             40000000
//SYSLIB     DD  DSNAME=SYS1.PL1LIB,DISP=SHR                              44000000
//SYSLMOD    DD  DSN=&GOSET(GO),DISP=(MOD,PASS),                          48000000
//  UNIT=SYSDA,SPACE=(TRK,(9,4,1))                                        52000000
//SYSUT1   DD  UNIT=SYSDA,                                                56000000
//  SPACE=(TRK,(14,5))                                                    60000000
//SYSPRINT DD  SYSOUT=A                                                   64000000
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                          68000000
//           DD  DDNAME=SYSIN                                             76000000
//GO       EXEC  PGM=*.LKED.SYSLMOD,COND=((9,LT,LKED),(11,LT,PL1L))       80000000
//SYSPRINT DD  SYSOUT=A                                                   84000000
```

PL1LFCLG - PL/I COMPILE, LINKAGE EDIT, AND EXECUTE (CONT.)

4. REFERENCE

IBM 360 OS PL/I(F) Programmers Guide, GC28-6594

PL1LFLG - PL/I LINKAGE EDIT AND EXECUTE

1. GENERAL

PL1LFLG is a two-step cataloged procedure to linkage edit an object or load module to produce a load module and execute it. The linkage edit step provides a reference to the automatic call library SYS1.PL1LIB.

2. USING PL1LFLG

2.1 PL1LFLG may be invoked by an EXEC statement.

```
//     EXEC    PL1LFLG
```

2.2 The following DD statements may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN    DD    (parameters describing linkage editor control input
                        and/or object module input)
         or

//LKED.ddname   DD    (parameters describing an additional call library)
```

2.3 Additional data sets required by the users program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //LKED.   DD statements.

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PL1LFLG
//LKED EXEC PGM=IEWL,PARM='XREF,LIST',REGION=90K                        09000000
//SYSLIB   DD  DSNAME=SYS1.PL1LIB,DISP=SHR                              18000000
//SYSLMOD  DD  DSN=&&GOSET(GO),DISP=(MOD,PASS),                         27000000
//  UNIT=SYSDA,SPACE=(TRK,(9,4,1))                                      36000000
//SYSUT1   DD  UNIT=SYSDA,SEP=(SYSLMOD,SYSLIB),                         45000000
//  SPACE=(TRK,(14,5))                                                  54000000
//SYSPRINT DD  SYSOUT=A                                                 63000000
//SYSLIN   DD  DDNAME=SYSIN                                             72000000
//GO      EXEC  PGM=*.LKED.SYSLMOD,COND=(9,LT,LKED)                     81000000
//SYSPRINT DD  SYSOUT=A                                                 90000000
```

4. REFERENCE

IBM 360 OS PL/I (F) Programmers Guide, GC28-6594
IBM 360 OS Linkage Editor and Loader, GC28-6538

RPGEC - RPG COMPILE

1. GENERAL

RPGEC is a one-step catalogued procedure to compile an RPG source program. It does not normally produce an object module.

2. USING RPGEC

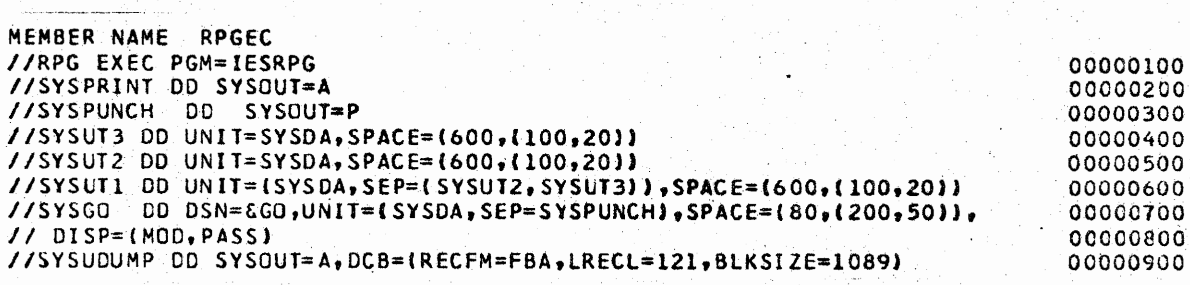
2.1 RPGEC may be invoked by an EXEC statement.

```
// EXEC RPGEC
```

2.2 The following DD statement must be added to the procedure.

```
//RPG.SYSIN  DD  *
  (RPG source deck)
          or
//RPG.SYSIN  DD  (appropriate parameters defining
                  a source input data set)
```

3. CATALOGUED PROCEDURE LISTING



```
MEMBER NAME  RPGEC
//RPG EXEC PGM=IESRPG                                                        00000100
//SYSPRINT DD SYSOUT=A                                                       00000200
//SYSPUNCH  DD  SYSOUT=P                                                     00000300
//SYSUT3 DD UNIT=SYSDA,SPACE=(600,(100,20))                                  00000400
//SYSUT2 DD UNIT=SYSDA,SPACE=(600,(100,20))                                  00000500
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSUT2,SYSUT3)),SPACE=(600,(100,20))            00000600
//SYSGO  DD DSN=&GO,UNIT=(SYSDA,SEP=SYSPUNCH),SPACE=(80,(200,50)),           00000700
// DISP=(MOD,PASS)                                                           00000800
//SYSUDUMP DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)                00000900
```

4. REFERENCE

IBM 360 OS RPG Specifications, GC24-3337

RPGECL - RPG COMPILE AND LINKAGE EDIT

1. GENERAL

RPGECL is a two-step cataloged procedure to compile and linkage edit an RPG source program. Programmers will generally use this procedure when they wish to retain a load module in the load library specified by the SYSLMOD statement in the linkage edit step.

2. USING RPGECL

2.1 The RPGECL procedure is invoked with an EXEC statement coded:

```
//    EXEC   RPGECL
```

2.2 The following DD statements must be added to the procedure.

2.21 An RPG.SYSIN DD statement defining the input.

```
//RPG.SYSIN  DD  *
  (RPG source deck)
         or
//RPG.SYSIN  DD  (parameters defining a source input
                  data set)
```

2.22 A LKED.SYSLMØD DD statement to define a storage place for the load module.

```
//LKED.SYSLMØD  DD  (parameters describing the load module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //RPG. DD statements.

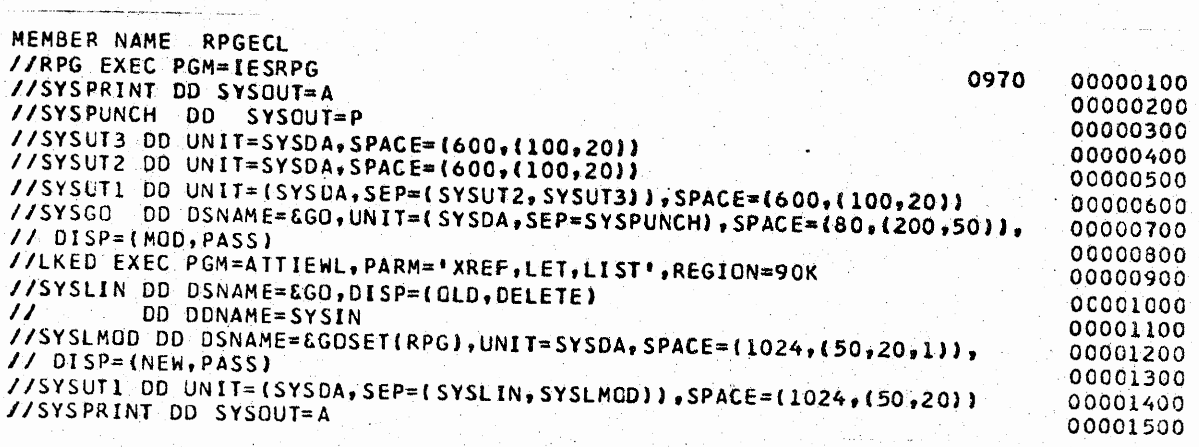
3. EXAMPLE OF USE

```
//    EXEC   RPGECL
//RPG.SYSIN  DD  *
  RPG source deck
//LKED.SYSLMØD  DD DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
/*
```

(Continued on Next Page)

RPGECL - RPG COMPILE AND LINKAGE EDIT (CONT.)

4. CATALOGED PROCEDURE LISTING



```
MEMBER NAME  RPGECL
//RPG EXEC PGM=IESRPG                                                    0970   00000100
//SYSPRINT DD SYSOUT=A                                                          00000200
//SYSPUNCH  DD  SYSOUT=P                                                        00000300
//SYSUT3 DD UNIT=SYSDA,SPACE=(600,(100,20))                                     00000400
//SYSUT2 DD UNIT=SYSDA,SPACE=(600,(100,20))                                     00000500
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSUT2,SYSUT3)),SPACE=(600,(100,20))               00000600
//SYSGO  DD DSNAME=&GO,UNIT=(SYSDA,SEP=SYSPUNCH),SPACE=(80,(200,50)),           00000700
// DISP=(MOD,PASS)                                                              00000800
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LET,LIST',REGION=90K                         00000900
//SYSLIN DD DSNAME=&GO,DISP=(OLD,DELETE)                                        0C001000
//         DD DDNAME=SYSIN                                                      00001100
//SYSLMOD DD DSNAME=&GOSET(RPG),UNIT=SYSDA,SPACE=(1024,(50,20,1)),              00001200
// DISP=(NEW,PASS)                                                              00001300
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(1024,(50,20))              00001400
//SYSPRINT DD SYSOUT=A                                                          00001500
```

5. REFERENCE

IBM 360 OS RPG Specifications, GC24-3337

RPGECLG - RPG COMPILE, LINKAGE EDIT, AND EXECUTE

1. GENERAL

RPGECLG is a three-step catalogued procedure to compile, linkage edit, and execute an RPG source program.

2. USING RPGECLG

2.1 RPGECLG may be invoked by an EXEC statement.

```
// EXEC RPGECLG
```

2.2 The following DD statement must be added to the procedure.

```
//RPG.SYSIN  DD  *
  (RPG source deck)
          or
//RPG.SYSIN  DD  (appropriate parameteɛs defining
                  a source input data set)
```

2.3 In order to supply additional input to the linkage editor, the following DD statement may be added.

```
//LKED.SYSIN  DD  (appropriate parameters)
```

This statement must follow all //RPG. DD statements.

2.4 Additional data sets required by the source program may be specified on DD statements.

```
//GØ.ddname  DD  (appropriate parameters)
```

These statements must follow all //RPG. DD statements, all //LKED. DD statements, and any overriding //GØ. DD statements.

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  RPGECLG
//RPG  EXEC  PGM=IESRPG                                              0568   00020000
//SYSPRINT DD SYSOUT=A                                                      00040000
//SYSPUNCH  DD  SYSOUT=P                                                    00060000
//SYSUT3 DD UNIT=SYSDA,SPACE=(600,(100,20))                                 00080000
//SYSUT2 DD UNIT=SYSDA,SPACE=(600,(100,20))                                 00100000
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSUT2,SYSUT3)),SPACE=(600,(100,20))           00120000
//SYSGO DD DSN=&GO,UNIT=(SYSDA,SEP=SYSPUNCH),SPACE=(80,(200,50)),           00140000
//              DISP=(MOD,PASS)                                             00160000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET',REGION=90K               BJI 8/72 00180000
//SYSLIN    DD DSNAME=&GO,DISP=(OLD,DELETE)                                 00200000
//          DD DDNAME=SYSIN                                                 00220000
//SYSLMOD   DD DSNAME=&GOSET(RPG),UNIT=SYSDA,SPACE=(1024,(50,20,1)),       X00240000
//              DISP=(NEW,PASS)                                             0C260000
//SYSUTI    DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(1024,(50,20))       00280000
//SYSPRINT  DD  SYSOUT=A                                                    00300000
//GO        EXEC PGM=*.LKED.SYSLMOD,COND=((9,LT,RPG),(5,LT,LKED))           0C320000
//SYSUDUMP  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)      0568   00340000
```

RPGECLG - RPG COMPILE, LINKAGE EDIT, AND EXECUTE (CONT.)

4. REFERENCE

IBM 360 OS RPG Specifications, GC24-3337

SORTSPAC - CALCULATE DISK SORT WORK SPACE

1. GENERAL

SORTSPAC is a cataloged procedure which facilitates use of a Western Electric Utility program of the same name, SORTSPAC, for calculating disk sort work space. The computations performed are based on formulas given in the IBM OS SORT/MERGE Programmers Guide SC33-4007.

2. CONTROL CARD PARAMETERS

SORTSPAC requires a control card coded as follows:

```
(beginning in column 1) LLLLL,SSSSSSSS,NN,DASD
where  L is the logical record length including leading zeros
       S is the record count including leading zeros
       N is the number of work areas (default=06)
    DASD is 2314 or 3330 (default=3330)
```

3. USING SORTSPAC

The SORTSPAC procedure is invoked with an EXEC statement and appropriate control cards.

```
//  EXEC SØRTSPAC
control cards (one or more cards)
```

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  SORTSPAC
//SORTSPAC PROC                                                         00000010
//SPAC EXEC PGM=SORTSPAC,REGION=30K                                     00000020
//SYSOUT DD SYSOUT=A                                                    00000030
//PARMS DD DDNAME=SYSIN                                                 00000040
```

5. REFERENCE

Division 3, Chapter 2, Section 2, SORTSPAC Utility Program.

WECT - CARD INPUT TO STANDARD LABEL TAPE WITH CATALOGING

1. GENERAL

WECT is a cataloged procedure which facilitates use of a Western Electric utility program of the same name, WECT, for the transfer of card input from the job input stream to a standard label tape and to catalog the data set under a user designated DSNAME.

2. WECT OPTIONS

2.1 WECT permits the user to specify one of three card-to-tape options, which are designed to meet the needs of systems interfacing with converted 7080 systems as well as the current generation systems. The options are governed by the symbolic parameter "REC" (Paragraph 3), which supplies a numeric value for the LRECL DCB parameter. The options are:

Option A - This is generally used in conjunction with converted 7080 systems and is equivalent to CTP-04, a 1401 card-to-tape program. Four blanks and a record mark character are generated at the end of each logical record. In addition, an incomplete block is filled with nines padding before the tape mark is written. Designating REC=85 selects this option. If this option is selected, the user must specify a numeric value for the BLK symbolic parameter (Paragraph 3).

Option B - The option was formerly used to create files for data transmission and was selected by designating REC=81 which resulted in the generation of a record mark character as the 81st byte of the output record.

Option C - The 80 column card image is copied as an 80 byte logical record, with blocking as specified in the "BLK" symbolic parameter. Designating REC=80 selects this option.

2.2 A message to SYSOUT-SYSUT3 indicates the option selected and the card count.

3. SYMBOLIC PARAMETERS

There are six symbol parameters which the user may employ to control the characteristics of his output file.

REG is a numeric value for REGION. The default is 30.

RETPD is the number of days the output tape is to be retained. The default is 1.

FM is the value for the RECFM parameter. The default is FB.

WECT - CARD INPUT TO STANDARD LABEL TAPE WITH CATALOGING (CONT.)

3. SYMBOLIC PARAMETERS (CONT.)

REC is the value for the LRECL parameter. The default is 80. Other values may be 85 and 81.

BLK is the value for the BLKSIZE parameter. The default is 800. This value must be a multiple of the REC parameter.

DSN is the dsname to be cataloged. This parameter must be supplied by the user or the job will fail. The name should be enclosed in apostrophes.

4. EXAMPLE OF USE

```
//CUT#CT01  JØB (P,12345,X),WN3499SMITH,CLASS=J,REGIØN=30K
//  EXEC WECT,DSN='PRDT5.CUT.LØG',RETPD=14,REC=85,BLK=850
//CT.SYSUT2  DD  UNIT=TAPE
//CT.SYSUT1  DD  *
  card input
/*
```

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  WECT
//WECT PROC REG=30,RETPD=1,FM=FB,REC=80,BLK=800                          00000010
//UNCATLG EXEC PGM=IEFBR14,REGION=&REG.K                                 00000020
//UNCATLG DD DSN=&DSN,DISP=(OLD,UNCATLG),UNIT=(TAPE,,DEFER),             00000030
//             VOL=SER=TAPE                                              00000040
//CT    EXEC PGM=WECT,REGION=&REG.K,COND=EVEN                            00000050
//SYSUT3 DD SYSOUT=A,DCB=LRECL=120                                       00000060
//SYSUT2 DD UNIT=TAPE,DSN=&DSN,DISP=(,CATLG),LABEL=RETPD=&RETPD,         00000070
//             DCB=(RECFM=&FM,LRECL=&REC,BLKSIZE=&BLK)                   00000080
```

6. REFERENCE

Division 2, Chapter 4, Section 2 contains a Warrenville Data Center Standard describing input data streams. It is recommended that the user become acquainted with these Standards prior to using the WECT cataloged procedure.

Reason for reissue: Reference to ASP SETUP card deleted.

WELIST - LISTING VTOC, PDS DIRECTORIES, CATALOGS

1. GENERAL

WELIST is a cataloged procedure which facilitates use of the IBM Utility Program IEHLIST for listing catalogs, partitioned data set directories, or volume table of contents of the system residence volume and/or a disk pack of the users designation. WELIST provides DD statements for SYSPRINT, the system residence pack, and/or a disk pack of the users designation.

2. SYMBOLIC PARAMETER

WELIST requires a VOLID parameter which supplies the volume serial number of the disk pack on which the functions are to be performed.

3. USING WELIST

The WELIST procedure is invoked with an EXEC statement which includes the symbolic parameter,VOLID, and is followed by IEHLIST control statements.

```
    //    EXEC WELIST,VØLID=xxxxxx
      control statements
```

4. EXAMPLE OF USE

```
//    EXEC WELIST,VØLID=TEST05
  LISTPDS DSNAME=DEV.LØADMØD,VØL=3330=TEST05
/*
```

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  WELIST
// PROC VOLID=SYSR01                                                       00000100
//LIST     EXEC PGM=IEHLIST,REGION=44K                                     00000200
//SYSPRINT  DD SYSOUT=A                                                    00000300
//DDRES DD VOLUME=REF=SYS1.SVCLIB,DISP=OLD                                 00000400
//DD1 DD UNIT=SYSDA,VOL=SER=&VOLID,DISP=OLD                                00000500
```

6. REFERENCE

IBM OS Utilities C28-6586,IEHLIST

WEMOD - CATALOG OPERATIONS, RENAMING, SCRATCHING

1. GENERAL

WEMOD is a cataloged procedure which facilitates use of the IBM Utility Program IEHPROGM for scratching, renaming, and performing catalog operations on direct access volumes. WEMOD provides DD statements for SYSPRINT, the system residence volume, and a pack of the users designation.

2. SYMBOLIC PARAMETER

WEMOD requires a VOLID parameter which supplies the volume serial number of the disk pack on which the functions are to be performed.

3. USING WEMOD

The WEMOD procedure is invoked with an EXEC statement which includes the symbolic parameter, VOLID, and is followed by IEHPRØGM control statements.

```
    //   EXEC WEMØD,VØLID=xxxxxx
      control statements
```

4. EXAMPLE OF USE

```
//   EXEC WEMØD,VØLID=TEST06
  SCRATCH DSNAME=USERS.DATASET,VØL=3330=TEST06
/*
```

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  WEMOD
// PROC VOLID=SYSR01                                                  00000100
//MOD      EXEC PGM=IEHPROGM,REGION=44K                               00000200
//SYSPRINT DD SYSOUT=A,SPACE=(TRK,(5,5))                  69160-EWR   00000300
//DDRES DD VOLUME=REF=SYS1.SVCLIB,DISP=OLD                            00000400
//DD1 DD UNIT=SYSDA,VOL=SER=&VOLID,DISP=OLD                           00000500
```

6. REFERENCE

IBM OS Utilities GC28-6586, IEHPROGM

DADUMP - DIRECT ACCESS FILE DUMP LISTING

1. GENERAL

DADUMP is a catalogued procedure which facilitates use of the Western Electric utility program WEDEBEPT to print a sequential direct access file whose characteristics may not be known. It employs symbolic parameters to supply the required parm fields as well as one for dsname. The data set under investigation is assumed to be catalogued or passed. If it is not, the volume serial number of the direct access volume must be supplied with an override card for ddname SYSUT1.

2. SYMBOLIC PARAMETERS

BEGBLK — 5 digit field to designate the block number at which printing is to begin. The default is 00001.

NOBLKS — 5 digit field to designate the number of blocks to be printed. The default is 00025. The keyword ALLPT may be used to print the entire file.

DSN — This parameter is required to designate the file's dsname.

REP — 1 byte code to designate character representation to be used in printing. The default is C for normal characters. H is used to designate hexadecimal representation.

3. USING DADUMP

3.1 The procedure is invoked with an EXEC card, the required symbolic parameter, DSN, and the optional symbolic parameters:

```
// EXEC DADUMP,DSN=dsname
```

3.2 The data set is assumed to be catalogued or passed. If it is not, the user must supply a volume serial number to indicate the data set's location via an override card for ddname SYSUT1.

4. EXAMPLES

4.1 Print 10 blocks of data of a passed data set named &&TEMPFILE.

```
// EXEC DADUMP,DSN='&&TEMPFILE',NØBLKS=00010
```

4.2 Print 50 blocks of data of a file located on disk pack PRØD07 in hexadecimal representation. The data set, PRD7.MASTER, is not catalogued.

```
// EXEC DADUMP,DSN='PRD7.MASTER',NØBLKS=00050,REP=H
//SYSUT1 DD VØL=SER=PRØD07
```

DADUMP - DIRECT ACCESS FILE DUMP LISTING (CONT.)

5. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  DADUMP
//DADUMP PROC BEGBLK=00001,NOBLKS=00025,REP=C,DSN=NONE
//*
//*   BEGBLK - 5 DIGIT FIELD, BLOCK TO BEGIN PRINTING
//*   NOBLKS - 5 DIGIT FIELD, NO. OF BLOCKS TO BE PRINTED
//*   REP - 1 BYTE CODE. C IS FOR NORMAL CHARACTER REPRESENTATION AND
//*   H IS FOR HEXADECIMAL.
//*   DSN - DSNAME OF DATA SET TO BE PRINTED
//*   IF DSN ISN'T CATALOGUED, OVERRIDE SYSUT1 WITH A VOL SER PARAMETER
//*
//DEBE EXEC PGM=WEDEBEPT,REGION=30K,
// PARM=(01,&BEGBLK,01,&NOBLKS,&REP)
//SYSUT1 DD DSN=&DSN,DISP=(SHR,PASS),UNIT=DISK
//SYSUT2 DD SYSOUT=A,DCB=(DSORG=PS)
//SYSUDUMP DD SYSOUT=A
```

6. REFERENCE

Division 3, Chapter 2, Section 2, WEDEBEPT.

TPDUMP - TAPE DUMP LISTING

1. GENERAL

TPDUMP is a catalogued procedure which facilitates use of the Western Electric utility program WEDEBEPT to print the files of a magnetic tape whose characteristics may not be known. It employs symbolic parameters to supply the required parm fields as well as one for the volume serial number.

2. SYMBOLIC PARAMETER

BEGFILE 2 digit field to designate begining file to be printed. The default is 01.

BEGBLK 5 digit field to designate the block at which printing is to begin. The default is 00001.

NOFILES 2 digit field to designate the number of files to be printed. The default is 01.

NOBLKS 5 digit field to designate the number of blocks to be printed. Default is 00025. The keyword ALLPT may be used to print the entire file.

REP 1 byte code to designate character representation. The default is C for normal characters. H is used to designate hexadecimal representation.

SER 6 digit volume serial number of tape being dumped. The SER parameter must be supplied by the user.

DSN The default is ANY. The label is not checked.

3. USING TPDUMP

3.1 The procedure is invoked with an EXEC card and the required SER parameter and the optional parameters:

```
// EXEC TPDUMP,SER=nnnnnn
```

3.2 Users must be aware that headers and trailers on standard label tapes constitue files. Standard label tapes are of the form: header-data-trailer header-data-trailer, etc. The appropriate file or files must be selected employing the symbolic parameters.

(Continued on next page)

Reason for reissue: As indicated and reference to ASP SETUP card deleted.

TPDUMP - TAPE DUMP LISTING (CONT.)

4. EXAMPLES

4.1 Print the headers and 25 blocks of data of standard label tape 712345.

```
// EXEC  TPDUMP,SER=712345,NØFILES=02
```

4.2 Print 100 blocks of data on standard label tape 700001 beginning with block 50.

```
// EXEC  TPDUMP,BEGFILE=02,BEGBLK=00050,NØBLKS=00100,SER=700001
```

5. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  TPDUMP
//TPDUMP PROC BEGFILE=01,BEGBLK=00001,NOFILES=01,NOBLKS=00025,REP=C,
// DSN=ANY
//*
//*   BEGFILE - 2 DIGIT FIELD, BEGINNING FILE TO BE PRINTED
//*   BEGBLK - 5 DIGIT FIELD, BLOCK TO BEGIN PRINTING
//*   NOFILES - 2 DIGIT FIELD, NO. OF FILES TO BE PRINTED
//*   NOBLKS - 5 DIGIT FIELD, NO. OF BLOCKS TO BE PRINTED
//*   REP - 1 BYTE CODE. C IS FOR NORMAL CHARACTER REPRESENTATION AND
//*   H IS FOR HEXADECIMAL.
//*   SER - 6 DIGIT SERIAL NUMBER FOR TAPE VOLUME IS REQUIRED.
//*
//DEBE EXEC PGM=WEDEBEPT,REGION=30K,
// PARM=(&BEGFILE,&BEGBLK,&NOFILES,&NOBLKS,&REP)
//SYSUT1 DD UNIT=TAPE,LABEL=(,BLP),DSN=&DSN,VOL=SER=&SER,DISP=OLD
//SYSUT2 DD SYSOUT=A,DCB=(DSORG=PS)
//SYSUDUMP DD SYSOUT=A
```

6. REFERENCE

Division 3, Chapter 2, Section 2, WEDEBEPT.

Reason for Reissue: As indicated and reference to ASP SETUP card deleted.

PPANAL - PROBLEM PROGRAM ANALYZER

1. GENERAL

PPANAL is a single step procedure which facilitates use of the Boole and Babbage analyzer program of the Problem Program Efficiency package. The procedure offers two symbolic parameters to designate the tape containing the extracted data to be analyzed. The programmer may add analyzer control statements to control the study.

2. SYMBOLIC PARAMETERS

DSN Supply the dsname of the extracted data. Enclose it in apostrophes if it contains any special characters (i.e. periods or ampersands.) The default name is PPDEXT.

SER Supply the volume serial number of the standard label tape.

3. USING PPANAL

3.1 The procedure is invoked with an EXEC card with the symbolic parameters designating volume serial number and dsname. The volume serial number is required unless the tape is passed from a previous step. If it is passed, the VOL parameter must be nullified with a overriding dd statement. Analyzer control statements may be added after the EXEC card or after an overriding DD statement if one is used.

4. EXAMPLES

4.1

```
//  EXEC PPANAL,DSN=EXT1,SER=123456
 analyzer control cards
/*
```

The above example will analyze the data on tape 123456 under the dsname of EXT1.

4.2

```
//  EXEC PPANAL
//FT04F001 DD VØL=
 analyzer control cards
/*
```

The above example uses the default dsname, PPDEXT. An override card is provided to nullify the volume parameter in order to accept a "passed" tape from an earlier step.

Reason for reissue: Reference to ASP SETUP card deleted.

PPANAL - PROBLEM PROGRAM ANALYZER (CONT.)

5. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME   PPANAL
// PROC   REG=90,DSN=PPDEXT,SER=000000                                        00000010
//PPANAL EXEC PGM=PPANAL,REGION=&REG.K                                        00000020
//FT04F001 DD DSN=&DSN,DISP=OLD,VOL=SER=&SER,UNIT=TAPE                        00000030
//FT06F001 DD SYSOUT=A,DCB=(DSORG=PS)                                         00000040
//FT05F001 DD DDNAME=SYSIN                                                    00000050
//SYSABEND DD  SYSOUT=A                                                       00000060
```

6. REFERENCE

6.1 SMS/360 Users Guide for PPE, Boole and Babbage
6.2 See catalogued procedure, PPEXT, this section.
6.3 SMS/360 (PPE), Division 3, Chapter 2.

PPEXT - PROBLEM PROGRAM EFFICIENCY EXTRACTOR

1. GENERAL

PPEXT is a single step procedure which facilitates use of the Boole and Babbage extractor program of the Problem Program Efficiency package. The procedure provides five symbolic parameters to allow designation of a load library, variation of the region, variation of the dsname of the extracted information, its disposition and retention period of the tape volume. The programmer provides his program's DD statements and PPE control cards.

2. SYMBOLIC PARAMETERS

LIB Supply the dsname of the load library which contains the program to be tested. The name must be enclosed in apostrophes if it contains any special characters. There is no default, the name must must be provided.

REG Supply a region value expressed as nnn, where nnn consists of the users program region plus 30K for the extractor. The default is 60.

DSN Supply a dsname for the extracted information for analysis. Enclose it in apostrophes if it contains periods. The default name is PPDEXT.

DISP This designates a disposition of the extracted data tape. It defaults to KEEP. Users may wish to use PASS if the information is to be processed by the analyzer program in the same job.

RET This designates a retention period in days for extracted data tape. The default is 0. If programmers wish to retain the tape for subsequent analysis runs, they should specify the number of days the tape is to be retained.

3. USING PPEXT

3.1 The procedure is invoked with an EXEC card with the optional and required symbolic parameters. PPE control cards including one with the name of the program under analysis (the name is entered starting in column 1) are placed after the EXEC card. DD statements required by the users program are placed after this card or cards.

```
//  EXEC PPEXT,REG=nnn,DSN=dsname,DISP=xxxx,RET=nn
 PPE control cards
//USERDD  DD  programs DD statements
```

3.2 Should one of the program's required ddnames be SYSIN, the following DD card must precede the PPE cards.

```
//PPE2EIN1  DD *
```

Reason for reissue: Reference to ASP SETUP card deleted.

PPEXT - PROBLEM PROGRAM EFFICIENCY EXTRACTOR (CONT.)

4. EXAMPLES OF USE

4.1
```
// EXEC PPEXT,LIB='DEV.LØADMØD',DISP=PASS,REG=90
PGM1
//DATAIN  DD
//REPØRT  DD
//ERRØRS  DD
```

The above example will analyze a run of PGM1 which resides in DEV.LØADMØD. A region of 90K is allowed for the run and data extraction. The extracted data is passed for subsequent processing under the default name PPDEXT and the tape will not be retained.

4.2
```
// EXEC PPEXT,LIB='&&LØADSET',RET=10,REG=120K,DSN=EXT1
PGMXYZ
//DD1  DD
//DD2  DD
```

The above example is prepared to analyze a program located in the temporary library &&LOADSET created and passed from a compile and link. The extracted data will be retained on a tape for 10 days under the name EXT1.

5. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  PPEXT
// PROC  REG=60,DISP=KEEP,RET=0,DSN=PPDEXT                                00000010
//PPE EXEC PGM=PPDEXT1,REGION=&REG.K                                      00000020
//PPE2EOU2 DD UNIT=TAPE,DSN=&DSN,DISP=(,&DISP),LABEL=RETPD=&RET           00000030
//PPE2EERR DD SYSOUT=A,DCB=(DSORG=PS)                                     00000040
//PPE2EIN1 DD DDNAME=SYSIN                                                00000050
//STEPLIB DD DSN=&LIB,DISP=SHR                                            00000060
```

6. REFERENCE

6.1 SMS/360 Users Guide for PPE, Boole and Babbage
6.2 See catalogued procedure, PPANAL, this section.
6.3 SMS/360 (PPE), Division 3, Chapter 2.

Reason for reissue: Reference to ASP SETUP card deleted.

SORT - SORT/MERGE WITH ROUTINES THAT REQUIRE LINK EDITING

1. GENERAL

The SORT cataloged procedure is designed to be used in sorting and merging applications that have modification routines that require link editing. You can use this procedure for all sort/merge applications, but it is inefficient for those that do not have modification routines that require link editing, because it causes unnecessary linkage editor data sets to be allocated.

2. USING SORT

2.1 SORT is invoked by an EXEC statement

```
// EXEC SØRT
```

2.2 The programmer must define the input data.

```
//SØRTIN DD parameters describing input
```

2.3 The file for sorted output must be defined.

```
//SØRTØUT DD parameters describing output.
```

2.4 Three to seventeen auxiliary devices must be defined as work files. They may be disk or tape.

```
//SØRTWKnn DD parameter describing work files.
```

2.5 A file must be defined for the sort control statements.

```
//SYSIN DD *
sort control cards
```

2.6 Additional data sets which contain modification routines may be defined under a user specified ddname.

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  SORT
//SORT     PROC                                                         00000010
//SORT     EXEC PGM=SORT                                                00000020
//SYSOUT DD SYSOUT=A                                                    00000030
//SYSPRINT DD  DUMMY                                                    00000040
//SYSLMOD  DD UNIT=SYSDA,SPACE=(TRK,(20,20,1))                          00000050
//SYSLIN  DD UNIT=SYSDA,SPACE=(TRK,(1,1))                               00000060
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR                                  00000070
//SYSUT1  DD   UNIT=(SYSDA,SEP=(SORTLIB,SYSLMOD,SYSLIN)),              X00000080
//             SPACE=(TRK,(10,4))                                       00000090
```

4. REFERENCE

IBM OS Sort/Merge Programmers' Guide, SC33-4007

SORTD - SORT/MERGE WITH NO ROUTINES OR ROUTINES THAT DO NOT REQUIRE LINK EDITING

1. GENERAL

The SORTD cataloged procedure is designed for sorting and merging applications that have no modification routines, or have modification routines that do not require link editing. It cannot be used for applications having modification routines that need link editing.

2. USING SORT

2.1 SORT is invoked by an EXEC statement.

```
// EXEC SØRTD
```

2.2 The programmer must define the input data.

```
//SØRTIN DD parameters describing input
```

2.3 The file for sorted output must be defined.

```
//SØRTØUT DD parameters describing output
```

2.4 Three to seventeen auxiliary devices must be defined as work files. They may be disk or tape.

```
//SØRTWKnn DD parameter describing work files
```

2.5 A file must be defined for the sort control statements.

```
//SYSIN DD *
 sort control cards
```

3. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  SORTD
//SORTD   PROC                                                  00000010
//SORT    EXEC PGM=SORT                                         00000020
//SYSOUT DD SYSOUT=A                                            00000030
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR                          00000040
```

4. REFERENCE

IBM OS Sort/Merge Programmers' Guide, SC33-4007

COB4COL - COBOL COMPILE, OPTIMIZE AND LINK-EDIT

1. GENERAL

COB4COL is a two-step catalog procedure to compile, optimize and link edit a source module into an optimized load module. The optimization is done with the Capex II Optimizer. (See Div. 3, Chap. 2, Sect. 3, App. F.)

2. USING COB4COL

2.1 COB4COL may be invoked by an EXEC statement:

```
// EXEC CØB4CØL
```

2.2 The following DD statements must be added to the procedure.

2.21 A COB.SYSIN DD statement defining the input.

```
//CØB.SYSIN DD *
  (COBOL source deck)
```

or

```
//CØB.SYSIN DD  (parameters defining a source input
                 data set)
```

2.22 A LKED.SYSLMØD DD statement to define a storage place for the load module.

```
//LKED.SYSLMØD DD  (parameters describing the load
                    module PDS)
```

2.3 The following DD statement may be added to supply additional input to the linkage editor.

```
//LKED.SYSIN DD  (appropriate parameters)
```

This statement must follow all //CØB. DD statements.

3. EXAMPLE OF USE

```
// EXEC  CØB4CØL
//CØB.SYSIN  DD  *
    COBOL source deck
//LKED.SYSLMØD  DD  DSN=DEV.LØADLIB(TESTMØD),DISP=SHR
```

COB4COL - COBOL COMPILE, OPTIMIZE AND LINK-EDIT (Cont.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  COB4CUL
//COB4COL PROC LINES=55                                                 00000010
//COB       EXEC PGM=CPXUPTSM,REGION=128K,                              00000020
//                PARM='NODECK,NOMAP,LINECNT=&LINES'                    00000030
//STEPLIB DD DSN=SYS2.ANSLIB,DISP=SHR                                   00000040
//         DD DSN=SYS2.COBOL4.LINK,DISP=SHR                             00000050
//SYSIN1 DD     DSNAME=&&SYSIN1,UNIT=SYSDA,SPACE=(605,(4640,580))       00000060
//SYSIN2 DD     DSNAME=&&SYSIN2,UNIT=SYSDA,SPACE=(400,(240,80))         00000070
//STATSDD DD DSN=OPTSTATU,DISP=SHR                                      00000080
//SYSUDUMP DD SYSOUT=A                                                  00000090
//SYSPRINT DD SYSOUT=A                                                  00000100
//SYSPUNCH DD SYSOUT=P,DCB=(RECFM=F,BLKSIZE=80)                         00000110
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(460,(6000,1500))              00000120
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSDA,SPACE=(460,(700,100))                00000130
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(460,(700,100))                00000140
//SYSUT4 DD DSN=&SYSUT4,UNIT=SYSDA,SPACE=(460,(700,100))                00000150
//SYSLIN DD DSN=&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),                    00000160
// SPACE=(400,(240,80)),DCB=BLKSIZE=400                                 00000170
//LKED EXEC PGM=ATTIEWL,REGION=128K,COND=(5,LT,COB),                    00000180
//        PARM='XREF,LIST,LET'                                          00000190
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                              00000200
//        DD DDNAME=SYSIN                                               00000210
//SYSLMOD DD DSN=&GOSET(LOADMOD),DISP=(,PASS),UNIT=SYSDA,               00000220
// SPACE=(TRK,(9,,1))                                                   00000230
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR                                00000240
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(14,5))                               00000250
//SYSPRINT DD SYSOUT=A                                                  00000260
```

5. REFERENCE

IBM ANS COBOL version 4 Programmers' Guide, SC28-6456

META-METACOBOL PRE-COMPILER PROCESSOR

1. GENERAL

META is a one step procedure to execute the Applied Data Research (ADR) METACOBOL pre-compiler processor. COBOL card image source statements preceeded by METACOBOL statements (if any) is the input used by METACOBOL and a COBOL card image output is produced. METACOBOL can access a LIBRARIAN type source file. Before executing this procedure, the user must access the METACOBOL manuals listed below for selecting the desired features via PARM input and/or control statements.

2. USING META

2.1 The META procedure is invoked with an EXEC statement which may or may not include "PARM.META" parameters as defined in the METACOBOL user manual. METACOBOL statements are also defined in the same manual. The region parameter varies depending upon the options chosen. At least a REGION=120K is required.

```
//     EXEC META ,PARM.META='xx,xx,xx=x,xx'
//META.CARDF DD *
      METACOBOL STATEMENTS (if any)
      SOURCE DECK (ANS COBOL.COBOL F source is acceptable only if
                   converting to ANS COBOL)
```

3. EXAMPLES OF USE

```
//STEP1    EXEC META,PARM.META='ID=PLARPT,LI,SY,TA,RE=010'
//META.CARDF DD *
      *MACRO DIVISION.
      *$COPY EFA
      COBOL F SOURCE STATEMENTS
```

This example is a COBOL F conversion to ANS COBOL. The new source statement output is on the file DDN=PUNCHF and may be used as input to an ANS COBOL compile. This example required a REGION=240K parameter.

4. CATALOGED PROCEDURE LISTING

```
//META EXEC PGM=MCT                                                     00000010
//LSTIN DD SYSOUT=A,DCB=(LRECL=121,BLKSIZE=968,RECFM=FB)                00000020
//PUNCHF DD SYSOUT=P,DCB=BLKSIZE=80                                     00000030
//FE DD UNIT=SYSDA,SPACE=(TRK,(20,10))                                  00000040
//FM DD UNIT=SYSDA,SPACE=(TRK,(20,10))                                  00000050
//FZ DD UNIT=SYSDA,SPACE=(TRK,99,,CONTIG)                               00000060
//SYSLIB DD DSN=SYS2.METACOB.SLIB,DISP=SHR                              00000070
//        DD DSN=SYS2.METALIB,DISP=SHR                                  00000080
```

META-METACOBOL PRE-COMPILE PROCESSOR (Contd.)

5. REQUIRED REFERENCES

ADR METACOBOL User Manual
ADR METACOBOL MACADR Manual
ADR METACOBOL Macro Facilities Reference Manual
ADR Macro Writing, Volume 1 and 2.

DASPACE - CALCULATE DIRECT ACCESS SPACE USAGE

1. GENERAL

DASPACE is a catalogued procedure which facilitates use of the Western Electric utility program, DASPACE, for calculating direct access space usage for a logical record of given length under varied blocking factors.

2. CONTROL CARD PARAMETERS

DASPACE requires a control card for each logical record length under investigation:
(beginning in column 1) LLLLL,SSSSSSSS,KKK

where L is the 5 digit logical record length
S is the 8 digit estimated record count
K is the 3 digit key length

3. USING DASPACE

The procedure is invoked with an EXEC statement and appropriate control cards.

```
  //  EXEC  DASPACE
      control cards (one or more cards)
```

4. CATALOGUED PROCEDURE LISTING

```
MEMBER NAME  DASPACE
//DASPACE PROC                                                                00000010
//DASPACE EXEC PGM=DASPACE,REGION=30K                                         00000020
//* CONTROL CARD INPUT IS OF THE FORM:                                        00000030
//*   LLLLL,SSSSSSSS,LLL     STARTING IN COLUMN 1                             00000040
//*   LLLLL IS A 5 DIGIT LRECL                                                00000050
//*   SSSSSSSS IS AN 8 DIGIT RECORD COUNT                                     00000060
//*   KKK IS A 3 DIGIT KEY LENGTH WHEN KEYS ARE USED                          00000070
//*                                                                           00000080
//*   A PARM=L MAY BE USED FOR A FULL RANGE LISTING                           00000090
//*   IF NO PARM IS SUPPLIED A CONDENSED LISTING IS PRODUCED                  00000100
//USAGE DD SYSOUT=A,DCB=(RECFM=FB,LRECL=121,BLKSIZE=1936)                     00000110
//SYSOUT DD SYSOUT=A                                                          00000120
//PARMS DD DDNAME=SYSIN                                                       00000130
```

5. REFERENCE

Division 3, Chapter 2, Section 2, DASPACE Utility Program

REACT - DATA RETRIEVAL/REPORT FORMATTING SYSTEM

1. GENERAL

1.1 REACT is a catalogued procedure which executes the five programs of the RE-ACT Data Retrieval and Report Formatting System. Data is retrieved as requested via RE-ACT statements from a designated sequential or index sequential file residing on tape or disk. Output consists of printed reports, formatted as requested via RE-ACT statements. Totaling, sub-totaling, computations, page headings, and report sequencing can also be requested.

2. USING REACT

2.1 The REACT procedure is invoked with the following JCL statements:

```
//*FØRMAT PR,DDNAME=PRINTER1,CØPIES=1
//     EXEC  REACT,REGIØN=60K
//REACT1.CARDSIN  DD  *
  (RE-ACT Statements. A maximum of 45 requests)
//REACT4.INFILE  DD  DSN=your input file
```

2.11 See "RE-ACT System" manual for description of RE-ACT statements.

(Continued on next page)

3. CATALOGUED PROCEDURE

```
MEMBER NAME  REACT
//REACTRTV PROC SWK=SYSDA,WKSPACE=8,PRSPACE=8,OPT=                      00000100
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 00000110
//*                                                                   * 00000120
//*       PROCEDURE TO EXECUTE 'RETRIEVAL' PORTION OF 'RE-ACT'        * 00000130
//* WKSPACE = SORT WORK SPACE                                          * 00000140
//* PRSPACE = SPACE FOR PRODUCT WORK FILE, SHOULD BE 5 TIMES WKSPACE  * 00000150
//* WKSPACE AND PRSPACE SHOULD ONLY BE CODED FOR VERY LARGE REQUESTS  * 00000160
//* OPT IS THE PARM PARAMETERS FOR REACT4 -- SEE USERS MANUAL         * 00000170
//*                                                                   * 00000180
//* REACT1.CARDSIN DD CARD REQUIRED FOR DATA NAME TABLE AND REQUESTS  * 00000190
//* REACT2.PRINTER DD CARD OPTIONAL FOR LISTING OF DATA NAMES         * 00000200
//* REACT4.INFILE  DD CARD REQUIRED TO DEFINE YOUR MASTER FILE        * 00000210
//* REACT6.TAPEXX  DD CARD REQUIRED ONLY FOR TAPE OR DISK PRODUCTS    * 00000220
//*                   WHERE XX = RE-ACT REQUEST NUMBER                * 00000230
//* REACT6.DATECARD DD CARD OPTIONAL TO OVER-RIDE IPL DATE            * 00000240
//*                                                                   * 00000250
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 00000260
//REACT1  EXEC PGM=REACT1                                               00000270
//STEPLIB  DD DSN=SYS3.REACT,DISP=SHR                                   00000280
//DISK      DD  DSNAME=&&REACT1,DISP=(,PASS),UNIT=SYSDA,                00000290
//              SPACE=(6400,(20,20),RLSE),DCB=BLKSIZE=6400              00000300
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 00000310
//*                                                                   * 00000320
//* THE FOLLOWING DD CARD DEFINES YOUR DATA NAME TABLE AND REQUESTS   * 00000330
//*                                                                   * 00000340
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 00000350
//CARDSIN  DD  DUMMY,DCB=BLKSIZE=6400                                   00000360
//SORT1  EXEC PGM=SORT,PARM='CORE=60000'                                00000370
//SYSOUT DD SYSOUT=A                                                    00000380
//SYSUDUMP DD SYSOUT=A                                                  00000390
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR                                  00000400
//SORTMSG  DD SYSOUT=A,SPACF=(1694,(7))                                 00000410
//SORTIN  DD DSN=&&REACT1,DISP=(OLD,DELETE)                             00000420
//SORTWK01 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000430
//SORTWK02 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000440
//SORTWK03 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000450
//SORTWK04 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000460
//SORTOUT   DD  DSNAME=&&REACT2,DISP=(,PASS),UNIT=SYSDA,                00000470
//              SPACE=(6400,(20,20)),                                   00000480
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)                    00000490
//SYSIN     DD DISP=SHR,DSN=SYS1.CONTROL(RCTSORT3)                      00000500
//REACT2  EXEC PGM=REACT2                                               00000510
//STEPLIB  DD DSN=SYS3.REACT,DISP=SHR                                   00000520
//SYSUDUMP DD SYSOUT=A                                                  00000530
//DISKIN    DD  DSNAME=*.SORT1.SORTOUT,DISP=(OLD,DELETE),               00000540
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)                    00000550
//DISKOUT   DD  DSNAME=&&REACT3,DISP=(,PASS),UNIT=SYSDA,SEP=DISKIN,     00000560
//              SPACE=(6480,(10,10),RLSE),                              00000570
//              DCB=(RECFM=FB,LRECL=81,BLKSIZE=6480)                    00000580
//PRINTER  DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=665)            00000590
//REACT3  EXEC PGM=REACT3                                               00000600
//STEPLIB  DD DSN=SYS3.REACT,DISP=SHR                                   00000610
//SYSUDUMP DD SYSOUT=A                                                  00000620
//INFILE    DD  DSNAME=*.REACT2.DISKOUT,DISP=(OLD,DELETE),              00000630
//              DCB=(RECFM=FB,LRECL=81,BLKSIZE=6480)                    00000640
//OUTFIL1   DD  DSNAME=&&REACT5,DISP=(,PASS),UNIT=SYSDA,                00000650
//              SPACE=(CYL,1),DCB=(RECFM=FB,LRECL=15,BLKSIZE=6000)      00000660
//OUTFIL2   DD  DSNAME=&&REACT4,DISP=(,PASS),UNIT=SYSDA,SEP=INFILE,     00000670
//              SPACE=(CYL,(&PRSPACE,2)),                               00000680
//              DCB=(RECFM=FB,LRECL=256,BLKSIZE=6144)                   00000690
//PRINTER   DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=665)          00000700
```

(Continued)

3. CATALOGUED PROCEDURE (CONT.)

```
//REACT4   EXEC PGM=REACT4,PARM='&OPT'                                  00000710
//STEPLIB  DD DSN=SYS3.REACT,DISP=SHR                                   00000720
//SYSUDUMP DD SYSOUT=A                                                  00000730
//TABLES    DD  DSNAME=*.REACT3.OUTFIL1,DISP=(OLD,DELETE),              00000740
//              DCB=(RECFM=FB,LRECL=15,BLKSIZE=6000)                    00000750
//OUTFILE   DD  DSNAME=*.REACT3.OUTFIL2,DISP=(MOD,PASS),                00000760
//              DCB=(RECFM=FB,LRECL=256,BLKSIZE=6144)                   00000770
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   00000780
//*                                                                 *   00000790
//* THE FOLLOWING DD STATEMENT DEFINES YOUR INPUT MASTER FILE       *   00000800
//*                                                                 *   00000810
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   00000820
//INFILE    DD  DUMMY                                                   00000830
//PRINTER   DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=665)          00000840
//SORT2   EXEC PGM=SORT,PARM='CORE=60000'                               00000850
//SYSOUT DD SYSOUT=A                                                    00000860
//SYSUDUMP DD SYSOUT=A                                                  00000870
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR                                  00000880
//SORTMSG   DD SYSOUT=A,SPACE=(1694,(7))                                00000890
//SORTWK01 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000900
//SORTWK02 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000910
//SORTWK03 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000920
//SORTWK04 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000930
//SORTWK05 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000940
//SORTWK06 DD UNIT=&SWK,SPACE=(CYL,(&WKSPACE),,CONTIG)                  00000950
//SORTIN    DD  DSNAME=&&REACT4,DISP=(OLD,DELETE),                      00000960
//              DCB=(RECFM=FB,LRECL=256,BLKSIZE=6144)                   00000970
//SORTOUT   DD  DSNAME=&&REACT6,DISP=(,PASS),UNIT=SYSDA,                00000980
//              SPACE=(CYL,(&PRSPACE,2)),                               00000990
//              DCB=(RECFM=FB,LRECL=256,BLKSIZE=6144)                   00001000
//SYSIN     DD DISP=SHR,DSN=SYS1.CONTROL(RCTSORT4)                      00001010
//REACT6   EXEC  PGM=REACT6                                             00001020
//STEPLIB   DD DSN=SYS3.REACT,DISP=SHR                                  00001030
//SYSUDUMP DD SYSOUT=A                                                  00001040
//INFILE    DD  DSNAME=*.SORT2.SORTOUT,DISP=(OLD,DELETE),               00001050
//              DCB=(RECFM=FB,LRECL=256,BLKSIZE=6144)                   00001060
//PRINTER   DD SYSOUT=A,DCB=(RECFM=FB,LRECL=132,BLKSIZE=931)            00001070
//PRINTER1 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)             00001080
//PRINTER2 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)             00001090
//PRINTER4 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)             00001100
//PRINTER6 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)             00001110
//TAPEXX    DD  DUMMY          TAPE PRODUCT, XX=REQUEST IDENTIFICATION  00001120
//DATECARD DD   DUMMY,DCB=BLKSIZE=80                                    00001140
//PUNCH     DD SYSOUT=P,DCB=(RECFM=FBA,LRECL=81,BLKSIZE=81)             00001150
```

4. REFERENCE

4.1 "RE-ACT System" manual.

TLSBINL - LIST THE CONTENTS OF THE TLS BIN TABLE

1. GENERAL

TLSBINL is a cataloged procedure which produces a formatted listing of the contents of the Tape Library System bin table by organization number. The program executed is TLSBINL which is documented in Division 2, Chapter 6, Section 6.

2. SYMBOLIC PARAMETER

An optional parameter, which supplies the PARM field to the TLSBINL program, may be specified to limit the bin table listing to one location or to one organization within a location.

The format of this parameter is CHECK=LLDDDD where LL is an alphabetic location code and DDDD is a department number (numeric). The default is a blank which will provide a complete listing of the bin table.

3. USING TLSBINL

TLSBINL is invoked with an EXEC statement.

```
// EXEC  TLSBINL
```

4. EXAMPLES

4.1 In this example a complete listing of the bin table is requested.

```
// EXEC  TLSBINL
```

4.2 In this example a listing of the bin table for location CB is requested.

```
// EXEC  TLSBINL,CHECK=CB
```

4.3 In this example a listing of the bin table for organization WN9413 is requested.

```
// EXEC  TLSBINL,CHECK=WN9413
```

5. CATALOGED PROCEDURE LISTING

```
//TLSBINL PROC CHECK=                                    00000100
//ST1  EXEC PGM=IEHLIST,REGION=28K                       00000200
//SYSPRINT DD DSN=&&BIN,UNIT=DISK,DISP=(,PASS),          00000300
//             SPACE=(TRK,(2,1)),DCB=BLKSIZE=968         00000400
//DD1      DD UNIT=DISK,VOL=SER=SYSR01,DISP=OLD          00000500
//SYSIN DD DISP=SHR,DSN=PRD4.CONTROL(TLSBINL)            00000600
//ST2  EXEC PGM=TLSBINL,REGION=14K,PARM=&CHECK           00000700
//STEPLIB DD DSN=PRD4.LOADLIB,DISP=SHR                   00000800
//IN   DD DSN=&&BIN,DISP=(OLD,DELETE)                    00000900
//LIST DD SYSOUT=A,DCB=BLKSIZE=968                       00001000
```

TLSBINL - LIST THE CONTENTS OF THE TLS BIN TABLE

1. GENERAL

TLSBINL is a cataloged procedure which produces a formatted listing of the contents of the Tape Library System bin table by organization number. The program executed is TLSBINL which is documented in Division 2, Chapter 6, Section 6.

2. SYMBOLIC PARAMETER

An optional parameter, which supplies the PARM field to the TLSBINL program, may be specified to limit the bin table listing to one location or to one organization within a location.

The format of this parameter is CHECK=LLDDDD where LL is an alphabetic location code and DDDD is a department number (numeric). The default is a blank which will provide a complete listing of the bin table.

3. USING TLSBINL

TLSBINL is invoked with an EXEC statement.

```
//  EXEC  TLSBINL
```

4. EXAMPLES

4.1 In this example a complete listing of the bin table is requested.

```
//  EXEC  TLSBINL
```

4.2 In this example a listing of the bin table for location CB is requested.

```
//  EXEC  TLSBINL,CHECK=CB
```

4.3 In this example a listing of the bin table for organization WN9413 is requested.

```
//  EXEC  TLSBINL,CHECK=WN9413
```

5. CATALOGED PROCEDURE LISTING

```
//TLSBINL PROC CHECK=                                                     00000100
//ST1  EXEC PGM=IEHLIST,REGION=28K                                        00000200
//SYSPRINT DD DSN=&&BIN,UNIT=DISK,DISP=(,PASS),                           00000300
//              SPACE=(TRK,(2,1)),DCB=BLKSIZE=968                         00000400
//DD1       DD UNIT=DISK,VOL=SER=SYSR01,DISP=OLD                          00000500
//SYSIN DD DISP=SHR,DSN=PRD4.CONTROL(TLSBINL)                             00000600
//ST2  EXEC PGM=TLSBINL,REGION=14K,PARM=&CHECK                            00000700
//STEPLIB DD DSN=PRD4.LOADLIB,DISP=SHR                                    00000800
//IN  DD DSN=&&BIN,DISP=(OLD,DELETE)                                      00000900
//LIST DD SYSOUT=A,DCB=BLKSIZE=968                                        00001000
```

DPMPSIL - LIST THE CONTENTS OF THE PSI TABLE

1. GENERAL

DPMPSIL is a cataloged procedure which produces a formatted listing of the contents of the PSI table by organization number. The program executed is DPMPSIL which is documented in Division 3, Chapter 2, Section 2.

2. SYMBOLIC PARAMETER

An optional parameter, which supplies the PARM field to the DPMPSIL program, may be specified to limit the PSI table listing to one location or to one organization within a location.

The format of this parameter is CHECK=LLDDDD where LL is an alphabetic location code and DDDD is a department number (numeric). The default is a blank which will provide a complete listing of the bin table.

3. USING DPMPSIL

DPMPSIL is invoked with an EXEC statement

```
// EXEC DPMPSIL
```

4. EXAMPLES

4.1 In this example a complete listing of the PSI table is requested.

```
//EXEC DPMPSIL
```

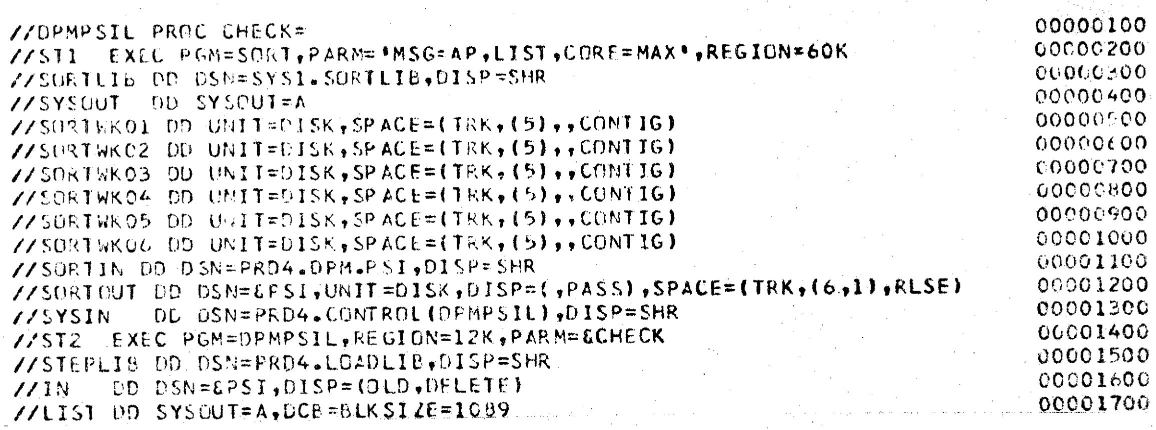
4.2 In this example a listing of the PSI table for location CB is requested.

```
// EXEC DPMPSIL,CHECK=CB
```

4.3 In this example a listing of the PSI table for organization WN9413 is requested.

```
// EXEC DPMPSIL,CHECK=WN9413
```

5. CATALOGED PROCEDURE LISTING



```
//DPMPSIL PROC CHECK=
//ST1   EXEC PGM=SORT,PARM='MSG=AP,LIST,CORE=MAX',REGION=60K
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSOUT   DD SYSOUT=A
//SORTWK01 DD UNIT=DISK,SPACE=(TRK,(5),,CONTIG)
//SORTWK02 DD UNIT=DISK,SPACE=(TRK,(5),,CONTIG)
//SORTWK03 DD UNIT=DISK,SPACE=(TRK,(5),,CONTIG)
//SORTWK04 DD UNIT=DISK,SPACE=(TRK,(5),,CONTIG)
//SORTWK05 DD UNIT=DISK,SPACE=(TRK,(5),,CONTIG)
//SORTWK06 DD UNIT=DISK,SPACE=(TRK,(5),,CONTIG)
//SORTIN DD DSN=PRD4.DPM.PSI,DISP=SHR
//SORTOUT DD DSN=&PSI,UNIT=DISK,DISP=(,PASS),SPACE=(TRK,(6,1),RLSE)
//SYSIN    DD DSN=PRD4.CONTROL(DPMPSIL),DISP=SHR
//ST2  EXEC PGM=DPMPSIL,REGION=12K,PARM=&CHECK
//STEPLIB DD DSN=PRD4.LOADLIB,DISP=SHR
//IN    DD DSN=&PSI,DISP=(OLD,DELETE)
//LIST DD SYSOUT=A,DCB=BLKSIZE=1089
```

WECDSK - CARD INPUT TO DISK

1. GENERAL

WECDSK is a cataloged procedure which facilitates use of the Western Electric utility program, WECT, for the transfer of card input from the job input stream to a disk data set and to catalog the data set under a user designated DSNAME.

2. WECT OPTIONS

2.1 WECT permits the user to specify one of three options, which are designed to meet the needs of systems interfacing with converted 7080 systems as well as the current generation systems. The options are governed by the symbolic parameter "REC" (Paragraph 3), which supplies a numeric value for the LRECL DCB parameter. The options are:

Option A - This is generally used in conjunction with converted 7080 systems and is equivalent to CTP04, a 1401 card-to-tape program. Four blanks and a record mark character are generated at the end of each logical record. In addition, an incomplete block is filled with nines padding before the tape mark is written. Designating REC=85 selects this option. If this option is selected, the user must specify a numeric value for the BLK symbolic parameter (Paragraph 3).

Option B - This option, selected by REC=81, generates a record mark character as the 81st byte of the output record.

Option C - The 80 column card image is copied as an 80 byte logical record, with blocking as specified in the BLK symbolic parameter. Designating REC=80 selects this option.

2.2 A message to SYSOUT, ddname SYSUT3, indicates the option selected and the card count.

3. SYMBOLIC PARAMETERS

There are ten possible symbolic parameters which the user may employ to control the characteristics and location of his output file.

REG	a numeric value for REGION. The default is 20.
UNIT	output device. The default is disk.
DSN	Data set name. The user must supply this value.

WECDSK - CARD INPUT TO DISK (CONT.)

3. SYMBOLIC PARAMETERS (CONT.)

VOL There are 3 options: if VOL is omitted the data set will be allocated to a "STOR" volume; if VOL='REF=PRDn.DATA' is used, a volume will be assigned via the catalog; if VOL='SER=volser' is used, the specified volume will be used.

REC a value for the LRECL parameter: use 80, 85 or 81. The default is 80.

BLK a value for the BLKSIZE parameter. The default is 3120. This value must be a multiple of the REC parameter.

FM The value for the RECFM parameter. The default is FB.

AMT The number of primary blocks to be allocated. The default is 100. (Enough for 3900 records.)

AMT2 The number of secondary blocks to be requested. The default is 25.

DISP Default disposition is '(,CATLG)'. If you wish to record on an existing data set, specify '(OLD,KEEP)'.

4. CAUTION

Users must determine whether the old data set needs to be scratched and/or uncataloged before executing this procedure.

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  WECDSK
//WECDSK PROC REG=20,UNIT=DISK,VOL=,FM=FB,REC=80,BLK=3120,
//  AMT=100,AMT2=25,DSN='PRD1.ERROR',DISP='(,CATLG)'
//CDSK EXEC PGM=WECT,REGION=&REG.K
//SYSUT3 DD SYSOUT=A,DCB=LRECL=120
//SYSUT2 DD UNIT=&UNIT,DSN=&DSN,DISP=&DISP,VOL=&VOL,
//  DCB=(RECFM=&FM,LRECL=&REC,BLKSIZE=&BLK),
//  SPACE=(&BLK,(&AMT,&AMT2),RLSE)
//SYSUT1 DD DDNAME=SYSIN
```

WECDSK - CARD INPUT TO DISK (CONT.)

6. EXAMPLES OF USE

6.1 Example 1: Placing a data set on a "STOR" volume. The data set must have been scratched and/or uncataloged prior to executing this procedure.

```
// EXEC WECDSK,DSN='PRD4.ABC.CARDSIN'
data cards here
```

6.2 Example 2: Placing a data set on a specified volume, 200 blocks of space requested, logical record length of 85, and a blocksize of 3145 specified.

```
// EXEC WECDSK,DSN='PRD4.ABC.CARDSIN(+1)',VØL='SER=PRØD11',
// AMT=200,REC=85,BLK=3145
data cards here
```

6.3 Example 3: Re-using an existing cataloged data set.

```
// EXEC WECDSK,DSN='PRD4.ABC.CARDSIN',DISP='(ØLD,KEEP)'
data cards here
```

6.4 Example 4: Using the volume reference technique. If a copy of the data set exists, it must be uncataloged and scratched prior to executing this procedure.

```
// EXEC WECDSK,DSN='PRD4.ABC.CARDSIN',VØL='REF=PRD4.DATA'
data cards here
```

WETPCOPY - MULTIPLE FILE TAPE COPY

1. GENERAL

WETPCOPY is a cataloged procedure that facilitates the use of a tape copy program which copies an entire tape to the double end of file mark. Multiple file tapes may be copied with a single EXEC card. No rewinding occurs between files. WETPCOPY uses the standard OS open for the first file indicated. All other files have their HDR1 and EOF1 type records copied from the input tape with the exception of the serial number field: this field contains the output tape number.

NOTE: Any data type records beginning with HDR1 or EOF1 would also have the output tape serial number moved to the serial number field.

2. SYMBOLIC PARAMETERS

2.1 WETPCOPY requires two symbolic parameters which are:

SER1	Serial number of the input tape.
DSN1	DSNAME of the first file to be copied.

2.2 WETPCOPY may have optional parameters which are:

DISP1 Disposition for the input tape. The default is (OLD,KEEP).

NO1 Input file number from which the tape copy is to begin. The default is 1.

DISP2 Disposition for the output tape. The default is (,KEEP).

NO2 Output file number. The default is 1.

RTPD Retention period to be assigned to the output tape. The default is 30.

DSN2 DSNAME of the first file copied. The default is *.SYSUT1, i.e. the same DSNAME as the input data set.

SER2 Serial number of the output tape. The default is SCRTCH.

WETPCOPY - MULTIPLE FILE TAPE COPY (CONT.)

3. USING WETPCOPY

The procedure is invoked with an EXEC card and the required DSN1 and SER1 parameters and any optional parameters:

```
//  EXEC WETPCOPY,DSN1='xxxxx.xxx.xxxxx',SER1=nnnnnn
```

Restrictions: see note in paragraph 1.

4. OUTPUT MESSAGES

WETPCOPY results in one of two types of messages appearing in the SYSMSG data set immediately following the LIB message.

```
***** SUCCESSFUL TAPE COPY ***** nn FILES COPIED
```

or

```
X**ERROR IN COPYING TAPE*****
```

where x indicates a POSITION, R, or W meaning a positioning error, a read error, or a write error respectively.

The condition code is set to zero for a successful completion and to 16 for any error.

5. CATALOGED PROCEDURE LISTING

```
//WETPCOPY  PROC DISP1='(OLD,KEEP)',NO1=1,DISP2='(,KEEP)',NO2=1,         00000010
//        RTPD=30,DSN2='*.SYSUT1',SER2=SCRTCH                            00000020
//*  COPY AN ENTIRE TAPE TO THE DOUBLE END-OF-FILE MARK                   00000030
//*  J.R.MALLECK DEPT 7784 INDIAN HILL                                    00000040
//TCOPY  EXEC PGM=TCOPY                                                   00000050
//STEPLIB DD DISP=SHR,DSN=PROD04.UTILPGMS                                 00000060
//SYSUT1 DD UNIT=TAPE,DISP=&DISP1,LABEL=(&NO1,SL),DSN=&DSN1,              00000070
//        VOL=SER=&SER1                                                   00000080
//SYSUT2 DD UNIT=TAPE,DISP=&DISP2,LABEL=(&NO2,SL,RETPD=&RTPD),            00000090
//        DSN=&DSN2,VOL=SER=&SER2                                         00000100
```

6. REFERENCE

J. R. Malleck, Dept. 7784, Indian Hill

TRMDUMMY - CREATE A "DUMMY" DISK GENERATION DATA SET

1. GENERAL

1.1 TRMDUMMY is a cataloged procedure developed for application systems interfacing with Mini-WEDGE. The function of this cataloged procedure is to create a "dummy" transmission data set on disk and make an entry in the system catalog for the "dummy" data set to avoid an empty GDG condition (no cataloged generations) which would cause job failure.

2. SYMBOLIC PARAMETERS

2.1 TRMDUMMY requires one symbolic parameter and may have three optional parameters, which are:

GDS, a required parameter, specifies the generation data group index (or base).

REG, an optional parameter, specifies the maximum region. The default is 10K. Ordinarily this default will not need to be changed.

VOLSER, an optional parameter, specifies the volume serial of the output disk pack. The default is the UNSTACK disk pack TRM002.

BLKSIZE, an optional parameter, specifies the DCB BLKSIZE value. The default is 1000.

3. ASP CONTROL CARD

3.1 All jobs using TRMDUMMY for systems interfacing with Mini-WEDGE must include an ASP //*MAIN statement which includes the CLASS=UNSTACK and FAILURE=CANCEL parameters.

4. USING TRMDUMMY

4.1 TRMDUMMY is invoked by an EXEC statement

```
// EXEC TRMDUMMY,GDS='GDSindex'
```

(Contined on next page)

TRMDUMMY (CONTD.)

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMDUMMY
//TRMDUMMY PROC GDS='GDS.DSNAME.BASE',                                  X00000100
//              VOLSER=TRM002,BLKSIZE=1000,                              X00000200
//              REG=10K                                                  00000300
//*             GENERATE A DUMMY GDS(+1) IF A GDS(0) DOES NOT ALREADY    00000400
//*             EXIST                                                    00000500
//*                                                                      00000600
//*             SYMBOLIC PARAMETER &GDS MUST SPECIFY THE GDG DATA SET    00000700
//*             NAME BASE                                                00000800
//STEP1   EXEC  PGM=TRMUST06,PARM='&GDS',REGION=&REG                     00000900
//STEPLIB  DD   DSNAME=PRD2.LOADLIB,DISP=SHR                             00001000
//STEP2   EXEC  PGM=WECOPY,COND=(0,EQ,STEP1),REGION=&REG                 00001100
//SYSPRINT DD   SYSOUT=A                                                 00001200
//SYSUT1   DD   DUMMY,DCB=(RECFM=U,BLKSIZE=1000,LRECL=1000)              00001300
//SYSUT2   DD   DSNAME=&GDS.(+1),DISP=(NEW,CATLG,DELETE),SPACE=(TRK,1), X00001400
//              VOLUME=SER=&VOLSER,UNIT=DISK,                            X00001500
//              DCB=(SYS2.DSCB,RECFM=U,BLKSIZE=&BLKSIZE)                 00001600
//LASTSTEP EXEC PGM=IEFBR14,REGION=4K                                    00001700
```

6. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

TRMHIST1 - EXTRACT BACKUP VOLUME INFORMATION FROM MINI-WEDGE HISTORY FILES

1. GENERAL

1.1 TRMHIST1 is a cataloged procedure developed for application systems interfacing with Mini-WEDGE. The function of this cataloged procedure is to search the STACK and UNSTACK History files for backup volume serial information pertaining to previous transmissions in the WEDGE system. This information may be used to recover transmission files if the need arises.

2. USING TRMHIST1

2.1 TRMHIST1 is invoked by an EXEC statement and an override card as follows:

```
//  EXEC  TRMHIST1
//STEPO.SØRTIN  DD  *
  (Data Cards - See Div. 2, Chap. 8, Sect. 2, Par. 5.2)
```

2.2 If no input data is submitted, an ABEND U0404 will occur with the message 'NO INPUT SUBMITTED'.

2.3 If the //STEPO.SØRTIN DD * override card is not used, an ABEND U0016 will occur.

3. EXAMPLE OF USE

3.1 Locate backup volume serial numbers from the STACK and UNSTACK History files.

```
//  EXEC  TRMHIST1
//STEPO.SØRTIN  DD  *
SØCSPSSUMHWTMHWPH74
UØCRØCHDRARAMØCØC74
```

(continued on next page)

TRMHIST1 (CONTD.)

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMHIST1
//TRMHIST1 PROC CURCHG='7400174001C*',LOC=PRDG2,R160=6400,              00000001
//       SWA=0500,SRTCTL=PRD2                                           00000101
//STEPO    EXEC PGM=SORT,REGION=110K                                    00000201
//SYSUDUMP  DD SYSOUT=A                                                 00000301
//SORTLIB   DD  DSNAME=SYS1.SORTLIB,DISP=SHR                            00000401
//SYSOUT    DD SYSOUT=A                                                 00000501
//*                                                                     00000601
//*  INPUT IS SUBMITTED VIA AN OVERRIDE     //STEPO.SORTIN DD *         00000701
//*  TYPICAL TRANSACTIONS ARE SHOWN BELOW                               00000801
//*  COL. 1 S OR U    S=STACK    U=UNSTACK                              00000901
//*  COL. 2-3  LOCATION ID  EXAMPLE SHOWS   *OC*                        00001001
//*  COL. 4-13   FILE-ID                                                00001101
//*  COL. 14-15  SENDING LOCATION                                       00001201
//*  COL. 16-17  RECEIVING LOCATION                                     00001301
//*  COL. 18-19  YEAR    (WILL EXTRACT ALL SERIAL NUMBERS FOR THIS YEAR) 00001401
//*  SOCSPSSUMHWTMHWPH74                                                00001501
//*  UOCROCHDRARAMOCOC74                                                00001601
//*                                                                     00001701
//SORTIN    DD DUMMY                                                    00001801
//SORTOUT   DD UNIT=DISK,DSN=&&FINDERS,DISP=(NEW,PASS,DELETE),          00001901
//           SPACE=(1600,(200,010),RLSE),                               00002001
//           DCB=(RECFM=FB,BLKSIZE=1600,LRECL=080)                      00002101
//SORTWK01  DD UNIT=DISK,SPACE=(TRK,(&SWA,),,CONTIG)                    00002201
//SORTWK02  DD UNIT=DISK,SPACE=(TRK,(&SWA,),,CONTIG)                    00002301
//SORTWK03  DD UNIT=DISK,SPACE=(TRK,(&SWA,),,CONTIG)                    00002401
//SYSIN     DD DISP=SHR,DSN=&SRTCTL..CONTROL(TRMHIST1)                  00002501
//*  SORT FIELDS=(1,23,CH,A)                                            00002601
//STEP1    EXEC PGM=TRMHST01,REGION=110K,PARM='RM01&CURCHG.',TIME=10    00002701
//STEPLIB   DD DSN=PRD2.LOADLIB,DISP=(SHR,PASS)                         00002801
//*                                                                     00002901
//*  THIS PROG. IS USED TO EXTRACT BACKUP VOLUME SERIAL-S FROM          00003001
//*       THE STACK AND UNSTACK HISTORY FILES.                          00003101
//*                                                                     00003201
//SYSOUT    DD SYSOUT=A                                                 00003301
//FINDERS   DD UNIT=DISK,DSN=&&FINDERS,DISP=(OLD,DELETE,DELETE),        00003401
//           DCB=(RECFM=FB,BLKSIZE=1600,LRECL=080)                      00003501
//STACKMST  DD UNIT=TAPE,                                               00003601
//           DSN=&LOC..TRM.STACK.HISTORY(+0),DISP=OLD,                  00003701
//           DCB=(SYS2.DSCB,RECFM=FB,BLKSIZE=&R160,LRECL=160)           00003801
//UNSTAMST  DD UNIT=TAPE,                                               00003901
//           DSN=&LOC..TRM.UNSTACK.HISTORY(+0),DISP=OLD,                00004001
//           DCB=(SYS2.DSCB,RECFM=FB,BLKSIZE=&R160,LRECL=160)           00004101
//IOEXTFB   DD SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)           00004201
//CBEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004301
//CSEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004401
//JJEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004501
//HMEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004601
//HWEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004701
//IHEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004801
//LEEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00004901
//MGEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00005001
//UCEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00005101
//UHEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00005201
//VCEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00005301
//WNEXTFB   DD DUMMY,SYSOUT=A,DCB=(BLKSIZE=0931,LRECL=133,RECFM=FB)     00005401
//WORKFILE  DD UNIT=DISK,DSN=&WORKFILE,                                 00005501
//           SPACE=(&R160,(200,010),RLSE),                              00005601
//           DCB=(RECFM=FB,BLKSIZE=&R160,LRECL=160)                     00005701
//* * * * * * * * * * LAST OF PROC TRMHIST1        * * * * * * * * * *  00005801
```

5. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

TRMRECVR - TRANSMISSION FILE RECOVERY

1. GENERAL

1.1 TRMRECVR is a one-step cataloged procedure that may be used to recover selected transmission files from any data set containing multiple transmission files that conform to the standards of CI 95.183, Section 35. The backup and unmatched data sets of the WDC Mini-WEDGE System are examples of data sets that could be used.

2. USING TRMRECVR

2.1 TRMRECVR is invoked by an EXEC statement coded:

```
// EXEC TRMRECVR,DSN='transmission.dsname'
```

2.2 The following DD statements must be included to execute the procedure:

```
//STEP1.SYSUT2  DD  (parameters describing the recovered data set)
//STEP1.SYSIN  DD  *
   (Select cards describing transmission files to be recovered -
     See Div. 2, Chap. 8, Sect. 2, Par. 6)
```

3. EXAMPLE OF USE

3.1

```
// EXEC TRMRECVR,DSN='PRDG2.TRM.UNSTACK.BACKUP.G0086V00'
//STEP1.SYSUT2  DD  (appropriate parameters)
//STEP1.SYSIN  DD  *
     Select cards
```

4. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMRECVR
//TRMRECVR PROC  DSN='ERROR'                                              00000100
//*             RECOVERS TRANSMISSION FILES FROM BACKUP OR                00000200
//*             UNMATCHED DATA SETS. THE USER MUST SPECIFY                00000300
//*             THE DSNAME OF THE INPUT FILE IN THE DSN                   00000400
//*             PARAMETER OF THE EXEC CARD AND SUPPLY DD                  00000500
//*             OVERRIDE CARDS - SYSUT2-FOR THE OUTPUT FILE               00000600
//*                            - SYSIN-FOR CONTROL CARDS IN               00000700
//*                EXAMPLE-                                               00000800
//*             // EXEC  TRMRECVR,DSN='PRDG2.TRM.UNSTACK.BACKUP.G0086V00' 00000900
//*             //STEP1.SYSUT2 DD DSN=PRD2.TRM.RECOVERD,DISP=....         00001000
//*             //STEP1.SYSIN   DD   *                                    00001100
//*                                                                       00001200
//*             EXAMPLE DSNAME OF THE FILE TO BE RECOVERED FROM (INPUT)-  00001300
//*                PRDG2.TRM.UNSTACK.BACKUP                               00001400
//*                PRDG2.TRM.UNSTACK.MERGED.BACKUP                        00001500
//*                PRDG2.TRM.UNSTACK.UNMATCH                              00001600
//*                PRDG2.TRM.STACK.CUMINPUT                               00001700
//*                                                                       00001800
//STEP1   EXEC  PGM=TRMUST05,REGION=40K                                   00001900
//STEPLIB   DD  DSNAME=PRD2.LOADLIB,DISP=SHR                              00001920
//SYSUDUMP DD   SYSOUT=A                                                  00002000
//RCVRYRPT DD   SYSOUT=A                                                  00002100
//SEARCHIN DD   DDNAME=SYSUT1                                             00002200
//SYSUT1   DD   DSNAME=&DSN,DISP=SHR                                      00002300
//RCVRYOUT DD   DDNAME=SYSUT2                                             00002400
//CNTLCRDS DD   DDNAME=SYSIN                                              00002500
```

TRMRECVR (CONTD.)

5. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

TRMSTKAA - COPY APPLICATION SYSTEM OUTGOING TRANSMISSION DATA SET TO STACK DISK PACK

1. GENERAL

1.1 TRMSTKAA is a cataloged procedure developed for application systems interfacing with Mini-WEDGE. The function of this cataloged procedure is to move outgoing transmission data files from an application system data set onto the STACK disk pack TRM001.

2. SYMBOLIC PARAMETERS

2.1 TRMSTKAA requires two symbolic parameters and may have three additional parameters, which are:

LOC, a required parameter, specifies the source location, for example HW, OC, OH, etc.

INPUT, a required parameter, specifies the data set name of the input data set. The parameter must be enclosed in apostrophes if the name is indexed (contains periods). This data set is assumed to be cataloged and on disk or standard label tape. If these assumptions are not true, appropriate overrides must be entered via a //STEP1.SYSUT1 DD card.

REG, an optional parameter, specifies the maximum REGION size. The default is 20K. Ordinarily this default will not need to be changed.

SPACE, an optional parameter, specifies an estimate of the number of 3330 tracks necessary to hold the data being transferred. The default is 10 tracks, which has a capacity of 110 blocks, 1000 characters in length, in the initial allocation. The value specified for SPACE is also used for secondary allocation.

BLKSIZE, an optional parameter, specifies the maximum blocksize of the transmission file being copied. The default is 1000.

3. ASP CONTROL CARD

3.1 All jobs which use TRMSTKAA must use the CLASS parameter on the ASP //*MAIN statement. This parameter must specify the transmission class assigned to the remote location and is of the form CLASS=xxWEDGE where xx is the source location, for example HW, OC, OH, etc.

TRMSTKAA (CONTD.)

4. USING TRMSTKAA

4.1 TRMSTKAA is invoked by an EXEC statement

```
// EXEC TRMSTKAA,LØC=HW,INPUT='IN.DSNAME'
```

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  TRMSTKAA
//TRMSTKAA PROC REG=20K,                                               X00000100
//             LOC=XX,                 SOURCE LOCATION (SEE LIST BELOW)X00000200
//             SPACE=10,               OUTPUT SIZE ESTIMATE (TRKS)     X00000300
//             BLKSIZE=1000,           MAXIMUM BLKSIZE                 X00000400
//             INPUT='INPUT.DSNAME'    DSNAME OF INPUT DATA SET         00000500
//*            COPY TRANSMISSION FILE FROM APPLICATION DATA SET TO      00000600
//*            LOCATION GDS FOR STACK INPUT                             00000700
//STEP1  EXEC  PGM=WECOPY,REGION=&REG                                   00000800
//SYSPRINT DD  SYSOUT=A                                                 00000900
//SYSUT1   DD  DSNAME=&INPUT,DISP=(SHR,KEEP),DCB=EROPT=ABE              00001000
//SYSUT2   DD  DSNAME=PRDG2.TRM.STACK.INPUT.&LOC.(+1),                 X00001100
//             SPACE=(TRK,(&SPACE,&SPACE),RLSE),                       X00001200
//             UNIT=DISK,VOLUME=SER=TRM001,DISP=(NEW,CATLG,DELETE),    X00001300
//             DCB=(SYS2.DSCB,DSORG=PS,RECFM=U,BLKSIZE=&BLKSIZE,       X00001400
//             EROPT=ABE)                                               00001500
//*                                                                     00001600
//*            SAMPLE EXECUTE CARD -                                    00001700
//*            // EXEC TRMSTKAA,LOC=OH,INPUT='PRDG19.XMM.TRANTAPE(0)'   00001800
//*                                                                     00001900
//*            INPUT IS ASSUMED TO BE CATALOGED AND HAVE DCB=RECFM=U    00002000
//*            INPUT IS ASSUMED TO BE ON DISK OR STANDARD LABEL TAPE    00002100
//*            IF INPUT ASSUMPTIONS ARE NOT TRUE, APPROPRIATE           00002200
//*            OVERRIDES MUST BE ENTERED VIA A //STEP1.SYSUT1 DD CARD   00002300
//*                                                                     00002400
//*            VALID LOCATIONS - CB, DJ, HW, LE, MG, OC, OH, & VC       00002500
//*            THIS PROCEDURE UPDATED 1/13/75 TO INCLUDE DCB=EROPT=ABE  00002600
```

6. REFERENCE

Division 2, Chapter 8, Section 2, WDC WEDGE Interface System.

PLICKC - PL/I CHECKOUT COMPILE

1. GENERAL

1.1 PLICKC is a one-step cataloged procedure which translates one or more PL/I procedures and retains the link-edit stubs on temporary data sets.

1.2 This cataloged procedure is intended for use in conjunction with the procedures PLICKL, PLICKLG, or PLICKG.

2. USING PLICKC

2.1 PLICKC may be invoked by an EXEC statement.

```
//  EXEC  PLICKC
```

2.2 The following DD statement which defines the source input must be added to the procedure.

```
//PLI.SYSCIN  DD  *
  source program
/*
```

3. CATALOGED PROCEDURE LISTING

```
MENBER NAME  PLICKC
//PLICKC    PROC                                                      00000100
//PLI       EXEC PGM=IEN512NS,REGION=150K,PARM='NORUN,OBJECT'         00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                               00000300
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                               00000400
//SYSPRINT DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)         00000500
//SYSLIN    DD  DSNAME=&&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),          00000600
//              SPACE=(80,(500,100))                                  00000700
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,          00000800
//              SPACE=(1024,(60,60),,CONTIG)                          00000900
//SYSITEXT DD  DSNAME=&&TEXT,UNIT=SYSDA,DISP=(MOD,PASS),              00001000
//              SPACE=(1024,(10,10,10)),DCB=BLKSIZE=1024              00001100
//SYSFORM   DD  DSNAME=&&SYSFORM,UNIT=SYSDA,DCB=BLKSIZE=400,          00001200
//              SPACE=(80,(250,250))                                  00001300
//SYSUT2    DD  DSNAME=&&SYSUT2,UNIT=SYSDA,SPACE=(80,(250,250))       00001400
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER PROGRAMMER'S GUIDE" SC33-0007.

PLICKCG - PL/I CHECKOUT COMPILE, LOAD AND GO

1. GENERAL

1.1 The PLICKCG is a two-step cataloged procedure which first translates one or more PL/I source programs and retains the intermediate texts (SYSITEXT) and the link edit stubs (SYSLIN) on temporary data sets. The second step processes the link-edit stubs with the loader and then executes the resulting load module.

1.2 A listing is always produced for translation and execution (SYSPRINT). A listing will also be produced for the loader (SYSLOUT) if the PRINT loader option applies.

2. USING PLICKCG

2.1 PLICKCG may be invoked by an EXEC statement.

```
// EXEC PLICKCG
```

2.2 The following DD statement which defines the primary input for the compiler must be added to the procedure.

```
//PLI.SYSCIN DD *
   source program
/*
```

2.3 A DD statement must be added for input data.

```
//GO.SYSIN  DD  *
  input data
/*
```

continued

PLICKCG - PL/I CHECKOUT COMPILE, LOAD AND GO (CONT.)

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME   PLICKCG
//PLICKCG   PROC
//PLI       EXEC PGM=IEN512NS,REGION=150K,PARM='NORUN,OBJECT'
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR
//SYSPRINT DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSLIN    DD   DSNAME=&&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),
//               SPACE=(80,(500,100))
//SYSUT1    DD   DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,
//               SPACE=(1024,(60,60),,CONTIG)
//SYSITEXT DD   DSNAME=&&TEXT,UNIT=SYSDA,DISP=(MOD,PASS),
//               SPACE=(1024,(10,10,10)),DCB=BLKSIZE=1024
//SYSFORM   DD   DSNAME=&&SYSFORM,UNIT=SYSDA,DCB=BLKSIZE=400,
//               SPACE=(80,(250,250))
//SYSUT2    DD   DSNAME=&&SYSUT2,UNIT=SYSDA,SPACE=(80,(250,250))
//GO        EXEC PGM=LOADER,PARM='MAP,LET',REGION=150K,
//               COND=(9,LT,PLI)
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR
//SYSLOUT   DD   SYSOUT=A
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR
//SYSPRINT DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSUT1    DD   DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,
//               SPACE=(1024,(60,60),,CONTIG)
//SYSITEXT DD   DSNAME=&&TEXT,DISP=(OLD,DELETE)
//PLIDUMP DD SYSOUT=A
//SYSLIN DD DSN=&&LOADSET,DISP=(OLD,DELETE)
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLICKCL - PL/I CHECKOUT COMPILE AND LINK

1. GENERAL

1.1 PLICKCL is a two-step cataloged procedure which first translates one or more PL/I procedures and then produces one or more load modules on a temporary data set (SYSLMOD) with the linkage editor.

2. USING PLICKCL

2.1 PLICKCL may be invoked by an EXEC statement.

```
// EXEC PLICKCL
```

2.2 The user must add a DD statement for the primary input for the compiler.

```
//PLI.SYSCIN  DD  *
        source program
/*
```

2.3 The user must add a SYSLMOD DD statement if the load module is to be placed in a permanent library.

```
//LKED.SYSLMOD DD DSNAME=LOAD(MEMBER),DISP=OLD
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLICKCL
//PLICKCL  PROC                                                         00000100
//PLI      EXEC PGM=IEN512NS,REGION=150K,PARM='NORUN,OBJECT'           00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000400
//SYSPRINT DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)           00000500
//SYSLIN   DD  DSNAME=&&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),             00000600
//             SPACE=(80,(500,100))                                     00000700
//SYSUT1   DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,             00000800
//             SPACE=(1024,(60,60),,CONTIG)                             00000900
//SYSITEXT DD  DSNAME=&&TEXT,UNIT=SYSDA,DISP=(MOD,PASS),                00001000
//             SPACE=(1024,(10,10,10)),DCB=BLKSIZE=1024                 00001100
//SYSFORM  DD  DSNAME=&&SYSFORM,UNIT=SYSDA,DCB=BLKSIZE=400,             00001200
//             SPACE=(80,(250,250))                                     00001300
//SYSUT2   DD  DSNAME=&&SYSUT2,UNIT=SYSDA,SPACE=(80,(250,250))          00001400
//LKED     EXEC PGM=IEWL,PARM='XREF,LIST',REGION=150K,                  00001500
//             COND=(9,LT,PLI)                                          C0001600
//SYSLIN   DD  DSNAME=&&LOADSET,DISP=(OLD,DELETE)                       00001700
//         DD  DDNAME=SYSIN                                             00001800
//SYSLMOD  DD  DSNAME=&&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),           00001900
//        SPACE=(1024,(50,,1))                                          00002000
//SYSUT1   DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,             00002100
//             SPACE=(1024,(200,20))                                    00002200
//SYSPRINT DD  SYSOUT=A                                                 00002300
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLICKCLG - PL/I CHECKOUT COMPILE, LINK-EDIT, AND GO

1. GENERAL

1.1 PLICKCLG is a three-step cataloged procedure which compiles one or more PL/I source modules, processes the translated source modules with the linkage editor and produces one or more load modules on a temporary data set (SYSLMOD). The third step executes the load module stored with the member name GO, specified in the DSNAME parameter for SYSLMOD.

1.2 The DDNAME parameter in the second DD statement for SYSLIN in the procedure step LKED allows you to concatenate additional modules, and also control statements, to the primary input data set for the linkage editor, by specifying their data sets in one or more new DD statements, with the qualified ddname LKED.SYSIN for the first, and blank name fields for the rest.

2. USING PLICKCLG

2.1 PLICKCLG may be invoked by an EXEC statement.

```
//  EXEC PLICKCLG
```

2.2 The following DD statement must be added to supply source input for the compile step.

```
//PLI.SYSCIN DD *
   source program
/*
```

2.3 The following DD statement must be added to supply input for the GO step.

```
//GO.SYSIN DD *
   input data
/*
```

2.4 If the user intends to execute the same load module more than once, you must override this DD statement for SYSLMOD to prevent the load module from automatically being deleted, for example:

```
//GO.SYSLMOD DD DISP=(OLD,PASS)
```

PLICKCLG - PL/I CHECKOUT COMPILE, LINK EDIT AND GO (CONT.)

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLICKCLG
//PLICKCLG PROC                                                         00000100
//PLI       EXEC PGM=IEN512NS,REGION=150K,PARM='NORUN,OBJECT'           00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000400
//SYSPRINT DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)          00000500
//SYSLIN    DD  DSNAME=&&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),            00000600
//              SPACE=(80,(500,100))                                    00000700
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,            00000800
//              SPACE=(1024,(60,60),,CONTIG)                            00000900
//SYSITEXT DD   DSNAME=&&TEXT,UNIT=SYSDA,DISP=(MOD,PASS),               00001000
//              SPACE=(1024,(10,10,10)),DCB=BLKSIZE=1024                00001100
//SYSFORM   DD  DSNAME=&&SYSFORM,UNIT=SYSDA,DCB=BLKSIZE=400,            00001200
//              SPACE=(80,(250,250))                                    00001300
//SYSUT2    DD  DSNAME=&&SYSUT2,UNIT=SYSDA,SPACE=(80,(250,250))         00001400
//LKED      EXEC PGM=IEWL,PARM='XREF,LIST',REGION=150K,                 00001500
//              COND=(9,LT,PLI)                                         00001600
//SYSLIN    DD  DSNAME=&&LOADSET,DISP=(OLD,DELETE)                      00001700
//          DD  DDNAME=SYSIN                                            00001800
//SYSLMOD   DD  DSNAME=&&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),          00001900
//        SPACE=(1024,(50,,1))                                          00002000
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,            00002100
//              SPACE=(1024,(200,20))                                   00002200
//SYSPRINT DD   SYSOUT=A                                                00002300
//GO        EXEC PGM=*.LKED.SYSLMOD,REGION=150K,                        00002400
//              COND=((9,LT,PLI),(9,LT,LKED))                           00002500
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00002600
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00002700
//SYSPRINT DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)          00002800
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,            00002900
//              SPACE=(1024,(60,60),,CONTIG)                            00003000
//SYSLMOD   DD  DSNAME=&&GOSET,DISP=(OLD,DELETE)                        00003100
//SYSITEXT DD   DSNAME=&&TEXT,DISP=(OLD,DELETE)                         00003200
//PLIDUMP DD SYSOUT=A                                                   00003300
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLICKG - PL/I CHECKOUT LOAD AND GO

1. GENERAL

1.1 PLICKG is a one-step cataloged procedure which processes the primary input with the loader and executes the resulting load module.

2. USING PLICKG

2.1 PLICKG may be executed by an EXEC statement.

```
// EXEC PLICKG
```

2.2 The user must always add a DD statement defining the primary input data set.

```
//GO.SYSLIN  DD  DSNAME=OBJECT,DISP=SHR
```

2.3 The user must always add a DD statement defining the primary text data set.

```
//GO.SYSITEXT  DD  DSNAME=TEXT,DISP=SHR
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLICKG
//PLICKG    PROC                                                        00000100
//GO        EXEC PGM=LOADER,PARM='MAP,LET',REGION=150K                  00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSLIN    DD  DDNAME=SYSIN                                            00000400
//SYSLOUT   DD  SYSOUT=A                                                00000500
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000600
//SYSPRINT DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)          00000700
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,            00000800
//              SPACE=(1024,(60,60),,CONTIG)                            00000900
//PLIDUMP DD SYSOUT=A                                                   00001000
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLICKI - PL/I CHECKOUT AND GO

1. GENERAL

1.1 PLICKI is a one-step cataloged procedure which invokes a load module which includes one or more link edit stubs produced by the translation stage of the checkout compiler.

2. USING PLICKI

2.1 PLICKI may be invoked by an EXEC statement.

```
// EXEC PLICKI,LOAD=xxx
```

where xxx must be the name of the load module or a backward reference to the DD statement defining the load module data set in a previous step.

2.2 The user must always add a DD statement defining the intermediate text data set (SYSITEXT) for the procedure step GO.

2.3 If the load module being executed is in a non-system (i.e. SYS1.xxxx) library a // STEPLIB DD card is needed to point to the data set where the load module resides.

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLICKI
//PLICKI   PROC                                                      00000100
//GO       EXEC PGM=&LOAD,REGION=150K                                00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                              00000300
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                              00000400
//SYSPRINT DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)        00000500
//SYSUT1   DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,          00000600
//             SPACE=(1024,(60,60),,CONTIG)                          00000700
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLICKLG - PL/I CHECKOUT LOAD AND GO

1. GENERAL

1.1 PLICKLG is a two-step cataloged procedure which first link edits input object code into load modules on a temporary data set (SYSLMOD) and then executes the load module.

2. USING PLICKLG

2.1 PLICKLG may be executed by an EXEC statement.

```
// EXEC PLICKLG
```

2.2 The user must always add a DD statement defining the input to the linkage editor.

either

```
//LKED.SYSLIN DD DSNAME=OBJECT
```

or

```
//LKED.SYSIN DD DSNAME=OBJECT
```

2.3 The user must always add a DD statement defining the intermediate text data set for the procedure GO step.

```
//GO.SYSITEXT DD DSNAME=TEXT,DISP=OLD
```

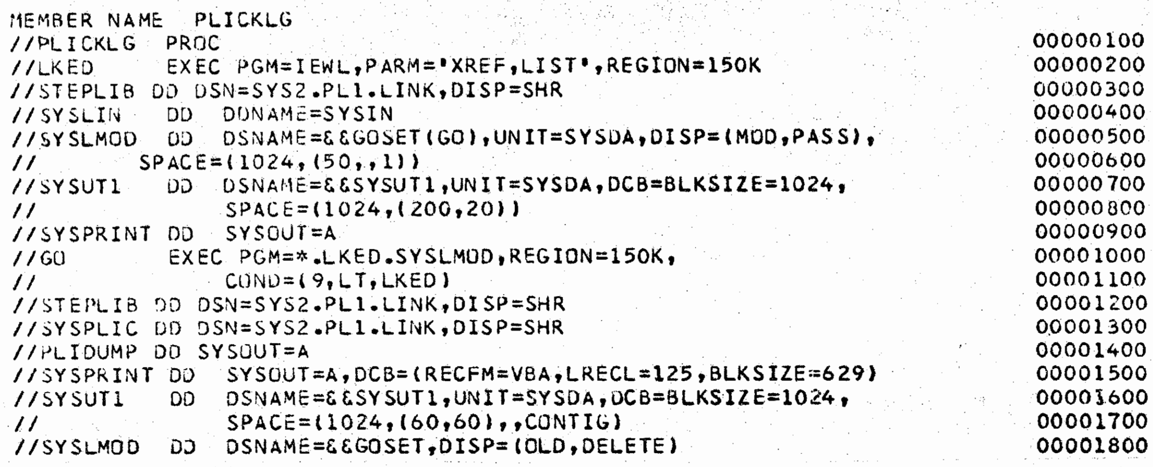
2.4 A DD statement may be added for user input to the GO step of the procedure.

```
//GO.SYSIN DD *
   input data
/*
```

continued

PLICKLG - PL/I CHECKOUT LOAD AND GO (CONT.)

3. CATALOGED PROCEDURE LISTING



```
MEMBER NAME  PLICKLG
//PLICKLG  PROC                                                         00000100
//LKED     EXEC PGM=IEWL,PARM='XREF,LIST',REGION=150K                   00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSLIN    DD  DDNAME=SYSIN                                            00000400
//SYSLMOD   DD  DSNAME=&&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),          00000500
//        SPACE=(1024,(50,,1))                                          00000600
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,            00000700
//              SPACE=(1024,(200,20))                                   00000800
//SYSPRINT DD   SYSOUT=A                                                00000900
//GO        EXEC PGM=*.LKED.SYSLMOD,REGION=150K,                        00001000
//              COND=(9,LT,LKED)                                        00001100
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00001200
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00001300
//PLIDUMP DD SYSOUT=A                                                   00001400
//SYSPRINT DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)          00001500
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,            00001600
//              SPACE=(1024,(60,60),,CONTIG)                            00001700
//SYSLMOD   DD  DSNAME=&&GOSET,DISP=(OLD,DELETE)                        00001800
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLICKR - PL/I CHECKOUT COMPILE

1. GENERAL

1.1 PLICKR is a one-step cataloged procedure which compiles, interprets, and executes one or more programs. By avoiding the link-edit step, it is the most efficient cataloged procedure for this type of program.

1.2 The user must always add a DD statement defining the primary input (SYSCIN). The only output is a listing (SYSPRINT).

2. USING PLICKR

2.1 PLICKR may be invoked by an EXEC statement.

```
// EXEC PLICKR
```

2.2 The following DD statement must be added to the procedure.

```
//GO.SYSCIN DD *
    source program
*DATA;
  input data
*PROCESS MACRO;
  source program
*DATA;
/*
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLICKR
//PLICKR    PROC                                                           00000100
//GO        EXEC PGM=IEN512NS,REGION=150K,PARM='RUN,NOOBJECT'              00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                   00000300
//SYSPLIC DD DSN=SYS2.PL1.LINK,DISP=SHR                                   00000400
//SYSPRINT DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)              00000500
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=SYSDA,DCB=BLKSIZE=1024,                 00000600
//              SPACE=(1024,(60,60),,CONTIG)                                 00000700
//SYSFORM   DD  DSNAME=&&SYSFORM,UNIT=SYSDA,DCB=BLKSIZE=400,                 00000800
//              SPACE=(80,(250,250))                                         00000900
//SYSUT2    DD  DSNAME=&&SYSUT2,UNIT=SYSDA,SPACE=(80,(250,250))              00001000
```

4. REFERENCE

4.1 "OS PL/I CHECKOUT COMPILER: PROGRAMMER'S GUIDE" SC33-0007.

PLIXC - PL/I OPTIMIZING COMPILE

1. GENERAL

1.1 PLIXC is a one-step procedure which compiles and optimizes a source module, in which the options specified are OBJECT and NODECK.

2. USING PLIXC

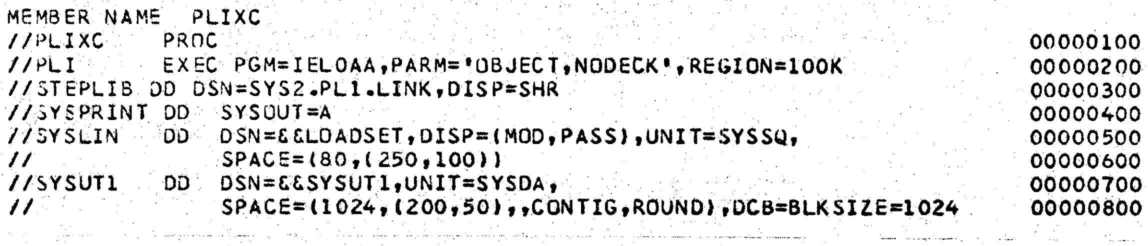
2.1 PLIXC may be invoked by an EXEC statement.

```
//  EXEC  PLIXC
```

2.2 The user must always supply a DD statement for the input data set.

```
//PLI.SYSIN  DD  *
   input
/*
```

3. CATALOGED PROCEDURE LISTING



```
MEMBER NAME  PLIXC
//PLIXC     PROC                                                        00000100
//PLI       EXEC PGM=IELOAA,PARM='OBJECT,NODECK',REGION=100K            00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSPRINT DD  SYSOUT=A                                                 00000400
//SYSLIN    DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,               00000500
//              SPACE=(80,(250,100))                                    00000600
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSDA,                                00000700
//              SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024    00000800
```

4. REFERENCE

4.1 "OS PL/I OPTIMIZING COMPILER: PROGRAMMER'S GUIDE" SC33-0006.

PLIXCG - PL/I OPTIMIZING COMPILE AND GO

1. GENERAL

1.1 PLIXCG is a two-step cataloged procedure which in the first step compiles and optimizes the source into an object module. In the second step, the loader processes the object module and executes the resultant executable program immediately.

2. USING PLIXCG

2.1 PLIXCG may be invoked with an EXEC statement.

```
//   EXEC   PLIXCG
```

2.2 The user must add a DD statement for the input to the compile step.

```
//PLI.SYSIN DD *
   source input
/*
```

2.3 The user may also add a DD statement for input to the GO step.

```
//GO.SYSIN DD *
  input data
/*
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLIXCG
//PLIXCG PROC LKLBDSN='SYS2.PL1.BASE'                                   00000100
//PLI       EXEC PGM=IELOAA,PARM='OBJECT,NODECK',REGION=100K             00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSPRINT DD  SYSOUT=A                                                 00000400
//SYSLIN   DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,                00000500
//             SPACE=(80,(250,100))                                     00000600
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,                                 00000700
//             SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024     00000800
//GO        EXEC PGM=LOADER,PARM='MAP,PRINT',REGION=100K,               00000900
//             COND=(9,LT,PLI)                                          00001000
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00001100
//SYSLIB   DD  DSN=&LKLBDSN,DISP=SHR                                    00001200
//        DD DSN=SYS2.PL1.BASE,DISP=SHR                                 00001300
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)                          00001400
//SYSLOUT  DD  SYSOUT=A                                                 00001500
//SYSPRINT DD  SYSOUT=A                                                 00001600
//PLIDUMP DD SYSOUT=A                                                   00001700
```

4. REFERENCE

4.1 "OS PL/I OPTIMIZING COMPILER: PROGRAMMER'S GUIDE SC33-0006.

PLIXCL - PL/I OPTIMIZING COMPILE AND LINK-EDIT

1. GENERAL

1.1 PLIXCL is a two-step catalog procedure; the first step PLI is identical to the catalog procedure PLIXC and the second step, LKED, invokes the linkage editor to link edit the object module produced in the first procedure step.

2. USING PLIXCL

2.1 PLIXCL may be invoked by an EXEC statement.

```
// EXEC PLIXCL
```

2.2 The user must always add a DD statement for the input to the compiler step.

```
//PLI.SYSIN DD *
     input
/*
```

2.3 If the user wants to retain the output of the linkage edit step beyond the duration of this job, a DD statement must be added.

```
//LKED.SYSLMOD DD DSN=EXAMPLE
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLIXCL
//PLIXCL PROC LKLBDSN='SYS2.PL1.BASE'                                   00000100
//PLI       EXEC PGM=IELOAA,PARM='OBJECT,NODECK',REGION=100K            00000200
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR                                 00000300
//SYSPRINT DD  SYSOUT=A                                                 00000400
//SYSLIN    DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,               00000500
//              SPACE=(80,(250,100))                                    00000600
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSDA,                                00000700
//              SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024    00000800
//LKED      EXEC PGM=IEWL,PARM='XREF,LIST',COND=(9,LT,PLI),REGION=100K  00000900
//SYSLIB    DD  DSN=&LKLBDSN,DISP=SHR                                   00001000
//         DD DSN=SYS2.PL1.BASE,DISP=SHR                                00001100
//SYSLMOD   DD  DSN=&&GOSET(GO),DISP=(MOD,PASS),UNIT=SYSDA,             00001200
//         SPACE=(1024,(50,,1))                                         00001300
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSDA,                                00001400
//              SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024    00001500
//SYSPRINT DD  SYSOUT=A                                                 00001600
//SYSLIN    DD  DSN=&&LOADSET,DISP=(OLD,DELETE)                         00001700
//          DD  DDNAME=SYSIN                                            00001800
```

4. REFERENCE

4.1 "OS PL/I OPTIMIZING COMPILER: PROGRAMMER'S GUIDE" SC33-0006.

PLIXCLG - PL/I OPTIMIZING COMPILE, LINK EDIT AND GO

1. GENERAL

1.1 PLIXCLG is a three-step cataloged procedure which compiles and optimizes sources in the first step creating an object module for the second step. The second step link-edits this object code into a load module which is executed in the third step.

2. USING PLIXCLG

2.1 PLIXCLG may be invoked by an EXEC statement.

```
// EXEC PLIXCLG
```

2.2 Input data for the compilation step must be specified in a DD statement.

```
//PLI.SYSIN DD *
  source input
/*
```

2.3 Input data may be specified for the GO step with a DD statement.

```
//GO.SYSIN DD *
  input data
/*
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLIXCLG
//PLIXCLG PROC LKLBDSN='SYS2.PL1.BASE'
//PLI        EXEC PGM=IELOAA,PARM='OBJECT,NODECK',REGION=100K
//STEPLIB DD DSN=SYS2.PL1.LINK,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSLIN    DD   DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,
//               SPACE=(80,(250,100))
//SYSUT1    DD   DSN=&&SYSUT1,UNIT=SYSDA,
//               SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024
//LKED       EXEC PGM=IEWL,PARM='XREF,LIST',COND=(9,LT,PLI),REGION=100K
//SYSLIB    DD   DSN=&LKLBDSN,DISP=SHR
//        DD DSN=SYS2.PL1.BASE,DISP=SHR
//SYSLMOD   DD   DSN=&&GOSET(GO),DISP=(MOD,PASS),UNIT=SYSDA,
//        SPACE=(1024,(50,,1))
//SYSUT1    DD   DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(200,20)),
//               DCB=BLKSIZE=1024
//SYSPRINT DD   SYSOUT=A
//SYSLIN    DD   DSN=&&LOADSET,DISP=(OLD,DELETE)
//          DD   DDNAME=SYSIN
//GO         EXEC PGM=*.LKED.SYSLMOD,COND=((9,LT,PLI),(9,LT,LKED)),
//               REGION=100K
//STEPLIB DD DSN=SYS2.PLI.LINK,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//PLIDUMP DD SYSOUT=A
```

4. REFERENCE

4.1 "OS PL/I OPTIMIZING COMPILER: PROGRAMMER'S GUIDE" SC33-0006.

PLIXLG - PL/I OPTIMIZING LINK-EDIT AND GO

1. GENERAL

1.1 PLIXLG is a two-step cataloged procedure which link-edits the output from a PLIXC procedure into a load module and then executes the load module.

2. USING PLIXLG

2.1 PLIXLG may be invoked with an EXEC statement.

```
// EXEC PLIXLG
```

2.2 The user must add a DD statement for the input to the linkage edit step.

```
//LKED.SYSIN DD *
    or
//LKED.SYSLIN DD *
```

2.3 Input to the GO step may be provided with a DD card.

```
//GO.SYSIN DD *
   input data
/*
```

3. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  PLIXLG
//PLIXLG    PROC LKLBDSN='SYS1.PLIBASE'                                      00000100
//LKED      EXEC PGM=IEWL,PARM='XREF,LIST',REGION=100K                       00000200
//SYSLIB    DD  DSN=&LKLBDSN,DISP=SHR                                        00000300
//          DD  DSN=SYS1.PLIBASE,DISP=SHR                                    00000400
//SYSLMOD   DD  DSN=&&GOSET(GO),DISP=(MOD,PASS),UNIT=SYSDA,                  00000500
//              SPACE=(1024,(50,20,1),RLSE)                                  00000600
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSDA,                                     00000700
//              SPACE=(1024,(200,20),,CONTIG,ROUND),DCB=BLKSIZE=1024         00000800
//SYSPRINT DD   SYSOUT=A                                                     00000900
//SYSLIN    DD  DDNAME=SYSIN                                                 00001000
//GO        EXEC PGM=*.LKED.SYSLMOD,COND=(9,LT,LKED),REGION=100K             00001100
//SYSPRINT DD   SYSOUT=A                                                     00001200
```

4. REFERENCE

4.1 "OS PL/I OPTIMIZING COMPILER: PROGRAMMER'S GUIDE" SC33-0006.

CATALIST - LIST CATALOG NODE POINTS

1. GENERAL

CATALIST is a catalogued procedure which facilitates use of the Western Electric Utility Program CATALIST. CATALIST provides more information than the IBM IEHLIST utility and prints the data in a more readable form.

2. SYMBOLIC PARAMETER

2.1 CATALIST requires an INDEX parameter which supplies the NODE point(s) to be listed.

3. USING CATALIST

3.1 The procedure is invoked with an EXEC card and the required INDEX parameter.

```
//   EXEC   CATALIST,INDEX='node'
```

where 'node' is of the format: PRDGxx.PSI(xx is the PGN)

3.2 If more than one PSI is to be listed, the field should be coded as INDEX='PRDn.PSI,PRDn.PSI'

3.3 If the data sets for a particular system do not use the naming convention PRDGnn.PSI.yyyyy or PRDTnn.PSI.yyyyy, the user should only specify the first level of the data set name; otherwise the user would have to specify each exact second level that is used in that particular system.

4. EXAMPLES OF USE

4.1 List the entire catalog of one system.

```
//   EXEC   CATALIST,INDEX='PRDT6.PSI'
```

4.2 List the entire catalog of two systems which are assigned to different Production Group Numbers (PGNs).

```
//   EXEC   CATALIST,INDEX='PRDG4.PSI'
//   EXEC   CATALIST,INDEX='PRDG6.PSI'
```

4.3 List the entire catalog of two systems which are assigned to the same PGN.

```
//   EXEC CATALIST,INDEX='PRDG4.PSI,PRDG4.PSI'
```

CATALIST - LIST CATALOG NODE POINTS (CONT.)

5. CATALOGED PROCEDURE LISTING

```
MEMBER NAME  CATALIST
//CATALIST  PROC  SYSRES=ASP31A,INDEX='(PRDO.ERROR)'                   00000100
//LIST EXEC PGM=CATALIST,PARM='&SYSRES&INDEX'                          00000200
//SYSPRINT DD SYSOUT=A                                                 00000300
//DD1  DD  DISP=SHR,UNIT=DISK,VOL=SER=PROD04                           00000400
//DD2  DD  DISP=SHR,UNIT=DISK,VOL=SER=PROD06                           00000500
//DD3  DD  DISP=SHR,UNIT=DISK,VOL=SER=PROD10                           00000600
//DD4  DD  DISP=SHR,UNIT=DISK,VOL=SER=PROD12                           00000700
//DD5  DD  DISP=SHR,UNIT=DISK,VOL=SER=TEST05                           00000800
//DD6  DD  DISP=SHR,UNIT=DISK,VOL=SER=TEST11                           00000900
```

6. REFERENCE

6.1 Division 3, Chapter 2, Section 2, CATALIST.

CATAUTIL - CATALOG MAINTENANCE UTILITY

1. GENERAL

CATAUTIL is a catalogued procedure which facilitates use of the Western Electric utility program CATAUTIL, which performs functions on the catalog that are not available through IEHPROGM.

2. USING CATAUTIL

2.1 The procedure is invoked with an EXEC card.

```
// EXEC CATAUTIL
```

2.2 The following DD statement must be added to the procedure.

```
//SYSIN DD *
  (control cards)
```

2.21 Control cards are required as described in Division 3, Chapter 2, Section 2, CATAUTIL.

3. EXAMPLES OF USE:

3.1 Delete an index.

```
//  EXEC  CATAUTIL
//SYSIN  DD  *
  DLTINDEX  INDEX=DEV05.F99
```

3.2 Change a Generation Data Group index.

```
//  EXEC  CATAUTIL
//SYSIN  DD  *
  CHNGBLDG  INDEX=PRDG22.F00.TRANSNET,OPTION=N
```

CATAUTIL - CATALOG MAINTENANCE UTILITY (CONT.)

4. CATALOGUED PROCEDURE LISTING

```
//CATAUTIL PROC
//CATAUTIL EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
```

5. RESTRICTIONS

5.1 CATAUTIL may not be used for a generation data set which contains an entry which resides on more than five reels.

6. REFERENCE

Division 3, Chapter 2, Section 2, CATAUTIL

VSBASIC - VS BASIC TRANSLATE AND RUN

1. GENERAL

VSBASIC is a one step cataloged procedure to translate and run a BASIC source module.

2. USING VSBASIC

2.1 VSBASIC may be invoked by an EXEC statement:

```
// EXEC VSBASIC
```

2.2 The following DD statements must be added to the procedure.

2.21 A CONTROL DD statement is required as follows:

```
//CONTROL  DD  *
RUN  ABC  SOURCE
/*
```

Where ABC is the name of the source program which appears in a DD statement named ABC

2.22 A DD statement for the source program with the name of the source program as the name of the DD statement.

```
//ABC DD  *
      .
      .
      .
      BASIC PROGRAM
      .
      .
      .

/*
```

3. EXAMPLE OF USE

```
//  EXEC  VSBASIC
//CONTROL DD *
RUN  ABC  SOURCE
/*
//ABC    DD    *
(source statements for ABC)
/*
```

VSBASIC - VS BASIC TRANSLATE AND RUN (CONT.)

4. CATALOG PROCEDURE LISTING

```
//VSBASIC  PROC                                                         00000100
//ST1      EXEC  PGM=ICDOSBSC                                           00000200
//STEPLIB  DD    DSN=SYS2.BASIC.LINK,DISP=SHR                           00000300
//SYSPRINT DD    SYSOUT=A                                               00000400
//SYSUDUMP DD    SYSOUT=A                                               00000500
```

5. REFERENCE

IBM S/370 VSBASIC OS/VS AND DOS/VS PROGRAMMER´S GUIDE,
SC 28-8308

MISCELLANEOUS INFORMATION

UPDATING OF PROGRAMMING STANDARDS AND REFERENCE MANUAL

SECTION 1 - ESTABLISHING AND CHANGING STANDARDS

1. INTRODUCTION

1.1 Current programming standards and relevant miscellaneous information are documented in this manual. As used herein, a standard is the prescribed procedure for accomplishing a specific task. Hence, it follows that as new and improved procedures are formulated and proposed as standards there is need for a procedure to change (modify, add or delete) these programming standards and subsequently this manual. This process of change involves many individuals and groups. System planners and programmers will be required to modify system hardware and software, the operating organization will need to adjust their routines, the application programmer will be called on to modify application programs and programming procedures, and the Control Group must plan, coordinate and document the change. These individuals and groups will need time to systematically prepare for and conform to the change. The following sequence of events, the standard procedure for changing standards and, therefore, this manual will assure each the time required.

2. PROCEDURE

2.1 Proposals for new standards and to change (modify, add to or delete) existing standards should be forwarded, in writing, to the Warrenville Data Center Control Group. The proposals can be originated by any application programmer, system planner, system programmer, operating section chief, Control Group member or their supervisor; located at either the central or any remote data center.

2.2 The Control Group shall promptly examine each proposal for feasibility, compatability and relevance and shall determine what course of action is to be taken.

2.21 Acceptable proposals, with or without Control Group modification, shall be documented by the Control Group and submitted to all interested organizations for comment. A reasonable deadline shall be established for receiving comments on each proposal.

2.22 Rejected proposals shall be returned by the Control Group to their source with a written explanation.

2.3 The Control Group shall promptly consider all comments received before the specified deadline. The original proposal shall be reexamined with respect to submitted comments. The Control Group shall determine what course of action is to be pursued.

2.31 Proposals determined to be essentially unchallenged via comments shall be prepared for implementation. The Control Group shall prepare an implementation check list and inform each functional organization, in writing, of their required participation.

2.32 Proposals determined to be unacceptable after consideration of comments shall be returned to their source with a written explanation.

2.4 The Control Group shall prepare and distribute an "Information System Memorandum - Advance Information" detailing the new or changed procedure to be implemented. The memorandum shall inform the programmer of the procedure to be implemented, the effect on their efforts, and the projected implementation plan.

2.5 The Control Group shall document or have documented by the functional organization the new or changed standard (and/or associated miscellaneous information). The documentation shall be issued as an update to the Programming Standards and Reference Manual prior to the planned implementation date.

2.6 The Control Group shall be functional for co-ordinating and expediting the implementation effort.

MISCELLANEOUS INFORMATION

UPDATING OF PROGRAMMING STANDARDS AND REFERENCE MANUAL

SECTION 2 - PREPARATION AND DISTRIBUTION OF MANUAL UPDATE MATERIALS

1. INTRODUCTION

1.1 Update materials for the Warrenville Data Center Programming Standards and Reference Manual shall be prepared and distributed per the instructions of this section.

1.2 The Manual was structured to update and reissue pages individually. A page is identified by division, chapter, section and page number. As updates are prepared, this structure shall be maintained. If additional pages are to be prepared, it shall be done in such a way as to avoid the use of decimal notations for chapter and section numbers. A one-place decimal notation shall be allowed for page number. If two or more places are required, the entire section should be re-numbered and re-issued.

2. PAGE FORMAT

2.1 Pages for the manual shall be typed on standard 8 1/2 X 11 inch bond paper and shall conform in format to that of the original issue (see Exhibit A for model).

2.2 Margins shall be as follows:

Top - 1/2 inch (3 lines)
Bottom - 1 inch (6 lines)
Left - 1 1/4 inches (15 spaces)
Right - 3/4 inch (9 spaces)

2.3 The left heading, section page number, and right heading shall be placed on each page.

2.31 The left heading is the title block and shall always contain the following information:

Western Electric Company
Warrenville Data Center
PROGRAMMING S & R MANUAL

3.32 The section page number shall be centered between the left and right headers. All pages of the section shall be numbered in one sequence including the first page and any appendices and/or exhibits added.

2.33 The right heading is the major page identification information. It shall be of the following format with the variable data (XX and YYYYYYYYYYYY) inserted as required.

Division XX Chapter XX
Section XX YYYYYYYYYYY
Issue XX Date XX/XX/XX

YYYYYYYYYYY - space to identify exhibits or appendices, formatted as follows:

Exhibit X (X an alphabetic character)
Appendix X (X an alphabetic character)

2.4 The first page of each section, appendix, and exhibit shall have division, chapter, and section titles included as illustrated in Exhibit A. These title lines shall be omitted from continuation pages.

2.5 The topics and paragraphs of each section, appendix and exhibit shall be numbered as illustrated in Exhibit A. Decimal number notations shall be used for paragraph numbering only. When it becomes necessary to insert topics the section shall be renumbered and reissued as required.

3. ISSUE NUMBER

3.1 Each page shall contain issue number and date. (Paragraph 2.33). An original page shall be "Issue 1". When reissued, the page issue number shall be increased by one and the date modified to indicate the day the reissue is to become effective.

4. CHANGE FLAG

4.1 Any page reissued to modify the existing text shall contain a heavy vertical line drawn 1/2 inch from the right edge of the sheet to flag that paragraph having new or changed text. Deleted material shall be indicated with a note placed in the lower margin.

5. DISTRIBUTION

5.1 The Control Group shall maintain a Manual Distribution List. Manual update materials shall be distributed to those organizations and individuals on the current list.

6. PAGE-ISSUE LIST

6.1 The Control Group shall maintain a current list of pages in the Manual. The list shall include the following data for each page.

ISSUE NUMBER
ISSUE DATE (Effective date)
CHANGE CODE

6.2 The change code shall be one of the following:

ƀ	=	no change
A	=	page added
M	=	page modified
D	=	page deleted

6.3 A current issue of the PAGE-ISSUE LIST shall be included with the distribution of any update materials.

(Margin-1/2")

(Margin-1 1/4")

(Triple Space) (Margin-3/4"

. . . DIVISION TITLE . . .

(Double Space)

. . . CHAPTER TITLE . . .

(Double Space)

SECTION X - . . . SECTION TITLE . . .

(Triple Space)

1. <u>AAAAAA</u>

1.1 Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

1.2 Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

2. <u>BBBBBB</u>

2.1 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.11 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.12 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.13 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.131 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.132 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.2 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

2.3 Bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

3. <u>CCCCCC</u>

3.1 Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

(Margin-1")

WDe

DIVISION 4 - MISCELLANEOUS INFORMATION

CHAPTER 2 - INFORMATION SYSTEMS MEMORANDUM

SECTION 3 - SPECIAL INFORMATION

SUBJECT - IMS SYSGEN SCHEDULE

1. General

1.1 Increases in requests for IMS System Generations have made it necessary to increase the time between the request deadline and effective date of the SYSGEN's. Effective September 1, 1977, all IMS on-line requests must be received at Warrenville by noon on Friday of the 2nd week prior to the effective date (i.e. 6 working days prior to the effective date instead of the current 4 working days).

1.2 The effective date (IMS Cold Start) will also be changed to Tuesdays instead of Mondays, effective September 1.

1.3 Attached is a revised schedule for the remainder of 1977.

1.4 All 4-character PSI conversion will be done in the normal SYSGEN schedules.

2. Questions

2.1 If there are any questions or comments, contact Trudy Lyvers (391-5371).

T Lyvers
T. LYVERS - 9412

APPROVED R J Mele
R. J. MELE 9411

APPROVED P. C. Free
P. C. FREE 9412

APPROVED P. C. Free for
R. R. SORENSEN 9415

DIVISION 4 - MISCELLANEOUS INFORMATION

CHAPTER 2 - INFORMATION SYSTEMS MEMORANDUM

SECTION 3 - SPECIAL INFORMATION

SUBJECT - NEW RELEASE OF LIBRARIAN

1. GENERAL

1.1 Release 5.0 of the LIBRARIAN will be installed at Warrenville on July 7, 1975. In addition to correcting several problem areas the new release has some added features which greatly enhance the LIBRARIAN.

1.2 The installation of the new release of the LIBRARIAN requires no change to any current Librarian control cards or source modules.

2. ADDITIONAL FEATURES

2.1 The -LANG command enables a user to store a three byte language code with the source module.

2.2 A new option on the -OPT control card is NOPC which suppresses page eject between modules on the updated listing. Also use of this parameter will suppress any spacing on the frontispiece.

2.3 The -EDIT control card parameter "STR=nn" provides the capability of specifying the column number the search is to start on.

2.4 The Librarian will not allow a user to attempt to add a module without any input cards.

2.5 The programmer has the option of overriding the LIBRARIAN generated password when adding a module by specifying his own four character code as an option on the -ADD card.

2.6 The -INC commands may be nested to a maximum of five levels with one obvious restriction (i.e. a module may not include itself nor any module in a higher level) which prevents looping.

2.7 Previously the -AUX card could only be placed after an -ADD card. Now, the -AUX card may also be used after the -REP or -INS command when selecting a module already on the master file.

3. RESOLVED PROBLEM AREAS

3.1 The UTILITY option on the -OPT card has been modified so that it no longer generates a -END card.

3.2 If the TEMP option is used on -ADD card the LIBRARIAN will reject the update and provide a diagnostic message indicating such.

3.3 If a -REP or -DEL card sequence number was beyond the range of the module the next -SEL was previously processed twice. This problem has been corrected in version 5.0.

3.4 A modification has been made to allow zero as an acceptable "SEQ1" on a -INC, -PRINT or -PUNCH card and "LAST" as an allowable "SEQ2".

3.5 The hash count check has been modified to check for all updating operations with the exception of "-REP ALL", even -EDIT and -FILL which previously were not checked.

3.6 The name of any -INC (included) module is checked against the ENQUEUE list to prevent the multiple ENQUEUE ABEND.

4. PROBLEMS/QUESTIONS

4.1 Any questions and/or problems concerning this matter should be directed to J. E. Tellez Dept 9412 (8-391-5254).

J. E. Tellez
J. E. TELLEZ - 9412

Approved: J. Loboda
J. LOBODA - 9412

WDC

DIVISION 4 - MISCELLANEOUS INFORMATION

CHAPTER 2 - INFORMATION SYSTEM MEMORANDUM

SECTION 3 - SPECIAL INFORMATION

SUBJECT - CONVERSION TO 6250 BPI TAPE DRIVES

1. GENERAL

1.1 In line with our continuing effort to improve service and to increase overall system throughput, Warrenville will replace all 64 9 track, 1600 BPI, STC tape drives with 9 track, 6250 BPI STC tape drives in late December, 1976. This change will reduce tape errors and job I/O time. The 6250 BPI tape drives automatically clean the tape before it is passed under the read/write head in addition to providing significantly enhanced error handling.

1.2 The hardware will be installed in several stages on December 29, 1976, with the following final configuration:

58	MOD5 drives 6250/1600 dual density drives defaulting to 6250 BPI on output (available December 29, 1976)
2	MOD7 high speed dual density drives (available on December 12, 1976)
2	MOD5 drives for MITS/IMS use
2	1600/800/200 dual density drives

The major part of the drive exchange will be done on December 26 through December 28, 1976, when the data center use will be low due to the Christmas holidays thus minimizing the impact on production schedules.

1.3 The new dual density 6250/1600 BPI drives will transfer 1600 BPI data 30% slower than the current drives. The impact of this slower speed on the data center needs to be minimized by copying 1600 BPI tapes to 6250 BPI prior to the data center´s return to full operation on January 3, 1977 after the Christmas/New Year´s break which is also the first fiscal week of processing. Most copying will be scheduled for December 30 and 31 with limited copying done after December 12.

1.4 Each location has a 6250 tape copying coordinator who, along with the application programmers at his location will identify those files to be copied. Candidates for copying are those tape files over 1 million bytes in size and those to be read after January 3 at 1600 BPI. Smaller files with significant impact on tape usage during the first fiscal week should also be considered.

2. COPYING PROCEDURES

2.1 Cataloged procedures are provided for the copying to insure uniformity and reliability of the copying. Each copy execution will be followed by a tape compare to insure that the copied files are identical to the old files. If the compare fails, the step and job will ABEND leaving the 1600 BPI tape intact. All 1600 BPI tapes that are copied will be retained until their normal expiration unless some overt action is taken by the responsible programmer to release them. The latter is recommended when it is determined that the backup is no longer needed.

2.11 COPYFILE is the cataloged procedure name for copying single files, either a generation dataset or a standard dataset. To execute the procedure, one must supply the following JCL with the indicated symbolic parameters:

```
//COPYFILE  JOB (M,522001,CON),lcddddpsi/pgmrname,REGION=196K RIN=xx
//   EXEC  COPYFILE,DSN1=´dataset name´,BLKNEW=blksize,RET=retpd,
           GDG=´model dscb´,DSN2=´dataset name(n)´
```

```
where:       DSN1=the dataset to be copied
             BLKNEW=the new output blocksize
             RET=the retention period of the new 6250 BPI data
             GDG=where the model DSCB is located
             DSN2=output dataset name (used only when addressing
                  generation data groups where n=+1,etc.)

default:

             DSN1=
             DSN2=*.COPY.SYSUT1(the input dataset name)
             BLKNEW=
             RET=5
             GDG
```

example:

```
a)  //S1  EXEC  COPYFILE,DSN1='PRD4.DPM.TEST6250',
     BLKNEW=32000,RET=15

b)  //S1  EXEC COPYFILE,DSN1='PRDG4.DPM.TRM001(0)',
     DSN2='PRDG4.DPM.TRM001(+1)',RET=25,GDG='SYS2.DSCB'
```

Note: Users who require that a file be a specific generation number should copy the generation as a regular DSN specifying the full DSNAME including the G000V00 and no model DSCB reference.

```
ex.:  //S1  EXEC  COPYFILE,DSN1='PRDG4.DPM.TRM001.G0070V00',RET=45
```

2.12 COPYTAPE is the cataloged procedure name for copying tapes with more than one dataset residing on it. To execute this procedure, one must supply the following JCL with the indicated symbolic parameters for each dataset on he tape:

```
//COPYTAPE JOB (M,522001,CON),1cddddpsi/pgmrname,REGION=196K BIN=xx
//  EXEC  COPYTAPE,DSNREF='first.data.set',DSNN='data.set.to.copy',
//          BLKNEW=blksize,RET=retpd,GDG='model.dscb',FILENO=labelno,
//          DSNP='data.set.name.out.'
```

where: DSNREF=the first dataset on the volume to be used in the
              VOL=REF parameter (do not code on the step
              copying the first dataset)

```
DSNN   =the dataset to be copied in this step
BLKNEW  =the new output blocksize
RET    =the retention period of the new 6250 dataset
GDG    =where the model DSCB is located
FILENO =the label number of the dataset being copied
        in this step
DSNP   =the name of the 6250 output dataset being created
   in this step
```

example:

```
//S1  EXEC COPYTAPE,DSNN='PRD4.DPM.SAFTEST1',
//         RET=45
//COPY.SYSUT2  DD  VOL=
//S2  EXEC  COPYTAPE,DSNREF='PRD4.DPM.SAFTEST1',
//    DSNN='PRD4.DPM.SAFTEST2',RET=45,FILENO=2
      .
      .
      .
```

or

```
//S1   EXEC   COPYTAPE,DSNN=´PRDG4.DPM.SAF001(0)´,
//            RET=45,GDG=´SYS2.DSCB´,DSNP=´PRDG4.DPM.SAF001(+1)´
//COPY.SYSUT2  DD  VOL=
//S2   EXEC   COPYTAPE,DSNREF=´PRDG4.DPM.SAF001(0)´,
//            DSNN=´PRDG4.DPM.SAF001(-1)´,RET=45,
//            DGD=´SYS2.DSCB´,FILENO=2,
//            DSNP=´PRDG4.DPM.SAF001´(0)´
                .
                .
                .
```

defaults:

```
            DSNREF  =
            DSNN    =
            BLKNEW= =
            RET     = 5
            GDG     =
            FILENO  = 1
            DSNP    = *.COPY.SYSUT1
```

Additions:

a) the override //COPY.SYSUT2 DD VOL= must be coded on the first dataset of the volume to satisfy JCL syntax requirements.

b) all datasets on the volume must be copied at one time to insure exact duplication

2.13 Each cataloged procedure includes a tape compare of the 1600 BPI to the 6250 BPI dataset. If the compare finds one mismatch, the step and job will ABEND with a USER12 completion code. If this happens during an execution of COPYTAPE on just one step the job should be rerun.

2.14 The following is a copy of the COPYFILE procedure:

```
//COPYFILE PROC DSN1=,BLKNEW=,REGN=196K,RET=5,GDG=,DSN2='*.COPY.SYSUT1'
//COPY      EXEC  PGM=WECOPY,REGION=&REGN
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD  DSN=&DSN1,DISP=(OLD,PASS,KEEP)
//SYSUT2    DD  DSN=&DSN2,DISP=(NEW,KEEP,DELETE),
//              UNIT=TAPE6250,DCB=(&GDG,BLKSIZE=&BLKNEW,DEN=4),
//              VOL=(,RETAIN),LABEL=RETPD=&RET
//COMPARE   EXEC  PGM=VNPUSER,REGION=&REGN,
//              PARM='COMPARE,001,6250 COPY COMPARE'
//STEPLIB   DD  DSN=PRD16.LOADLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD  DSN=*.COPY.SYSUT1,DISP=(OLD,UNCATLG,KEEP),VOL=(,RETAIN)
//SYSUT2    DD  DSN=*.COPY.SYSUT2,DISP=(OLD,CATLG,DELETE),
//              VOL=(,RETAIN)
//ABEND     EXEC  PGM=WEABEND,COND=(0,EQ,COMPARE),REGION=30K
//SYSUT1    DD  DSN=*.COPY.SYSUT2,DISP=(OLD,UNCATLG,UNCATLG)
//SYSUT2    DD  DSN=*.COPY.SYSUT1,DISP=(OLD,CATLG,CATLG)
//END       EXEC  PGM=IEFBR14
```

2.15 The following is a copy of the COPYTAPE procedure:

```
//COPYTAPE PROC DSNREF=,DSNN=,REGN=196K,BLKNEW=,RET=5,GDG=,FILENO=1,
//                  DSNP='*.COPY.SYSUT1'
//COPY      EXEC PGM=WECOPY,REGION=&REGN
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD DSN=&DSNN,DISP=(OLD,PASS,KEEP)
//SYSUT2    DD   DSN=&DSNP,DISP=(NEW,PASS,DELETE),
//               UNIT=TAPE6250,DCB=(&GDG,BLKSIZE=&BLKNEW,DEN=4),
//               VOL=(,RETAIN,,,REF=&DSNREF),
//               LABEL=(&FILENO,,,,RETPD=&RET)
//COMPARE   EXEC   PGM=VNPUSER,REGION=&REGN,
//               PARM='COMPARE,001,6250 COPY COMPARE'
//STEPLIB   DD   DSN=PRD16.LOADLIB,DISP=SHR
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   DSN=*.COPY.SYSUT1,DISP=(OLD,UNCATLG,KEEP),VOL=(,RETAIN)
//SYSUT2    DD   DSN=*.COPY.SYSUT2,DISP=(OLD,CATLG,DELETE),VOL=(,RETAIN)
//ABEND     EXEC   PGM=WEABEND,COND=(0,EQ,COMPARE),REGION=30K
//SYSUT1    DD   DSN=*.COPY.SYSUT2,DISP=(OLD,UNCATLG,UNCATLG)
//SYSUT2    DD   DSN=*.COPY.SYSUT1,DISP=(OLD,CATLG,CATLG)
//END       EXEC   PGM=IEFBR14
```

2.2 The application programmer responsible for the tapes should prepare the necessary decks for the location coordinator by December 20 if not before. The location coordinator will also need the last date that the tape is to be used before December 29 so the copying may be scheduled. Those locations and systems that will be processing December 29 and after will convert automatically. Those locations outside of the Chicago area not open December 29 through 30, 1976, will have copying done by Warrenville personnel during that period. Procedures for the location coordinators to follow to get his location cards to Warrenville will be provided. Those locations in the Chicago area will be able to come to Warrenville for copying if their location is closed.

2.3 The two MOD7 high speed 6250/1600 dual density drives (UNIT=TAPE6250) will be available for copying after December 12. The cataloged procedures (see par. 2.1) will access these two drives before December 29. At that time, these procedures will then access the newly installed MOD5 drives. Any tapes that are copied before December 29 must have at least a 30 day retention period to insure reliability of the tape scratch pool.

2.4 The location coordinators are:

Columbus PECC	- Paul Ferguson	
Columbus Works	- Fred Thompson	
Dallas	- Ron McCauley	
Indian Hill	- Joyce Malleck	
Indianapolis	- Jack Eller	
Lisle	- Don Cervenka	
Montgomery	- Dick Park	
Oklahoma City	- Ron Cox	
Omaha	- Bob Wilson	
Warrenville	- Bob Pozorski	- 9416
	- Joe Kucharski	- 9417
	- Jim Sterner	- 9428, 29

3. JCL CONSIDERATIONS

3.1 System defaults will be changed as of December 29, 1976. UNIT=TAPE will default to the MOD5 6250/1600 dual density drives and the density will default to DEN=4 for 6250 BPI tapes. Any 1600 BPI tape mounted for input will trigger 1600 BPI processing on the MOD5 drive. All this means is that the user utilizing standard system defaults will be required to make no JCL changes. Any JCL with DEN=3 coded needs to have this code removed to use the new 6250 BPI density default. Any files that must be 1600 BPI (microfiche, MINI-Wedge, etc.) must have UNIT=TAPE6250 and DEN=3 on the DD card. Anything else will not produce 1600 BPI output. The generic name of 6250 BPI volumes is 3400-5 to be used in IEHPROGM applications. This is a change from 3400-3 used currently for 1600 BPI.

3.2 Spool tapes will be limited to 1/4 tape. This is no reduction in the amount of data allowed currently.

3.3 All Tape Library tapes need to be converted by March 1, 1977. At this time the 1600 BPI dual capability will be removed from the MOD5 drives. A listing of all 1600 BPI tapes remaining will be prepared in February and these files must then be copied. However, as 80% of the tapes in our library have a retention period of 30 days or less, the number of remaining 1600 BPI tapes should be small.

4. CASECODE

4.1 WDC Users should use the following casecode

522001,CON

for computer charges and the associated ISD time charges for copying User files to 6250 BPI.

5. QUESTIONS/PROBLEMS

5.1 Questions or problems concerning the above should be directed to Sue Field or Harry King, both on CORNET 391-5446.

S. A. Field

S. A. FIELD
Data Center Hardware
and Software
Development and
Operations Research

Approved: RJ Mele
R. J. MELE - 9411

Approved: L. J. Jweid
L. J. JWEID - 9412

Approved: RR Sorensen
R. R. SORENSEN - 9415

DIVISION 4 - MISCELLANEOUS INFORMATION

CHAPTER 2 - INFORMATION SYSTEMS MEMORANDUM

SECTION 3 - SPECIAL INFORMATION

SUBJECT - CONVERSION OF THE WDC ANS COBOL VERSION 4 CATALOGUED PROCEDURES FOR VS2

1. GENERAL

1.1 The IBM ANS COBOL COMPILERS are designed to use main core via control parameters supplied to the COMPILER on the PARM field of the execute card or by defining parameters at system generation time. The compiler does not use the REGION parameter on the EXECUTE or JOB card to determine how much core is available for the COMPILER to use for execution. The WDC ANS COBOL V4 catalogued procedures were designed for OS/MVT where core is allocated in multiple of 2K. Since VS2 core is allocated in multiples of 64K, the compiler when run under VS2 has core allocated to it which it is not using.

1.2 By merely modifying the SIZE and BUFSIZE parameters passed to the COBOL V4 COMPILER so that all allocated core is used by the compiler our studies show that the elapsed time for compiles may be reduced on the average 25%. The studies to determine the optimum region and values for the SIZE and BUFSIZE parameters were done by R. L. Wilson at Omaha and R. A. Burns at Warrenville.

1.3 These WDC ANS COBOL Version 4 Catalogued procedures will be modified as of January 12, 1976. All COBOL users should review any job which uses these catalogue procedures to ascertain that the REGION on the JOB card and the REGION on the EXECUTE card are the same to prevent any unnecessary job failures.

2. COBOL CATALOGUED PROCEDURES

2.1 Following is a listing of the modified catalogued procedures.

2.11 COB4C

```
//COB4C PROC LINES=55
//COB EXEC PGM=IKFCBL00,REGION=128K,
// PARM='NOLOAD,NODECK,LINECNT=&LINES,BUFSIZE=16K,SIZE=120K'
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(9,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(6,5))
```

2.12 COB4CG

```
//COB4CG PROC LDREG=84K,LINES=55
//COB EXEC PGM=IKFCBL00,REGION=128K,
// PARM='LINECNT=&LINES,BUFSIZE=16K,SIZE=120K'
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR
//SYSPRINT  DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)
//SYSUT1   DD   UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT2   DD   UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(9,5))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(6,5))
//SYSLIN   DD   DSNAME=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)    1/71
//GO   EXEC  PGM=LOADER,PARM='MAP,LET,SIZE=&LDREG',REGION=&LDREG,
//     COND=(5,LT,COB),TIME=10
//SYSLIN   DD   DSN=&&LOADSET,DISP=(OLD,DELETE)
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR
//SYSLOUT   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=121)
//SYSUDUMP DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)   1/71
//SYSOUT   DD   SYSOUT=A,DCB=(RECFM=FBA.LRECL=120,BLKSIZE=120)
```

2.13 COB4CL

```
//COB4CL PROC LINES=55
//COB EXEC PGM=IKFCBL00,REGION=128K,
// PARM='LINECNT=&LINES,BUF=16K,SIZE=120K'
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR
//SYSPRINT  DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)
//SYSUT1   DD   UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT2   DD   UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(9,5))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(6,5))
//SYSLIN   DD   DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)    1/71
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST,LET',COND=(5,LT,COB),
//               REGION=128K
//SYSLIN   DD   DSNAME=&LOADSET,DISP=(OLD,DELETE)
//         DD   DDNAME=SYSIN
//SYSLMOD   DD  DSN=&GODATA(RUN),DISP=(NEW,PASS),
//              UNIT=SYSDA,SPACE=(TRK,(9,,1))
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(TRK,(14,5))
//SYSPRINT  DD   SYSOUT=A
```

2.14 COB4CLG

```
//COB4CLG PROC LINES=55
//COB        EXEC PGM=IKFCBL00,PARM='LINECNT=&LINES',REGION=128K
//STEPLIB DD DSN=SYS2.COBOL4.LINK,DISP=SHR
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1089)
//SYSUDUMP DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT2 DD UNIT=SYSDA,SPACE=(TRK,(12,5))
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(9,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(6,5))
//SYSLIN DD DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//  SPACE=(400,(50,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//LKED EXEC PGM=ATTIEWL,PARM='XREF,LIST,LET',COND=(5,LT,COB),
//              REGION=128K
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//SYSLMOD DD DSN=&GODATA(RUN),DISP=(NEW,PASS),UNIT=SYSDA,
//  SPACE=(TRK,(9,,1))
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(TRK,(14,5))
//SYSPRINT DD SYSOUT=A,SPACE=(TRK,(5,5))
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,COB),(5,LT,LKED)),TIME=10
//SYSUDUMP DD SYSOUT=A
//SYSOUT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=120)
```

2.15 COB4COL

```
//COB4COL PROC LINES=55
//COB        EXEC PGM=CPXUPTSM,REGION=128K,
//              PARM='NODECK,NOMAP,LINECNT=&LINES'
//STEPLIB DD DSN=SYS2.ANSLIB,DISP=SHR
//          DD DSN=SYS2.COBOL4.LINK,DISP=SHR
//SYSIN1 DD    DSNAME=&&SYSIN1,UNIT=SYSDA,SPACE=(605,(4640,580))
//SYSIN2 DD    DSNAME=&&SYSIN2,UNIT=SYSDA,SPACE=(400,(240,80))
//STATSDD DD DSN=OPTSTATU,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=P,DCB=(RECFM=F,BLKSIZE=80)
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(460,(6000,1500))
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT4 DD DSN=&SYSUT4,UNIT=SYSDA,SPACE=(460,(700,100))
//SYSLIN DD DSN=&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),
// SPACE=(400,(240,80)),DCB=BLKSIZE=400
//LKED EXEC PGM=ATTIEWL,REGION=128K,COND=(5,LT,COB),
//        PARM='XREF,LIST,LET'
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//        DD DDNAME=SYSIN
//SYSLMOD DD DSN=&GOSET(LOADMOD),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(9,,1))
//SYSLIB DD DSN=SYS2.COBOL4.LIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(14,5))
//SYSPRINT DD SYSOUT=A
```

3. QUESTIONS/PROBLEMS

3.1 Any problems or questions regarding these changes should be directed to D. J. Sloan on CORNET 391-5256 or H. L. King on CORNET 391-5446.

H. L. King

H. L. KING
Divisional Planning,
Data Base Administration,
Standards and Control

Approved: R. J. Bonelli
R. J. BONELLI - 9412

PRELIMINARY COPY

STANDARDS FOR USING THE DATA PROCESSING SYSTEM

NAMING STANDARDS

SECTION 3 - DATA SET NAME STANDARDS

1. PRODUCTION DATA SET NAME STANDARDS

1.1 All production data sets which are not temporary data sets shall have a DSNAME consisting of at least three levels.

1.11 Level 1 of the DSNAME must be one of the following:

PRDn.	for production data sets
PRDGn.	for production generation data sets

where n is the assigned PGN (Production Group Number) for the application creating the data set (see Division 2, Chapter 1, Section 3 for assignment of a PGN).

1.12 Level 2 of the DSNAME must be the assigned PSI (Project or System Identifier) for the application creating the data set. (see Division 2, Chapter 1, Section 3 for assignment of PSI).

1.13 All other levels used in a production DSNAME are chosen by the application system developer as indicated by "x" in the following examples where "COS" is the system PSI and 7 is the PGN:

```
DSNAME=PRD7.COS.xxxxxxxx
DSNAME=PRDG7.COS.xxxxx.xxxxxxxx(+1)
DSNAME=PRD7.COS.xxx.xxxxx.xxxxxxxx
```

2. NON-PRODUCTION DATA SET NAME STANDARDS

2.1 All non-production data sets (excluding temporary data sets) which are either tape data sets which will be catalogued, or are disk data sets must have a DSNAME of at least three levels.

2.11 Level 1 of the non-production DSNAME shall be of the form DSNAME=DEV for data sets created by the developers at the central site, or DSNAME=DEVnn for those created by developers at remote locations, where nn represents station number assigned to the remote locations (see Division 2, Chapter 1, Section 2, Appendix A).

2.12 Level 2 of the non-production DSNAME shall be the PSI of the application creating the data set.

PRELIMINARY COPY

2.13 All other levels of the DSNAME are at the option of the developer. For example:

```
DSNAME=DEV.DPM.xxxxxxxx
DSNAME=DEV06.XMM.xxx.xxxxx
```

3. CONFORMANCE TO NAMING STANDARDS

3.1 All development or extensions of current systems are expected to observe naming standards.

3.2 Systems now in production are not required to change data set names which conform to the previous standard, however when systems are modified, the standards should be observed.

DIVISION 4 - MISCELLANEOUS INFORMATION

CHAPTER 2 - INFORMATION SYSTEMS MEMORANDUM

SECTION 3 - SPECIAL INFORMATION

SUBJECT - CONVERSION FROM OS MVT RELEASE 21.7 TO OS/VS2 RELEASE 1.7

1. TESTING OF PRODUCTION JOBS UNDER VS2

1.1 Central Operations has noticed that very few production jobs are being processed under VS2. It is imperative that all production jobs be made operative under VS2 via either a parallel test or an actual production run prior to the 370/158 (SY3)'s conversion to VS2 on October 20, 1975. Failure to do so could result in a huge production backlog on this date. The large majority of production jobs should incur no problem under VS2; however, based on the experience of development jobs run under VS2, it is apparent that there are many programs performing quasi-legal or illegal operations during execution which were never detected under MVT and which will almost certainly fail under VS2.

1.2 To facilitate testing production jobs under VS2, central operations will bring up VS2 on the 370/168 (SY2) at midnight WDC time as of Sunday September 21, 1975. VS2 will be available for testing from 00:01 to 17:00 WDC time Monday through Friday.

1.3 A modification has been made to the operating system to allow users to selectively run production jobs via a //*MAIN TYPE=ANY control card under VS2 if VS2 is available and under OS/MVT if VS2 is not available. Attached is a copy of "Production Jobs on VS2" which appeared in ASP NEWS and which further explains this modification.

2. CONVERSION OF CPU'S TO VS2

2.1 On October 20, 1975 the 370/158 (SY3) will be converted to VS2 release 1.7. At this time the two real CPU's (the CPU's ASP is not resident on) will be run under VS2 twenty-four hours a day. The local ASP processor will continue to run OS/MVT release 21.7. Any user who feels that a particular job must run under OS/MVT has to have the approval of Department 9411 prior to submitting a job to run only on the local CPU. Any job which does not have prior approval of Department 9411 is subject to cancellation by Central Operations with no prior notice.

3. PROBLEMS/QUESTIONS

3.1 Any problems or questions concerning this matter should be directed to H. L. King, Department 9412 on 8-391-5245 or R. J. Shitter, Department 9411 on 8-391-5455.

H. L. King
H. L. KING - 9412

cv

Att.

APPROVED R. S. Lampert
R. S. LAMPERT - 9411

L. J. SWEID - 9412

PRODUCTION JOBS ON VS2

EFFECTIVE MONDAY AUGUST 25, THERE WILL BE A MODIFICATION IN ASP TO PERMIT SELECTED PRODUCTION JOBS TO RUN ON VS2. IT IS ESSENTIAL THAT WE TRY AS MANY PRODUCTION SYSTEMS AS POSSIBLE AT LEAST ONCE BEFORE PROCEEDING ANY FURTHER WITH THE CUTOVER. BASED ON THE EXPERIENCE WITH DEVELOPMENT JOBS ON VS2, IT IS APPARENT THAT THERE ARE MANY PROGRAMS DOING QUASI-LEGAL OR OUTRIGHT ILLEGAL THINGS DURING EXECUTION THAT WERE NEVER DETECTED ON MVT. THESE ARE ALMOST CERTAIN TO FAIL ON VS2.

WE WOULD LIKE ALL SUBMITTERS OF PRODUCTION JOBS TO CONSIDER RUNNING THEM ON VS2. SINCE VS2 IS NOT NOW AS STABLE AS MVT, IT WOULD BE WISE INITIALLY TO SELECT JOBS WITH EASY RERUN CAPABILITIES, ETC. AS THE SYSTEM BECOMES MORE AND MORE STABLE, SOME MORE CRITICAL PRODUCTIONS JOBS COULD BE RUN. THE INITIAL PRODUCTION JOBS RUN ON VS2 ARE EXPECTED TO EXPERIENCE EXTREMELY QUICK TURNAROUND, BUT THE TURNAROUND FOR OTHER PRODUCTION WILL ALSO IMPROVE AS THE LOAD ON THE FIRST SHIFT PRODUCTION CPU'S IS REDUCED.

TO INDICATE A PRODUCTION JOB'S ELIGIBILITY FOR VS2, INSERT

```
//*MAIN TYPE=ANY
```

IN THE RUN DECK. NORMALLY, TYPE=ANY INDICATES THAT THE JOB CAN RUN UNDER ANY CONTROL PROGRAM, AND IS THE ASP DEFAULT. WARRENVILLE OPERATIONS KEEPS PRODUCTION OFF VS2 BY TURNING OFF THE SCHEDULING FOR CLASS P (PRODUCTION) ON THE VS2 MAIN. WITH THE MODIFICATION WE ARE INSTALLING, AN EXPLICIT REQUEST FOR TYPE=ANY WILL RESULT IN THE JOB'S BEING PUT IN A SPECIAL SCHEDULING CLASS (=ANY) WHILE STILL RETAINING ITS OTHER ATTRIBUTES AS A PRODUCTION JOB. THESE JOBS MAY THEN RUN ON ANY SYSTEM. TO FORCE A JOB TO RUN ONLY ON VS2, CODE TYPE=VS2. NOTE THAT IF VS2 IS NOT UP, A JOB WITH TYPE=VS2 WILL NOT RUN AT ALL (UNTIL VS2 IS NEXT IPL'D).

IMPORTANT NOTES:

1) THIS FEATURE IS INCOMPATIBLE WITH THE USE OF THE //*MAIN CLASS= PARAMETER. IF A //*MAIN CARD WITH THE CLASS= PARAMETER IS PRESENT, THE TYPE=ANY FEATURE WILL BE IGNORED, AND THE NORMAL SCHEDULING FOR THE SPECIFIED CLASS WILL APPLY (WHICH MEANS THE JOB WILL MOST LIKELY NOT GET TO A VS2 MAIN).
2) THIS METHOD IS SLIGHTLY DIFFERENT FROM THE METHOD UNOFFICIALLY ANNOUNCED PREVIOUSLY. THE PARAMETER IS NOW TYPE=ANY, NOT SYSTEM=ANY AS PREVIOUSLY PLANNED.
3) THE SAME RESULT MAY BE OBTAINED BY USING //*MAIN CLASS=ANY, BUT THIS HAS TWO DISADVANTAGES:
   A) THE CLASS=ANY MAY OVERRIDE ANOTHER CLASS SPECIFICATION (SUCH AS CLASS=XXWEDGE) DEPENDING ON THE ORDER OF THE //*MAIN CARDS. TYPE=ANY WILL NOT OVERRIDE, AS EXPLAINED IN NOTE (1).
   B) AN ADDITIONAL MAIN SCHEDULING CLASS ADDS OVERHEAD TO THE SYSTEM, SO WE INTEND TO REMOVE THE "ANY" CLASS AS SOON AS VS2 IS CUT OVER. AT THAT TIME, ALL CLASS=ANY CARDS MUST BE REMOVED OR THE JOB WILL FAIL IN INPUT SERVICE. WITH TYPE=ANY, ALL WE NEED DO IS REMOVE THE MODIFIED CODE, AND THE TYPE=ANY WILL BE EFFECTIVELY A NO-OP.

STANDARDS FOR USING THE DATA PROCESSING SYSTEM

SECONDARY STORAGE

SECTION 6 - THE TAPE LIBRARY SYSTEM

1. GENERAL

1.1 Purpose

1.11 The Tape Library System is used to control Standard Label tape assignments to individual programmers and production systems, to maintain enforced write protection for assigned tapes, and to release for scratch those assigned tapes as specified by the owner via methods of retention period or cyclic tape retention. Tapes with no standard labels or with non-standard labels are outside the scope of this system.

1.12 The Tape Library System was originally created at the Hawthorne Data Center to enable users to keep track of their magnetic tapes when access to the physical tape reel was eliminated.

1.121 A batch run is made daily to maintain a set of up-to-date records on those tapes currently assigned to each user listing the specifications of those tapes (i.e. density, labels, recording format, volume serial, etc.), what files reside on each tape, and when the retention period of each tape expires.

1.122 The user interface with the Tape Library System batch run is designed to reduce to a minimum the effort required for this record keeping.

1.123 At the time the Tape Library System batch run was designed, the operating organization was still using external labels for all tapes in use.

1.13 When the move of data processing operations from Hawthorne to the Warrenville Data Center was being planned, it became apparent that it would no longer be practical for the operations staff of the computer room to be solely responsible for maintaining all the Standard Label tapes due to the increasing number of tapes to be handled. The Tape Library System was therefore expanded for the Warrenville Data Center to include on-line facilities which would automate some of the functions previously performed manually:

1.131 The on-line Tape Library System automatically assures proper retention of production and non-production output tape reels.

1.132 The on-line Tape Library System automatically releases for reuse (scratch) those tapes which have been retained past the release date designated when the tape was created or last renewed.

1.133 Virtually all standard label tapes are controlled by the on-line Tape Library System. Both production and non-production jobs must interface with the system to use standard label tapes.

1.134 The on-line Tape Library System automatically releases for reuse those tapes for which Cyclic Tape Retention has been specified and which, according to owner specifications, are no longer required.

1.14 The Tape Library System, both on-line and batch modes, allow assignment of tapes only to registered data processing users (see Div. 2, Chap. 1 for registration methods) as indicated by the user-ID in the name field of each JOB card. User-ID is defined in Div. 2, Chap. 4, Sect. 4 of this manual.

2. TAPE LIBRARY SYSTEM - BATCH MODE FACILITY

2.1 The Tape Library batch run is made daily at the central site (WDC) during the second shift.

2.11 Input to the daily batch run is from several sources:

2.111 Input request cards (as described below) prepared by programmers at the central site are collected at the programmer service counter in the computer room.

2.112 Input request cards originated by programmers at remote sites are collected and are submitted as SYSIN data in the TLSDAYxx (where "xx" is the remote site terminal line number) job. This job must be run once (and only once) each day Monday through Friday at 16:00 (4:00 P.M.) Warrenville time, even if no request cards are submitted.

2.2 Preparing Request Card Input

2.21 There are three types of action which are performed for users by the batch run:

2.211 A user may acquire a tape for his own use for a specific period of time. This is not recommended as a method of acquiring a scratch tape. Sucb tapes should be acquired as needed by the on-line Tape Library System (see Div. 2, Chap. 6, Sect. 6, paragraph 3.4) during JOB execution.

2.212 A user may renew (extend the specified period of of time) of a tape previously acquired either via the batch run (2.11 above) or via the on-line Tape Library System facility (paragraph 3 of this section).

2.213 A user may release a previously acquired tape prior to the expiration of the specified time period.

2.22 These actions are accomplished by the user's submission of punched request cards containing one of three permissable request codes with appropriate operands. Each field of the request card must be separated by at least one blank column:

2.221 ACQUIRE request cards are submitted to obtain the exclusive use of a standard label magnetic tape and must contain the following fields:

keyword 'ACQUIRE'
* release date (MM/DD/YY or M/D/YY)
user-ID (see Div. 2, Chap. 4, Sect. 4)

* Optional, if omitted maximum development retention period is assumed. (see paragraph 2.231 below)

2.222 RENEW request cards are submitted to change the release date of a tape previously acquired and must contain the following fields:

keyword 'RENEW'
volume serial number (must be six digits)
* new release date (MM/DD/YY or M/D/YY)
user-ID (see Div. 2, Chap. 4, Sect. 4)

* Optional, if omitted maximum development retention period is assumed. (see paragraph 2.231 below)

2.223 RELEASE request cards are submitted to release, for use as scratch, a previously acquired tape prior to the expiration of the current release date recorded for the tape by the Tape Library System, and must contain the following fields:

keyword 'RELEASE'
volume serial number (must be 6 digits)
user-ID (see Div. 2, Chap. 4, Sect. 4)

If a RENEW request card is not submitted prior to the date recorded for release by the Tape Library System, the tape will automatically be released for use as a scratch tape during the batch run of that date or the next working day thereafter.

Releasing a tape from the Tape Library System does not uncatalog it. When a tape is released, its owner should also uncatalog that tape to maintain consistency between the catalog and the Tape Library System.

2.23 Retention period restrictions apply to both RENEW and ACQUIRE request cards release dates. These restrictions are set by the Operating Organization depending on the inventory of Standard Label tapes available for general use. The current retention period restrictions are shown below but are subject to change at the descretion of the Operating Organization with advance notice to all users.

2.231 Maximum Retention Period	Tape Use for:
2 years (on-line only)	Production
60 days (on-line only)	Maintenance
30 days (on-line and all batch)	Development

2.24 Request Card Examples

```
ACQUIRE WN3411SMITH
RENEW 703205 7/2/73 WN3411SMITH
RENEW 703205 UNTIL 07/02/73 FOR WN3411SMITH
RELEASE 703205 WN3411SMITH
```

2.3 User Output From Tape Library Batch Run

2.31 The response to a request card submitted by a user to the Tape Library batch run will be in one of the following forms and will be available (in the BIN number registered on the bin table for the user-ID punched in the input card) the first working day after the day the request was submitted.

2.311 Printed Error Message - If the syntax of the request card is found in error or if for any other reason the request cannot be serviced, error messages will be routed to the registered BIN number for the user-ID punched in the input request card. Messages for incorrect user-IDs are posted or the person is contacted if necessary.

2.312 Printed Verification of Action Taken - The response to a RELEASE request is a message routed to the user's BIN number.

2.313 Punched RENEW card - The response to a successful ACQUIRE or RENEW request (also for any non-production tape acquired during JOB execution by the on-line Tape Library System) is a punched and interpreted RENEW card. Example:

```
RENEW 703205 UNTIL 07/02/73 FOR WN3411SMITH    BIN-139
```

This punched card can be used for reference and also by duplication with a new release date, can be used for creating a RENEW request input card.

2.32 The Output Received Weekly.

2.321 Each week users having active acquired tapes within the Tape Library System will receive a "TAPE RETENTION LIST" report in their respective bins. This output lists all pertinent information for each tape assigned to the user. Included are:

volume serial number	JOBNAME creating tape
date created	DSNAME (44 characters)
release date	users BIN number

2.322 Each user is encouraged to submit RELEASE request cards prior to the recorded release date whenever the assigned tape is no longer required.

3. TAPE LIBRARY SYSTEM - ON-LINE MODE FACILITY

3.1 The Tape Library System on-line facility is in execution whenever the Data Processing System is in operation. Its purpose is to provide for on-line tape assignments to users or production systems of output tapes created during a JOB execution. This assignment is recorded within the Tape Library System just as those tapes assigned by the Tape Library batch run upon the receival of an ACQUIRE request card.

3.2 The basic features of the on-line Tape Library System are:

3.21 Production system jobs and non-production jobs can acquire tapes while executing a job. Only unassigned tapes will be accepted for a scratch tape (JCL DD statement has no VOL=SER= parameter,DISP parameter is NEW).

3.22 Assigned tapes within the Tape Library System are write protected. No user (user-ID in the JOB card) may write on a tape which is assigned in the Tape Library System to some other user (a different user-ID). This eliminates the accidental overwriting of data. An attempt to violate this feature will cause the mounted tape to be unloaded and a scratch tape substituted.

3.221 This write protection feature will not prevent a user's non-production JOB from overwriting his own data on a tape reel assigned to him, when the DD statement specifically requests the tape by specifying the volume serial number or by reference to the catalogue. Overwriting such a tape will establish a new release date for the tape serial number as indicated in the JCL DD statement. This allows the user to continue to use the same tape volume serial number any number of times.

3.222 If, during the execution of a production JOB, an attempt is made to overwrite data on a tape previously assigned to the same production system identifier (user-ID) by the Tape Library System, a message to the main console will be generated, requesting a reply from the operator:

```
LIB070 TAPE nnnnnn ASSIGNED TO xxxxxxxx .... x
```

The operator's reply, based upon information contained in the Operating Instruction for the production system as prepared by the applications programmer, can be "U" to allow overwriting the tape or "M" which will cause dismounting of the tape and a request for a mount of a scratch tape in its place.

3.23 Tape used within a JOB and not previously acquired nor assigned during the execution of the JOB by the on-line Tape Library System are considered work tapes and are immediately returned to the scratch pool after JOB completion.

3.24 The Tape Library System determines what tapes are scratch tapes (retention period has expired). A list of scratch tape volume serial numbers is available to the operating organization at any time. Such a list is used by the tape vault librarian to select scratch tapes for use by the Data Processing System.

3.25 Standard Label tapes stored in the WDC Computer Room tape vault have no external labels other than the permanent volume serial number label affixed to the tape reel when it was purchased.

3.26 No standard label tapes under the control of the Tape Library System shall be removed from the Computer Room. Any JOB which creates a data set which must be physically transported on tape out of the WDC Computer Room must adhere to Div. 2, Chap. 6, Sec. 7: Creating Tapes for Disposition Outside Warrenville Data Center.

3.27 The required tapes for all JOBs must be specified via JCL DD statements and ASP control cards either by use of specific volume serial number or by reference to a catalogued data set.

3.28 Whenever a Tape Library System tape is used in a STEP and the third DISP parameter is explicitly specified as DELETE and the STEP fails, then that data set will be uncataloged and its expiration date will be changed to the current date plus one; provided the USERID and data set name are the same as when the tape was originally created.

3.3 To ACQUIRE or RENEW a tape on-line, the JOB card name field must contain a registered user-ID.

3.31 Registration procedures are detailed in Div. 2, Chap. 1.

3.32 User-ID format is defined in Div. 2, Chap. 4, Sec. 4.

3.33 An attempt to assign a tape on-line to an unregistered user-ID will cause the job to be automatically cancelled (ABEND 222).

3.4 To ACQUIRE a tape via the on-line Tape Library System, all of the following conditions must be true. A successful ACQUIRE during a non-production job will cause the generation of a RENEW request card by the next Tape Library batch run (see paragraph 2.313 of this section).

3.41 The tape to be assigned must be a scratch tape.

3.42 The tape file must be opened for OUTPUT

3.421 Tapes are assigned by the on-line Tape Library System during OPEN processing for output tape files and during volume switching for multi-reel output files.

3.43 EXPDT or RETPD must be specified in the LABEL parameter.

3.431 The length of time the tape is to remain assigned must be specified. Production systems should use the RETPD parameter, which allows retention period to be stated in length of time from the current date. EXPDT should be used only when it is desirable to have a tape released on a specific date. The normal OS EXPDT and RETPD have been disabled since the protection afforded by the Tape Library System is more complete.

3.4311 Specifying RETPD-0 assigns the tape for only the duration of the run day and is automatically released during the next Tape Library batch run.

3.5 To RENEW a previously assigned tape via the on-line Tape Library System, all of the following conditions must be true. A successful RENEW during a non-production job will cause the generation of a RENEW request card by the next Tape Library batch run (see paragraph 2.313 of this section).

3.51 The tape to be renewed must be already assigned to the user-ID in the JOB card.

3.511 The tape must have been previously acquired by either the batch run or on-line facility of the Tape Library System. The user or production system will overwrite their own previously recorded data on the assigned tape.

3.52 The actual volume serial number of the assigned tape must be specified (by JCL or catalog reference).

3.521 The DD statement must not request a scratch tape.

3.53 The tape file must be opened for OUTPUT.

3.531 Tape renewal by the on-line Tape Library System occurs during OPEN processing for output tape files.

3.54 EXPDT or RETPD must be specified in the LABEL parameter.

3.541 These two parameters are further explained in paragraph 3.43 of this section.

3.55 The DISP parameter must not be MOD

3.551 The Tape Library System will not change (RENEW) the release date for any assigned tape when DISP=MOD is used.

3.56 The data set sequence number subparameters of the LABEL parameter must be "1" or omitted.

3.561 The Tape Library System will not change (RENEW) the release date of an assigned tape if the data set is not the first one on the tape.

3.6 Specifying Cyclic Tape Retention. This method is to be used whenever a user needs to specify the number of generation levels of a Generation Data Group which are to be retained in the Tape Library System in his name.

3.61 In order to build a Generation Data Group, an index must be created via the IMB utility IEHPROGM (See IBM System/360 Operating System:UTILITIES). Cyclic Tape Retention will be indicated whenever the BLDG statement contains the DELETE parameter when the index is created.

3.62 If Cyclic Tape Retention is indicated, tapes which are removed from the index are not only uncataloged, but also the retention date on that tape is changed to the current date plus one in the Tape Library System.

3.63 All statements in Div. 2, Chap. 6, Sec. 6, regarding acquiring, renewing, and releasing tapes apply to tapes which are controlled by Cyclic Tape Retention. Therefore, when a tape's expiration date is reached, it will be released to the scratch pool. With this in mind, a generous retention period should be specified whenever Cyclic Tape Retention is used.

3.64 In releasing tapes as they are deleted from the Generation Data Group index, there must be a match between the user ID of the owner of the tape and the user ID of the JOB during which deletion is to take place. This implies that Generation Data Sets which have members generated by more than one user ID should avoid using Cyclic Tape Retention.

4. TAPE LIBRARY SYSTEM - UTILITY PROGRAMS

4.1 The TLSBINL utility program produces a formatted listing of the contents of the bin table by organization number. A parameter may be specified to limit the list to one location or to one organization within a location.

4.2 Possible formats for a user supplied parameter are:

PARM=LL where LL is an alphabetic location code. The bin table list produced will include only User-ID's registered for this location.

PARM=LLDDDD where LL is an alphabetic location code and DDDD is a department number (numeric). The bin table list produced will include only User-ID's registered for this organization.

If no parameter is specified, the entire bin table will be listed.

4.3 Execution of TLSBINL should be accomplished via the cataloged procedure TLSBINL which is documented in Division 3, Chapter 5, Cataloged Procedures.

CATAUTIL - CATALOG MAINTENANCE UTILITY

1. GENERAL

1.1 CATAUTIL is a Western Electric utility whose purpose is to perform functions on the catalog that are not available through IEHPROGM.

1.2 CATAUTIL incorporates the following facilities:

1.21 Delete an index and all subordinate indexes and uncatalog any data sets that are subordinate to these indexes.

1.22 Change generation data group index specification without previously uncataloging all data sets and subsequently re-cataloging all data sets.

1.3 The return code will be the highest one encountered in the execution of the program.

2. CONTROL CARDS

2.1 The control card for deleting an index is as follows:

```
label   DLTINDEX   INDEX=na,e ,CVOL=serial
```

where:

2.11 INDEX = name

Specifies the index level that is to be deleted and this includes deleting of all subordinate indexes and uncataloging all subordinate data sets.

2.12 CVOL = serial

Specifies the volume serial number of the control volume on which the search for the index is to begin. If CVOL is omitted the index should either reside on the sytem residence volume or on a volume that is appropriately connected to by the first level of the index.

2.2 The control card for changing a Generation Data Group index specification is as follows:

```
label    CHNGBLDG  INDEX=name ,ENTRIES=n
                    ,OPTION=optional  ,CVOL=serial
```

where:

2.21 INDEX = name

Specifies the name of the Generation Data Group index that is to be changed.

2.22 ENTRIES=n

Specifies the new number of entries for the Generation Data Group index which must not exceed 255.

2.23 OPTION=option

Specifies one character option selection as follows:

<u>OPTION</u>	<u>MEANING</u>
L	Leave option as previously specified.
E	Empty option as stated for BLDG statement in the IBM utilities manual.
D	Delete option as stated for BLDG statement in the IMB utilities manual.
B	Both Empty and Delete
N	Neither Empty or Delete just uncatalog

2.24 CVOL=serial

Specifies same as in paragraph 2.12 above.

3. JCL STATEMENTS

3.1 In general the following JCL is required

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD * control cards follow
```

3.2 If the control volume is not a permanently resident volume then a DD statement should be placed in the JCL to insure the volume is mounted.

3.3 The basic module takes 24K to execute and requires additional 2K for each 30 data sets cataloged in a Generation Data Group that is being changed with a CHNGBLDG control card.

4. EXAMPLES OF USE

4.1 Example 1: There exists two Generation Data Groups below the NODE A.B both of which are no longer needed.

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DLTINDEX INDEX=A.B
```

4.2 Example 2: The number of entries for Generation Data Group index A.B.C is to be changed from 15 to 35 the option is to be left as is.

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  CHNGBLDG INDEX=A.B.C,ENTRIES=35
```

4.3 Example 3: The option of delete is to be removed for Generation Data Group index A.B.D on control volume 222222

```
// EXEC PGM=CATAUTIL
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  CHNGBLDG INDEX=A.B.D,OPTION=N,
  CVOL=222222
```

5. REFERENCE

5.1 Description of IEHPROGM in I.B.M. Operation System Utilities Manual (GC28-6586)

5.2 Return codes from catalog operations are described in F19.CTLG2 IBM System/360 Operating System (GC28-6550) Data Management for System Programmers.

5.3 Cataloged procedure CATAUTIL in Division 3, Chapter 5, Section 1.

5.4 Questions may be directed to J. R. Jackson, 11CB059430.

CATAUTIL - CATALOG MAINTENANCE UTILITY

1. GENERAL

CATAUTIL is a catalogued procedure which facilitates use of Western Electric utility program CATAUTIL. It employs a symbolic parameter for the purpose of designating the control volume.

2. SYMBOLIC PARAMETERS

CVOL 6 digit volume serial number of the control volume. Required only when control volume is not permanently resident. Default value is PROD06.

3. USING CATAUTIL

3.1 The procedure is invoked with an EXEC card with the optional CVOL parameter

```
//   EXEC   CATAUTIL ,CVOL=cvol
```

3.2 The cvol parameter is only required if the control volume is not permanently resident

3.3 Control cards are required as described in Division 3, Chapter 2, Section 2, CATAUTIL

4. EXAMPLES OF USE:

4.1 Delete index from TEST05

```
//   EXEC   CATAUTIL,CVOL=TEST05
//SYSIN   DD   *
  DLTINDEX   INDEX=DEV05.F99
```

4.2 Change Generation Data Group index on PROD06

```
//   EXEC   CATAUTIL
//SYSIN   DD   *
  CHNGBLDG   INDEX=PRDG22.FOO.TRANSNET,OPTION=N
```

5. CATALOGUED Procedure Listing

```
//CATAUTIL   PROC   CVOL=PROD06
//CATAUTIL   EXEC   PGM=CATAUTIL,REGION=30K
//SYSPRINT   DD   SYSOUT=A
//SYSUDUMP   DD   SYSOUT=A
```

```
//RESVOL   DD   VOL=REF=SYS1.SVCLIB,DISP=SHR
//CVOL   DD   UNIT=3330,DISP=SHR,VOL=SER=&CVOL
```

6. **REFERENCE**

Division 3, Chapter 2, Section 2, CATAUTIL

CATALIST - LIST CATALOG NODE POINTS

1. GENERAL

CATALIST is a catalogued procedure which facilitates use of the Western Electric Utility Program CATALIST. It employs symbolic parameters to specify control volume and node points.

2. SYMBOLIC PARAMETER

CVOL 6 digit serial numbers of control volume. Default PROD06

NODE NODE point(s) designated to be listed

3. USING CATALIST

3.1 The procedure is invoked with an EXEC card and the required NODE parameter and optional CVOL Parameter.

```
// EXEC CATALIST,NODE='node'
```

3.2 The CVOL parameter is required if the control volume is other than the default.

3.3 IF MORE THAN ONE CVOL at a time is to be listed the PARM field should be coded as PARM=' '.

4. EXAMPLES OF USE:

4.1 List node point PRD22.F99 from catalog on PROD06

```
// EXEC CATALIST,NODE='PRD22.F99'
```

4.2 List node points DEV07.V01 and DEV05.F99 from catalog on TEST05

```
// EXEC CATALIST,CVOL=TEST05,NODE=
          'DEV07.V01,DEV05.F99'
```

5. CATALOGUED Procedure Listing

```
//CATALIST  EXEC  PGM=CATALIST,
//              PARM='&CVOL.(&NODE)'
//SYSPRINT  DD   SYSOUT=A
```

6. REFERENCE

Division 3, Chapter 2, Section 2, CATALIST

```
           DSNAME OR POINTER                      TYPE      VOLSER   FILE #   DEVICE TYPE        CODE     SPECIAL FEATURES      ALIAS OF

PRDG4.MPC.MDMINPUT.G0017V00                    DATA SET    704307       1   MAGNETIC TAPE      3400
PRDG4.MPC.MDMINPUT.G0016V00                    DATA SET    726786       1   MAGNETIC TAPE      3400
PRDG4.MPC.MDMINPUT.G0015V00                    DATA SET    700272       1   MAGNETIC TAPE      3400
PRDG4.MPC.MDMINPUT.G0014V00                    DATA SET    727172       1   MAGNETIC TAPE      3400
PRDG4.MPC.MDMINPUT.G0013V00                    DATA SET    725203       1   MAGNETIC TAPE      3400
PRDG4.MPC.MDMINPUT.G0012V00                    DATA SET    706919       1   MAGNETIC TAPE      3400
PRDG4.MPC.MDMINPUT.G0011V00                    DATA SET    701930       1   MAGNETIC TAPE      3400
PRDG4.MPC.MHSFILE.                             GENR PTR                     MAX=  3            CURRENT=  3      DELETE
PRDG4.MPC.MHSFILE.G0054V00                     DATA SET    717818       1   MAGNETIC TAPE      3400
PRDG4.MPC.MHSFILE.G0053V00                     DATA SET    722678       1   MAGNETIC TAPE      3400
PRDG4.MPC.MHSFILE.G0052V00                     DATA SET    702365       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.                            GENR PTR                     MAX= 10            CURRENT= 10      NEITHER
PRDG4.MPC.MMCSRMDT.G0031V00                    DATA SET    723680       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0030V00                    DATA SET    704829       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0029V00                    DATA SET    711496       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0028V00                    DATA SET    714709       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0027V00                    DATA SET    713037       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0026V00                    DATA SET    716786       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0025V00                    DATA SET    714059       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0024V00                    DATA SET    723496       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0023V00                    DATA SET    709702       1   MAGNETIC TAPE      3400
PRDG4.MPC.MMCSRMDT.G0022V00                    DATA SET    716086       1   MAGNETIC TAPE      3400
PRDG4.MPC.RFMVRPT.                             GENR PTR                     MAX=  3            CURRENT=  3      DELETE
PRDG4.MPC.RFMVRPT.G0003V00                     DATA SET    719537       1   MAGNETIC TAPE      3400
PRDG4.MPC.RFMVRPT.G0002V00                     DATA SET    726579       1   MAGNETIC TAPE      3400
PRDG4.MPC.RFMVRPT.G0001V00                     DATA SET    707905       1   MAGNETIC TAPE      3400
PRDG4.MPC.RHSFILE.                             GENR PTR                     MAX=  3            CURRENT=  3      DELETE
PRDG4.MPC.RHSFILE.G0091V00                     DATA SET    721013       1   MAGNETIC TAPE      3400
PRDG4.MPC.RHSFILE.G0090V00                     DATA SET    701720       1   MAGNETIC TAPE      3400
PRDG4.MPC.RHSFILE.G0089V00                     DATA SET    709157       1   MAGNETIC TAPE      3400
PRDG4.MPC.STKCSTU.                             GENR PTR                     MAX= 10            CURRENT=  0      DELETE
PRDG4.MPC.WEDGEMD.                             GENR PTR                     MAX= 20            CURRENT=  2      DELETE
PRDG4.MPC.WEDGEMD.G0006V00                     DATA SET    TRM002           DIRECT ACCESS      3330      TRK OVFLOW,RPS
PRDG4.MPC.WEDGEMD.G0005V00                     DATA SET    TRM002           DIRECT ACCESS      3330      TRK OVFLOW,RPS
PRDG4.MPC.WEDGEMW.                             GENR PTR                     MAX= 20            CURRENT=  3      DELETE
PRDG4.MPC.WEDGEMW.G0003V00                     DATA SET    TRM002           DIRECT ACCESS      3330      TRK OVFLOW,RPS
PRDG4.MPC.WEDGEMW.G0002V00                     DATA SET    TRM002           DIRECT ACCESS      3330      TRK OVFLOW,RPS
PRDG4.MPC.WEDGEMW.G0001V00                     DATA SET    TRM002           DIRECT ACCESS      3330      TRK OVFLOW,RPS
PRDG4.MTG.                                     INDEX PTR
PRDG4.MTG.COLO.                                GENR PTR                     MAX= 10            CURRENT= 10      NEITHER
PRDG4.MTG.COLO.G0360V00                        DATA SET    707975       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0359V00                        DATA SET    704160       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0358V00                        DATA SET    716084       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0357V00                        DATA SET    728557       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0356V00                        DATA SET    701373       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0355V00                        DATA SET    700392       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0354V00                        DATA SET    703286       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0353V00                        DATA SET    722067       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0352V00                        DATA SET    707867       1   MAGNETIC TAPE      3400
PRDG4.MTG.COLO.G0351V00                        DATA SET    722274       1   MAGNETIC TAPE      3400
PRDG4.MTG.COSDUL.                              GENR PTR                     MAX=  8            CURRENT=  8      NEITHER
PRDG4.MTG.COSDUL.G0600V00                      DATA SET    720701       1   MAGNETIC TAPE      3400
```

MISCELLANEOUS INFORMATION

INFORMATION SYSTEMS DEVELOPMENT TRAINING PROGRAM

SECTION 1 - GENERAL

1. OBJECTIVE

1.1 The availability of training material and policies concerning training differ throughout the Switching Division. This chapter heading has been provided to allow each resident organization to provide such training information as they deem desirable within the structure of this manual.

MISCELLANEOUS INFORMATION

INFORMATION SYSTEM LOAD RATES

SECTION 1 - HARDWARE & ISD LOAD RATES

1. USAGE MEASUREMENT

1.1 Use of the facilities of the Warrenville Data Center ASP/VS network, via central or remote input readers, is measured by quantities derived and reported using the facilities of the Operating System's System Measurement Facility (SMF) and the Quantitative Computer Management System (QCM).

1.11 VS/CPU TIME

The time, in seconds, the Central Processing Unit is dedicated to the processing of a job. The standard measure is 370/168 VS/CPU time.

1.12 I/O TIME

The time, in seconds, an I/O Channel is dedicated to the control and movement of data associated with a job.

1.13 UNIT RECORDS - LOCAL

The total number of cards read, cards punched and lines printed for a job utilizing the input/output service at the central site (Warrenville).

1.14 UNIT RECORDS - REMOTE

The total number of cards read, cards punched and lines printed for a job utilizing the teleprocessing facilities and the input/output service at a remote site. All service charges associated with this resource are determined by the remote sites.

1.2 In addition to OS/SMF and QCM data described in paragraph 1.1, three additional quantities are used to calculate a run cost for each job. These quantities are obtained by converting station log and time sheet data into SMF records.

Reason for reissue: As indicated; and former paragraph 1.13, "CORE ALLOCATION", deleted.

1.21 KEYSTROKES - GCS

The number of keystrokes required to enter data into the Data Entry System.

1.22 MAN HOURS

The time, in hours and minutes, charged by ISD personnel to a project number.

1.23 TRANSTIME

The time, in hours, minutes, and seconds, to receive/transmit a tape via Honeywell transmission facilities.

2. JOB COSTS

2.1 The usage data compiled by VS/SMF is periodically processed using the QCM programs to summarize computer usage by job, programmer, run type and project code. A run cost is calculated for each job as follows:

```
JOB RUN COST = (CPU TIME x CPU TIME LOAD RATE)
              + (I/O TIME x I/O TIME LOAD RATE)
              + (LOCAL UNIT RECORDS x LOCAL UNIT RECORD LOAD RATE)
              + (DATA ENTRY KEYSTROKES x DATA ENTRY LOAD RATE)
              + (MAN HOURS x ISD LOAD RATE)
              + (TRANSMISSION TIME x TRANSMISSION LOAD RATE)
```

2.2 The Job Run Cost reported on the Warrenville Data Center Computer Usage Report does not include a cost for UNIT RECORD - REMOTE. However, this factor should be included in estimations of job run costs since this load rate is determined and applied at each remote operation.

3. LOAD RATE TABLE

3.1 The load rates applied in the Job Run Cost formula of paragraph 2.1 and referenced in 2.2 are determined periodically by the Divisional Accounting, Results and Project Accounting Organization, Department 9413, Warrenville. The current load rates, which can be used for calculating Job Run Cost estimates are listed in Exhibit A.

Reason for reissue: CORE ALLOCATION deleted from Job Run
Cost formula in paragraph 2.1.

LOAD RATE TABLE

FOR DETERMINING JOB RUN COST

1. HARDWARE LOAD RATES.

1.1 August, 1973 thru March, 1974

CPU	- $.054/IBM 370/155 CPU Sec.
I/O	- $.0445/I/O Sec.
MBS	- $.09/IBM 370/155 MEGB Sec.
U/R - Warrenville	- $.002/Unit Record
GCS - Data Entry System	- $.00028/Keystroke
Honeywell Transmission	- $.01305/Sec.

1.2 April and May, 1974

CPU	- $.054/IBM 370/155 CPU Sec.
I/O	- $.032/I/O Sec.
MBS	- $.063/IBM 370/155 MEGB Sec.
U/R - Warrenville	- $.002/Unit Record
GCS - Data Entry System	- $.00019/Keystroke
Honeywell Transmission	- $.11/Sec.

1.3 June, 1974

CPU	- $.059/IBM 370/155 CPU Sec.
I/O	- $.035/I/O Sec.
MBS	- $.069/IBM 370/155 MEGB Sec.
U/R - Warrenville	- $.00335/Unit Record
GCS - Data Entry System	- $.00019/Keystroke
Honeywell Transmission	- $.11/Sec.

1.4 July, 1974

CPU	- $.141/IBM 370/168 CPU Sec.
I/O	- $.032/I/O Sec.
MBS	- $.063/IBM 370/155 MEGB Sec.
U/R - Warrenville	- $.002/Unit Record
GCS - Data Entry System	- $.00019/Keystroke
Honeywell Transmission	- $.11/Sec.

1.5 August, 1974 thru December 29, 1974

CPU	- $.216/IBM 370/168 CPU Sec.
I/O	- $.032/I/O Sec.
MBS	- $.089/IBM 370/168 MEGB Sec.
U/R - Warrenville	- $.002/Unit Record
GCS - Data Entry System	- $.00019/Keystroke
Honeywell Transmission	- $.11/Sec.

1.6 <u>December 30, 1974 thru December 28, 1975</u>

CPU	- $.167/IBM 370/168 CPU Sec.
I/O	- $.033/I/O Sec.
MBS	- $.067/IBM 370/168 MEGB Sec.
U/R - Warrenville	- $.0017/Unit Record
GCS - Data Entry System	- $.00042/Keystroke
Honeywell Transmission	- $.1072/Sec.

1.7 <u>Current - Effective December 29, 1975</u>

VS/CPU	- $.164/IBM 370/168 VS/CPU Sec.
I/O	- $.032/I/O Sec.
U/R - Warrenville	- $.0018/Unit Record
GCS - Key Data Entry	- $.00037/Keystroke
Honeywell Transmission	- $.1351/Sec.

1.8

<u>Unit Record - Remote</u> (Per Unit Record)	1975	1976
Canal Street	- $.002037	- N/A
Columbia River	- $.004172	- N/A
Columbus	- $.002767	- $.001840
Dallas	- $.003003	- $.002018
Hawthorne/Merch.	- $.002090	- $.002230
Hawthorne Works	- $.001830	- $.001293
Indian Hill	- $.001500	- $.001400
Lisle	- $.001703	- $.002646
Montgomery	- $.002988	- $.002549
Oklahoma City	- $.002837	- $.002260
Omaha	- $.003365	- $.002480

1.9 <u>On-Line System Load Rates</u>

1.91 <u>MITS - Effective January 1, 1975 - Current</u>

Terminal Connect Time	- $2.50/hour
Monthly Permanent Storage	- $.11/PSR
Peripheral Tape/Print	- $1.95/1000 lines

1.92 <u>CALL/OS - Effective December 29, 1975</u>

Connect Time	- $6.50/Connect Hour
CPU Time	- $.19/IBM 360/65 CPU Sec.
Monthly Storage	- $1.40/halftrack storage

1.93 <u>TSO - Effective December 29, 1975</u>

Connect Time	- $6.50/Connect Hour
CPU Time	- $.3095/IBM 370/168 VS/CPU sec.
I/O Time	- $.0608/I/O sec.

2. ISD LOAD RATE FOR DETERMINING JOB RUN COST.

2.1 The hourly load rate for all 9400 ISD personnel is as follows:

1973	-	$15.25
1974	-	$17.84
1975	-	$19.04
1976	-	$23.51

2.2 The Security Accrual percentage for 1976 is 28.4%.

3. MAGNETIC TAPE PROCESSING CHARGES FOR DISPOSITION OF TAPES OUTSIDE WARRENVILLE.

3.1 A charge is issued for each magnetic tape processed for disposition outside Warrenville (Division 2, Chapter 6, Section 7).

3.2 The rate per copy is as follows:

1976 - $18.80 effective 12/29/75

D. L. MATHIS 7736
N. ILL. WORKS

DISTRIBUTE TO HOLDERS OF
WARRENVILLE DATA CENTER
PROGRAMMING S & R MANUAL

IMS ON-LINE STANDARDS AND REFERENCE

IMS DC SYSTEM OVERVIEW

SECTION 1 - GENERAL

PREFACE

Division 5 of the PROGRAMMING S & R MANUAL pertains to On-line IMS Systems. The distribution of this division, except for Chapter 1, Section 1, is controlled at a locational level by a designated Location Co-ordinator. The Location Co-ordinators are listed below:

1.	CEC	- E. W. Dovan	- 13444
2.	Columbus	- D. L. Cross	- 9441
3.	Dallas	- R. M. McCauley	- 9471
4.	Indian Hill	- J. R. Malleck	- 7124
5.	Indianapolis	- J. E. Eller	- 218
6.	Lisle	- J. F. Tvrdik	- 9435
7.	Montgomery	- R. H. Patterson	- 6323
8.	Oklahoma City	- C. D. Emerson	- 9460
9.	Omaha	- F. H. Garey	- 9451
10.	Warrenville	- E. F. Nassiff	- 9412

In order to receive documentation of the entire division, the User must contact his respective Location Co-ordinator.

Communications concerning On-line IMS Systems at a Divisional level should be directed to the On-line Systems Co-ordinator, Dept. 9412, Warrenville.

1. GENERAL

1.1 The Warrenville Data Center currently supports the generalized IBM - Information Management System (IMS) for the control of application-oriented on-line systems. IMS provides two basic features:

1.11 Data Language/I (DLI) - is a data management facility that assists in the creation and maintenance of hierarchical data base structures which may be used in the on-line environment.

1.12 Data Communication (DC) - is a teleprocessing monitor that supports message communication between the application system, data bases, and a number of supported remote and/or local terminals.

1.2 This documentation pertains to those application systems which are controlled by the Data Communication facility of IMS (IMS DC). Even though strictly batch-oriented systems using DLI are not directly covered by this documentation, it would be prudent for the designer, who later intends to introduce his system to the IMS DC System, to follow the standards outlined in this document.

1.3 IMS currently runs with OS under Release 2.4.1 and is available in the following periods of time:

1.31 IMS DC Production Schedule

Monday through Friday, 7:00 a.m. to 11:00 p.m.(WDC time).

1.32 IMS DC Test Schedule

Upon request only, generally Monday through Friday 5:30 a.m. to 6:30 a.m. (WDC time). The Location Co-ordinator must make the request to the IMS Master Terminal Operator (MTO) by 4:00 p.m. (WDC time) on the preceding day.

TO ALL HOLDERS OF THE WDC PROGRAMMING S & R MANUAL

Re: Proposed Computer Assisted Restart/Recovery Standards

The standards for using the Computer Assisted Restart/Recovery System are presented here for your consideration and comment.

Your written comment should be submitted before August 23, 1976, to the Data Administration Department 9412, attention Mr. D. Cowan. A final version of the standards will be prepared after consideration of the comments received, for entry in the WDC Programming Standards and Reference Manual.

IMPORTANT: Division 5 - Data Management and Administration will replace Division 5 - IMS DC Standards; the current contents of Division 5 will be placed in Chapters within the new Division 5. This replacement is not effective immediately, and the current Division 5 should not be discarded at this point in time.

D. K. Cowan

D. K. COWAN - 9412
Divisional Data Administration

:JMI

Approved: DKC for R.J. Bonelli
R. J. BONELLI - 9412

DIVISION 5 - DATA MANAGEMENT AND ADMINISTRATION

CHAPTER 5 - DATA INTEGRITY

SECTION 2 - COMPUTER ASSISTED RESTART/RECOVERY FACILITIES

1. GENERAL

1.1 The CRR (Computer Assisted Restart/Recovery) System is designed as an augment to the IMS (Information Management System) restart/recovery facilities as they are distributed by IBM. CRR contains facilities which are both integrated into the IMS control region and executed as independent batch programs. The modifications to IMS control region code necessary to incorporate CRR are minimal and limited to only two modules (DFSFLOG0 and DFSFLOS0). CRR capabilities lie mainly in four areas:

1.2 Information Ledgering. Information on three components of an IMS system (job, database, and log tape) is stored in a number of VS datasets termed "ledgers". These datasets are updated dynamically to reflect a change in the status of a component of any functioning IMS System (this includes the on-line control region as well as batch jobs).

1.3 Status Tracking. As data based system execution progresses, component status is recorded in three ways:

LEDGER - The status field of the proper ledger entry is changed.

DISPLAY - A message identifying the component and its new status is written to the SYSMSG dataset, the ASP Master Console, and the ASP Master Log.

TRACE - A copy of the new ledger entry is recorded on a special trace dataset for reference and ledger recovery.

Using these three facilities, the status of any component at any point in time can be easily determined.

1.4 Data Base Integrity. By checking component status in the ledgers, CRR is able to prevent inadvertent improper reference to a data base. CRR forces a proper sequence of activities before data base access is allowed. These activities include log tape correction, data base backout, and data base recovery.

1.5 Batch Job Recovery. CRR facilitates the restart of IMS batch jobs by automatically invoking procedures when required for log tape correction and data base backout. CRR performs these functions to correct for system as well as application program failure. Any job which executes only one IMS data base updating program can be restarted from the beginning.

2. THE CRR LEDGERS

2.1 There are eight ledger data sets which contain job, data base and log tape information. Their content is shown here, including any special display characters. (See Par. 3 below for status codes.)

2.2 Job ledger contains the following information on IMS batch jobs and IMS on-line control region:

JOBNAME

* STATUS *

C- M COMPLETION CODE

P- PSB NAME

T- DATE-TIME (YYDDD HHMM) APPL PROGRAM EXECUTION LAST STARTED OR STOPPED

2.3 DBD ledger contains the following information on IMS data bases:

DBDNAME

* STATUS *

J- JOBNAME OF JOB WHICH LAST REFERENCED THE DATA BASE

M- BACKUP MODE ('FD' = FAST DUMP, 'IC' = IMAGE COPY)

I- USERID AS IT SHOULD APPEAR ON ANY RECOVERY JOB CARDS

C- CASECODE AS IT SHOULD APPEAR ON ANY RECOVERY JOB CARDS

D- DATE-TIME (YYDDD HHMM) LAST BACKUP WAS TAKEN

O- DATE-TIME (YYDDD HHMM) DATABASE WAS LAST MADE OFF-LINE TO THE IMS ON-LINE CONTROL REGION

2.4 Log Ledger contains information on IMS batch and on-line control region log tapes:

VOLSER

* STATUS *

J- JOBNAME OF JOB WHICH CREATED THE TAPE

Y- TYPE ('NORMAL', 'BACKOUT', 'INVALID')

S- VOLUME SEQUENCE NUMBER

T- DATE-TIME (YYDDD HHMM) TAPE WAS LAST OPENED OR CLOSED

2.5 Merged Log Ledger contains information on the merged log tapes created daily by Warrenville Operations. The format is the same as that of the Log Ledger.

2.6 Job-Data Base Ledger contains the name of each data base referenced by a job in the following format:

JOBNAME (TYPE) WHERE TYPE IS EITHER 'UPDATE' OR 'INQUIRY'

2.7 Data Base-Job Ledger contains the PSI of each system which updates the data base in the following format:

PSI

2.8 Data Base-Data Set Ledger contains the following information on the data sets which make up a data base:

DDNAME

DSNAME

DSNAME USED FOR BACKUP CATALOGING

2.9 Trace Ledger contains a record of all ledger changes in the following format:

TYPE	- TRACE RECORD TYPE (CURRENTLY ONLY'WRS')
MODULE	- ID OF CRR MODULE WHICH INVOKED THE CHANGE
FUNCTION	- ID OF CRR READ/WRITE MODULE FUNCTIONAL FOR MAKING THE CHANGE (FIRST CHARACTER INDICATES LEDGER TYPE: 'J' = JOB, 'D' = DATA BASE, 'L' = LOG, 'M' = MERGED LOG - THESE ARE THE ONLY LEDGERS WHICH ARE DYNAMICALLY UPDATED) (THIRD CHARACTER INDICATES CHANGE TYPE: 'A' = ADDED, 'R' = REPLACED)
DATE	- DATE RECORD WAS WRITTEN (YYDDD)
TIME	- TIME RECORD WAS WRITTEN (HHMM)
RECORD	- IMAGE OF THE LEDGER CHANGED RECORD

3. COMPONENT STATUS

3.1 The status of job, data base, and log components can take on the following values:

3.2 Job Status:

COMPLETE - APPLICATION PROGRAM EXECUTION HAS COMPLETED NORMALLY

FAILED - APPLICATION PROGRAM EXECUTION HAS ABENDED (NOTE - SYSTEM FAILURES WILL NOT CAUSE A JOB STATUS OF FAILED)

INQUIRY - APPLICATION PROGRAM EXECUTION HAS BEGUN BUT NO LOG TAPE DATA SET HAS BEEN OPENED

UPDATE - APPLICATION PROGRAM EXECUTION HAS BEGUN AND A LOG TAPE DATA SET HAS BEEN OPENED

3.3 Data Base Status:

CLOSED - DATA BASE IS NOT CURRENTLY BEING ACCESSED

INQUIRY - AN APPLICATION PROGRAM WHICH REFERENCES THE DATA BASE IN AN INQUIRY MODE HAS BEGUN EXECUTION

UPDATE - APPLICATION PROGRAM EXECUTION HAS BEGUN AND A LOG TAPE DATA SET HAS BEEN OPENED

3.4 Log Status:

CLOSED - TAPE HAS BEEN OPENED AND CLOSED SUCCESSFULLY

OPEN - TAPE HAS BEEN OPENED BUT NOT YET CLOSED SUCCESSFULLY

4. BATCH JOB PHASES

4.1 There are ten CRR phases that a CRR supported batch job passes through during execution. They are:

LOG TAPE VALIDATION
DATA BASE VALIDATION
EXECUTION PRE-PROCESSING
APPLICATION PROGRAM EXECUTION - LOG TAPE OPEN
APPLICATION PROGRAM EXECUTION - LOG TAPE OEV
APPLICATION PROGRAM EXECUTION - LOG TAPE CLOSE
APPLICATION PROGRAM EXECUTION - ABNORMAL TERMINATION
EXECUTION POST-PROCESSING

LOG TAPE VALIDATION
DATA BASE VALIDATION

4.2 The log tape and data base validation phases ahead of execution pre-processing will correct for system failures, those behind execution post-processing for program failures. A description of each phase is given below:

4.21 Log Tape Validation. The log tape validation phases are responsible for ensuring the integrity of IMS log tapes created by the job. If a system failure, log tape I/O error, or job cancel occurs after a log tape has been opened and before it has been closed, that tape cannot be used for data base restoration. Therefore, if a log tape with an ´open´ status is detected, log tape verification will invoke a program to copy the contents of that tape to a new tape, closing the new tape properly and replacing the volume serial number of the old tape with the volume serial number of the new tape in the log ledger. Log tape verification is also responsible for cataloging the volume serial numbers of any log tapes which will be required in the data base validation phase.

4.22 Data Base Validation. The data base validation phases are responsible for ensuring the integrity of all data bases referenced by the job. A failure during application program executing may create imcomplete results on data bases being referenced in the update mode. Therefore, if it is detected that an update program has been interrupted during execution, the IMS data base backout utility will be invoked to restore all update mode data bases to their status prior to the beginning of execution of the interrupted program. (Note - the backout utility will not be invoked unless a log tape was actually opened.)

4.23 Application Program Pre-Processing. This phase is responsible for ensuring that the application program does not begin execution until the status of any data base the job references is satisfactory. Execution will be prevented in any of the following circumstances:

- THE STATUS OF THE JOB IS UPDATE
- THE STATUS OF ANY DATA BASE TO BE REFERENCED BY THE JOB IS NOT ´CLOSED´ AND THE JOBNAME OF THE JOB TO LAST REFERENCE THE DATA BASE DOES NOT MATCH THE JOB NAME OF THE CURRENT JOB

If all tests are passed, the status of the job is changed to inquiry and application program execution is allowed to begin.

4.24 Application Program Execution - Log Tape Open. When a DD statement with the DDNAME IEFRDER is included in the execution step of the job (without a dummy specification) a log tape will be opened by the IMS control region. During this open processing, CRR modules are invoked to perform two functions:

- CREATE AN ENTRY IN THE LOG LEDGER FOR THE LOG TAPE WITH A STATUS OF ´OPEN´

- CHANGE THE STATUS OF THE JOB LEDGER ENTRY FOR THE JOB TO ´UPDATE´

4.25 Application Program Execution - Log Tape EOV. When enough log records have been written to fill a log tape, the IMS control region performs a volume change. After the change has taken place, CRR modules are invoked to:

- CHANGE THE STATUS OF THE LOG LEDGER ENTRY FOR THE OLD TAPE TO ´CLOSED´

- CREATE AN ENTRY IN THE LOG LEDGER FOR THE NEW TAPE WITH A STATUS OF ´OPEN´

4.26 Application Program Execution - Log Tape Close. When the application program completes processing, if a log tape has been opened, the IMS control region will perform a close. After close processing, CRR modules are invoked to:

- CHANGE THE STATUS OF THE LOG LEDGER ENTRY FOR THE CLOSED LOG TAPE TO ´CLOSED´

- CHANGE THE STATUS OF THE JOB LEDGER ENTRY FOR THE JOB TO ´INQUIRY´

4.27 Application Program Execution - Abnormal Termination Processing. When an application program abends, an IMS STAE EXIT receives control and attempts to close a log tape should one be open. If this close is successful, CRR modules are invoked to:

- CHANGE THE STATUS OF THE LOG LEDGER ENTRY FOR THE CLOSED LOG TAPE TO ´CLOSED´

4.28 Execution Post-Processing. If the application program terminates successfully, CRR modules are invoked to:

- CHANGE THE STATUS OF THE JOB LEDGER ENTRY FOR THE JOB TO ´COMPLETE´

- CHANGE THE STATUS OF THE DATA BASE LEDGER ENTRIES FOR ALL DATA BASES REFERENCED TO ´CLOSED´

DIVISION 5 - DATA MANAGEMENT AND ADMINISTRATION

CHAPTER 5 - DATA INTEGRITY

SECTION 3 - COMPUTER ASSISTED RESTART/RECOVERY FACILITIES AND JCL REQUIREMENTS FOR IMS BATCH JOBS

1. FACILITIES

1.1 The Computer Assisted Restart/Recovery System (CRR) provides the following facilities for IMS batch JOBs:

1.11 Protection against concurrent update of application data bases. The status of each application data base is maintained by CRR on a VS data set. This status is dynamically changed to reflect update and inquiry access by IMS batch JOBs and the IMS on-line control region (PRODIMS). If an IMS batch JOB attempts to access a data base which is currently being updated by another batch JOB or PRODIMS, that JOB will be abended by CRR before any data base access is attempted.

1.12 Log tape tracking, validation and consolidation. Information about all IMS log tapes is recorded by CRR on a VS data set when the log tapes are opened by an IMS control region (both batch and on-line). CRR modules invoked during batch JOB processing check the validity of previously created log tapes and, if necessary, copy log data from the "bad" tape to a new "good" tape. Also, on a daily basis, data from all log tapes created during the previous days processing is consolidated onto a "merged log" tape. These merged logs are stored in the Warrenville Tape Library for an extended period of time to be referenced for data base recoveries.

1.13 Dynamically invoked DB BACKOUT. The IMS DB BACKOUT utility is designed to reverse the effect of changes made to data bases by an IMS application program. Where a batch IMS update program fails, CRR modules (executed before and after application program execution) will automatically invoke the DB BACKOUT utility. Therefore, only in situations where DB BACKOUT will not work (e.g. I/O errors) must a full DB recovery be processed to recover from program failure.

2. JCL REQUIREMENTS

2.1 For recovery purposes, IMS batch JOBs are considered in one of categories: 1) those in which IMS data bases are updated by (one) application program within the JOB and 2) those in which IMS data bases are accessed but not updated (inquiry) by (one or more) application programs within the JOB.

2.2 Update JOB JCL requirements. The following JCL is required for the inclusion of CRR in an IMS data base update production job:

```
                                                              REF. SEC.
//JOBNAMEX JOB  ----------                                      2.201
//CRRUPRE  EXEC  CRRUPRE,                                       2.202
//                 JOB=JOBNAMEX,                                2.203
//                 RETPD=10                                     2,204
//DBPROC1.FORCE DD DUMMY                                        2.205
        .
        .
 (APPLICATION DB DD STATEMENTS)                                 2.206
        .
        .
//          EXEC (APPL. PROG/PROC EXEC STMT)                    2.207
//STEPLIB   DD DSN=SYS2.CRR.LOADLIB,DISP=SHR                    2.208
//          DD  (APPLICATION PROGRAM EXECUTION                  2.209
                 LIBRARIES)
//IEFRDER   DD DSN=SYS2.CRR.IMSLT.JOBNAMEX,                     2.210
//             UNIT=(TAPE,,DEFER),DISP=(NEW,KEEP),
//             LABEL=RETPD=10,
//             DCB=(RECFM=VBS,BLKSIZE=3968,LRECL=3964,BUFNO=2)
//DFSVSAMP DD DSN=SYS1.CONTROL(CRRLTWA),DISP=SHR                2.211
        .
        .
//JOBLEDGR DD DSN=SYS2.CRR.IMSDS.JOBLEDGR,DISP=SHR              2.212
//DBDLEDGR DD DSN=SYS2.CRR.IMSDS.DBDLEDGR,DISP=SHR
//LOGLEDGR DD DSN=SYS2.CRR.IMSDS.LOGLEDGR,DISP=SHR
//JDBLEDGR DD DSN=SYS2.CRR.IMSDS.JOBLEDGR,DISP=SHR
//DBJLEDGR DD DSN=SYS2.CRR.IMSDS.DBJLEDGR,DISP=SHR
//DBSLEDGR DD DSN=SYS2.CRR.IMSDS.DBSLEDGR,DISP=SHR
//TRALEDGR DD DSN=SYS2.CRR.IMSDS.TRALEDGR,DISP=SHR
        .
        .
 (APPLICATION PROGRAM EXECUTION DD STATEMENTS)
        .
        .
//CRRUPOST EXEC CRRUPOST,JOB=JOBNAMEX,RETPD=10                  2.213
//DBPROC2.FORCE DD DUMMY
        .
        .
 (APPLICATION DB DD STATEMENTS)
      .
      .
//
```

2.201 The name of the JOB (JOBNAMEX) is used by CRR to identify log tapes.

2.202 The CRRUPRE (update pre-processing) cataloged procedure contains JCL to execute CRR programs for log tape validation, data base validation and application program preprocessing (see CRR Concepts and Facilities).

2.203 The JOBNAME parameter is necessary for substitution into the log tape data set name.

2.204 The log tape retention period used should be consistent throughout the JOB. It indicates the number of days the JOBs log tapes will remain assigned in the Tape Library System.

2.205 This statement is used for forcing DD statement additions to the proper step.

2.206 These should match the data base DD statements included in the DB update step.

2.207 More than one application program may be executed, but there may be only <u>one</u> DB update program and it must be the <u>last</u> program executed (before CRRUPOST).

2.208 The CRR load library must be concatenated ahead of...
and
2.209 any other load libraries referenced.

2.210 The log tape data set created during DB update must have a data set name in the format shown and DD parameters as shown.

2.211 The log tape write ahead (LTWA) feature of IMS must be used to assure that DB updates are recorded on the log tape before any DB is actually changed. The LTWA feature is invoked with the DD statement shown.

2.212 One DD statement for each CRR data set must be included in the application update step.

2.213 The CRRUPOST (update post-processing) cataloged procedure contains JCL to execute CRR programs for log tape validation, data base validation and application program post-processing (see CRR Concepts and Facilities).

2.2 Inquiry JOB JCL requirements. The following JCL is required for the inclusion of CRR in an IMS data base inquiry production JOB:

```
                                                              REF. SEC.
//JOBNAMEY JOB  ----------                                      2.301
//CRRIPRE   EXEC  CRRIPRE                                       2.302
//          EXEC  (APPLICATION PROGRAM/PROCEDURE EXEC STMT)     2.303
       .
       .
 (APPLICATION PROGRAM EXECUTION DD STATEMENTS)
       .
       .
//CRRIPOST EXEC CRRIPOST                                        2.304
//
```

2.301 Although no log tapes are created in an inquiry JOB, the JOB name is still used by CRR modules for JOB and DB status tracking.

2.302 The CRRIPRE (inquiry pre-processing) cataloged procedure contains JCL to execute CRR programs for application program pre-processing.

2.303 One or more application inquiry programs may be executed.

2.304 The CRRIPOST (inquiry post-processing) cataloged procedure contains JCL to execute CRR programs for application program program post-processing.

DIVISION 5 - DATA MANAGEMENT AND ADMINISTRATION

CHAPTER 5 - DATA INTEGRITY

SECTION 4 - COMPUTER ASSISTED RESTART/RECOVERY UTILITIES

1. GENERAL

1.1 CRR utilities are provided to display CRR LEDGER contents, modify CRR LEDGER entries and generate JCL (Job Control Language) for the execution of IMS and CRR programs. These utilities are intended for use by Data Administration personnel responsible for performing data base restart/recovery procedures and require a knowledge of CRR and IMS restart/recovery facilities.

1.2 This section provides a description of the circumstances in which to use CRR utilities. See Division 3, Chapter 5 for details on the cataloged procedures used to invoke these utilities.

1.3 The following is a summary of the CRR utilities available.

		DIVISION 3
		CHAPTER 5
		SECTION 1
UTILITY	PROC-NAME	APPENDIX
Partitioned Data Set Ledger List	CRRPLIST	CC
Direct Data Set Ledger List	CRRDLIST	CD
Backup Processing	CRRBKUP	CE
Log Tape Termination	CRRLTAPE	CF
Backup Reset	CRRBAR	CG
Job Reset	CRRJAR	CH
Recovery JCL Generation	CRRRECV	CI

2. LEDGER DISPLAY

2.1 General. The CRR LEDGER data sets are of two types: direct and partitioned. Two cataloged procedures, one for each data set type, are available for listing the contents of CRR LEDGERS.

2.2 LEDGER data set names are as follows:

LEDGER	TYPE	DATA SET NAMES
JOB	P	SYS2.CRR.IMSDS.JOBLEDGER
JOB-DATA BASE	P	SYS2.CRR.IMSDS.JDBLEDGER
DATA BASE	P	SYS2.CRR.IMSDS.DBDLEDGER
DATA BASE-JOB	P	SYS2.CRR.IMSDS.DBJLEDGER

DATA BASE-DATA SET	P	SYS2.CRR.IMSDS.DBSLEDGER
LOG	D	SYS2.CRR.IMSDS.LOGLEDGER
MERGED LOG	D	SYS2.CRR.IMSDS.MLGLEDGER
TRACE	D	SYS2.CRR.IMSDS.TRALEDGER

2.3 Partitioned data set LEDGER display. Partitioned data set LEDGER can be displayed using the cataloged procedure CRRPLIST (see Div. 3, Chap. 5, Sect. 1, App. CC). In addition to this procedure for batch listing, the ISM data set display facility can be used for on-line access to partitioned data set LEDGERS. See Div. 5, Chap. 5, Sect. 2, paragraphs 2.2 - 3.4 for an explanation of ledger contents.

2.4 Direct data set LEDGER display. Direct data set ledgers can be displayed using the cataloged procedure CRRDLIST (see Div. 3, Chap.5, Sect. 1, App. CD). These LEDGERS can only be displayed in a batch mode.

3. BACKUP PROCESSING

3.1 The backup processing utility is used to record information in CRR ledgers about data base magnetic tape backups for future use in data base recovery procedures.

3.2 This utility should be executed as the last user processing step in each data base backup job:

```
//PSIBKUP JOB ...
//CRRIPRE EXEC CRRIPRE
   DB BACKUP STEP 1
         .
         .
         .
   DB BACKUP STEP M

//CRRBKUP EXEC CRRBKUP
//SYSIN DD *
   DBDNAME1  MODE1
         .
         .
         .
   DBDNAMEn  MODEn

//CRRIPOST EXEC CRRIPOST
```

3.9 This utility is invoked using the cataloged procedure CRRBKUP (see Div. 3, Chap. 5, Sect. 1, App. CE).

4. LOG TAPE TERMINATION

4.1 The log tape termination utility is used to close an IMS log tape which has not been successfully closed by normal IMS or CRR processing.

4.2 When to execute.

4.21 On-line log tapes. This utility is used by WDC operating personnel to close all log tapes left unclosed by PRODIMS.

4.22 Batch log tapes. Batch log tapes left unclosed by batch IMS code will normally be closed by CRR modules invoked in CRR pre and post processing. This utility is available for use in terminating batch log tapes in abnormal situations; it should, however, be used with discretion.

4.3 This utility is invoked using the cataloged procedure CRRLTAPE (see Div. 3, Chap. 5, Sect. 1, App. CF).

5. BACKUP RESET

5.1 The BACKUP RESET utility is used to modify the date-time and mode information stored in the Data Base LEDGER for an individual data base.

5.2 This procedure is intended for use where the most recent data base backup is physically or logically damaged and cannot be used for DB recovery. It should be executed (if required) before recovery JCL is generated by the RECOVERY JCL GENERATION utility.

5.3 This utility is invoked using the cataloged procedure CRRBAR (see Div. 3, Chap. 5, Sect. 1, App. CG).

6. JOB/DATA BASE STATUS RESET

6.1 The JOB/DATA BASE STATUS RESET utility is used to reset the status a job and all the data bases it references to 'COMPLETE' and 'CLOSED' respectively.

6.2 This procedure is intended to be used only where all normal procedures for restoring status have failed or cannot be used and the condition of all data bases referenced by the job is valid. Use of this procedure should be closely controlled by Data Administration personnel; for this reason, a two character code, controlled by Dept. 9412, must be supplied on the input. Local Data Administration personnel will be made aware of the current code.

6.3 This utility is invoked using the cataloged procedure CRRJAR (see Div. 3, Chap. 5, Sect. 1, App. CH).

7. RECOVERY JCL GENERATION

7.1 The RECOVERY JCL GENERATION utility is used to create Job Control Language (JCL) which is in turn executed to perform data base restoration and recovery.

7.2 This utility should be executed as the first phase of data base recovery. Using the CRR LEDGERs, the utility generates JCL to execute the following programs.

7.21 DB RESTORE. If the mode of the data base backup is 'FD', Innovation Data Processing DSF restore JCL is generated for each data set comprising the data bases to be restored. If the mode of the data base backup is 'IC', no DB restore JCL is generated and the Image Copy backup tape(s) are inputted directly to the DB RECOVERY program execution (see 7.23 below).

7.22 CHANGE ACCUMULATION. The generated JCL executes the IBM Data Base Change Accumulation Utility (DFSUCUM0) to separate data base change records for the data bases to be recovered from all other data base change records stored on the IMS log tape input, which includes <u>all</u> log tape data for <u>all</u> jobs (including PRODIMS) which updated the data bases indicated from the time of the data base backup to be used. This constitutes a "standard" log input. If it is desired to only partially recover data bases, that is, recover the data bases up to a point in time prior to the most recent update, the DD statements for all log tapes past the recovery point must be removed from the DFSUCUM0 JCL before it is executed. These DD statements will be concatenated under the DD name DFSULOG. The ability to remove DFSUCUM0 input is somewhat limited due to the daily merge of IMS log tapes. If a daily merged log tape (DSN=SYS2.CRR.IMSML.CRRDAILY) contains the partial recovery point, there is no way to separate the appropriate log records. In this event, a substitution of individual log tapes for the daily merged log tape must be made. This type of recovery should be coordinated by Dept. 9412.

7.23 DB RECOVERY. The generated JCL executes the IBM data Base Recovery Utility (DFSURDB0) once for each data set for each data base to be recovered. It is important to note that recovery is accomplished by data set and not by data base, allowing individual data sets within a data base to be recovered, possibly reducing recovery time. To do this, DFSURDB0 steps can be selectively executed, but care must be taken to bring all data sets to the same point in time. Note: when a partial recovery is being performed, all datasets for the data base must be recovered.

7.3 This utility is invoked using the cataloged procedure CRRRECV (see Div. 3, Chap. 5, Sect. 1, App. CI). The output of this procedure is placed on a data set defined by the following DD statement:

```
//JCL DD SYSOUT=P
```

To punch and print this data set during execution of the utility, the following JCL is required:

```
         //PSIRECV JOB
         //*MAIN CARDS=3
         //*FORMAT PR,DDNAME=JCL
         // EXEC CRRRECV
         //EXEC.SYSIN DD *
           DBDNAME1
              .
              .
              .
           DBDNAMEn
```

DIVISION 5 - DATA MANAGEMENT AND ADMINISTRATION

CHAPTER 5 - DATA INTEGRITY

SECTION 5 - COMPUTER ASSISTED RESTART/RECOVERY MESSAGES AND CODES

1. GENERAL

1.1 The Computer Assisted Restart/Recovery system may issue the following informational or diagnostic messages in the format:

AAANNN TEXT

where:

AAA - Is a three character module id.

NNN - Is a message sequence number for the module named by AAA.

TEXT - Is the message description.

2. MESSAGES

MESSAGE ID	MESSAGE TEXT
BAR001	The old backup mode specified on the input is not the same as the current backup mode on the DBD ledger entry. Correct the input and rerun.
BAR002	The old backup date and time specified on the input are not the same as the backup date and time on the DBD ledger entry. Correct the input and rerun the job.
BIC001	There are no log entries in the log ledger.
BIC002	The log tape for this job has not been successfully closed.
BIL001	The log tape for this job has not been successfully closed.
BIL002	The log volume serial table has exceeded the maximum number of entries.

MESSAGE ID	MESSAGE TEXT
BIL003	There are no log entries in the log ledger.
BIL004	The attempt to catalog a log tape has been unsuccessful.
BOP003	The job attempting to backout a data base is not the job specified in the DBD ledger entry.
BOP004	The attempted backout of a data base was unsuccessful.
BOP005	The number of volumes associated with this dataset name exceeded the maximum number allowed.
BUP001	Backup processing of the DBD ledger was attempted for a database currently open.
CIL001	There are no log entries in the log ledger.
DBG001	The requested entry could not be found on the DBJ ledger.
DBS001	A job attempted to access a data base which has a current status of update.
DBS002	A job with update intent attempted to access a data base already in update mode for another job.
DBS003	A job whose intent was not inquiry attempted to access a data base already in use in a non-inquiry mode.
DLG001	The requested entry could not be found on the DBD ledger. The specified member name can be found on the SYSMSG listing.
DSC001	The current status on the input does not match the current status of the DBD ledger entry referenced. Correct the input and rerun.
DSC002	An attempt was made to change the status of a DBD ledger entry to other than inquiry, update, or closed. Correct the input and rerun the job.
DSG001	The requested entry could not be found on the DBS ledger.

MESSAGE ID	MESSAGE TEXT
JAR001	An unauthorized attempt was made to change the status of a job ledger entry.
JAR002	An unauthorized attempt was made to change the status of a job ledger entry.
JAR003	The date and time on the input did not match the date and time on the job ledger entry. Correct the input and rerun the job.
JDB001	The requested entry could not be found on the JDB ledger.
JLG001	The requested entry could not be found on the job ledger. The specified member name can be found on the SYSMSG listing.
LEU001	There are no log entries in the log ledger.
LLL001	A log tape for this PSI has been found that has not been successfully closed.
LLL002	A log tape for this PSI has been found that is not usable.
LLP001	An invalid type of operation was requested during log ledger processing.
LTP002	An unsuccessful link to log tape termination was detected during log tape processing.
LTP003	There are no log entries in the log ledger.
MLU001	The merged log tape volume serial number could not be found.
MLU002	The merged log volume serial table has exceeded the maximum number of entries.
PRP001	The job´s intended activity for the data base referenced could not be allowed at this time. See messages DBS001, DBS002, and DBS003 for possible causes.
PRP002	The update status of this job prohibits execution at this time.
RWP010	The read/write processing program has detected an end of file condition on the DBD ledger,

MESSAGE ID MESSAGE TEXT

job ledger, log ledger, or mlg ledger when this condition should not exist.

TRP010 The trace processing program has detected an end of file condition on the trace ledger when this condition should not exist.

TTU001 The volume serial number specified in the input could not be found in the log ledger. Correct the input and rerun the job.

TTU002 The job specified in the input could not be found on the job ledger. Correct the input and rerun the job.

TTU003 An unsuccessful link to log tape termination was detected during execution of the log tape termination utility program.

3. ABEND CODE

3.1 All CRR abends produce the user abend code 2001.

CRRPLIST - PARTITIONED DATA SET LIST

1. GENERAL

1.1 This procedure invokes the program WECOPY to list the contents of the ledgers which are partitioned data sets (JOB, DBD, JOB-DBD, DBD-JOB, DBD-DS LEDGERS).

2. USING CRRPLIST

2.1 // EXEC CRRPLIST,LED-P1,MEM=P2
where:

LED - three character ID of the leuger to be listed
MEM - maximum eight character ID of the entry to be listed

3. EXAMPLE OF USE

3.1 // EXEC CRRPLIST,LED=JOB,MEM=PCCWY020

This execution will list the job ledger entry for the job PCCWY020.

4. CATALOGED PROCEDURE LISTING

```
//CRRPLIST PROC LEV1=SYS2
//LIST      EXEC PGM=WECOPY
//SYSPRINT DD SYSOUT=A
//SYSUT1    DD DSN=&LEV1..CRR.IMSDS.&LED.LEDGR(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=A
```

CRRDLIST - DIRECT DATA SET LIST

1. GENERAL

1.1 This procedure invokes the program WECOPY to list the contents of the ledgers which are direct data sets (LOG, MERGED LOG, and TRACE ledgers).

2. USING CRRDLIST

```
// EXEC CRRDLIST,LED=P1,ENTRIES=P2,SKIP=P3
```

P1 - three character ID of the ledger to be listed (LOG, MLG, TRA)

P2 - number of entries to be listed (default is 100)

P3 - number of entries to be skipped (default is 0)

3. EXAMPLE OF USE

```
// EXEC CRRDLIST,LED=LOG,ENTRIES=25
```

This execution will list the first 25 log ledger entries.

4. CATALOGED PROCEDURE LISTING

```
//CRRDLIST PROC LEV1=SYS2,ENTRIES=100,SKIP=0
//LIST     EXEC PGM=WECOPY,PARM=(&ENTRIES,&SKIP)
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=&LEV1..CRR.IMSDS.&LED.LEDGR,DISP=SHR
//SYSUT2   DD SYSOUT=A
```

CRRBKUP - BACKUP PROCESSING

1. GENERAL

1.1 This procedure invokes the program CRRBUP to change the backup mode on the DBD ledger for a given data base. It also updates the date and time stamp of the data base ledger entry using the current date and time expressed as YYDDDHHMM.

2. USING CRRBKUP

```
// EXEC CRRBKUP
//SYSIN    DD *
DBDNAME MODE
```

DBDNAME - (col. 1) a maximum eight character dbdname is allowed

MODE - (col. 10) two character backup mode

3. EXAMPLE OF USE

```
// EXEC CRRBKUP
//SYSIN    DD *
DBDNAME1 FD
```

This execution will change the backup mode of data base DBDNAME1 to FD and update the date/time stamp of the ledger entry.

4. CATALOGED PROCEDURE LISTING

```
//CRRBKUP   PROC LEV1=SYS2,LIBRARY='SYS2.CRR.LOADLIB'
//EXEC      EXEC PGM=CRRBUP
//STEPLIB   DD DSN=&LIBRARY,DISP=SHR
//JOBLEDGR  DD DSN=&LEV1..CRR.IMSDS.JOBLEDGR,DISP=SHR
//DBDLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBDLEDGR,DISP=SHR
//LOGLEDGR  DD DSN=&LEV1..CRR.IMSDS.LOGLEDGR,DISP=SHR
//JDBLEDGR  DD DSN=&LEV1..CRR.IMSDS.JDBLEDGR,DISP=SHR
//DBJLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBJLEDGR,DISP=SHR
//TRALEDGR  DD DSN=&LEV1..CRR.IMSDS.TRALEDGR,DISP=SHR
//MLGLEDGR  DD DSN=&LEV1..CRR.IMSDS.MLGLEDGR,DISP=SHR
```

CRRLTAPE - LOG TAPE TERMINATION

1. GENERAL

1.1 This procedure invokes the program CRRTTU which will close the volume named in the input and update the log ledger with the new volume information.

2. USING CRRLTAPE

```
// EXEC CRRLTAPE
//SYSIN   DD *
VOLSER    JOBNAME
```

VOLSER - (col. 1) volume serial number of the tape

JOBNAME - (col. 11) name of the job that created this tape

3. EXAMPLE OF USE

```
// EXEC CRRLTAPE
//SYSIN     DD *
700700    LTAPERR
```

This execution will copy log tape 700700 created by job LTAPERR to a new volume and update the log ledger.

4. CATALOGED PROCEDURE LISTING

```
//CRRLTAPE   PROC LIB='SYS2.CRR',LEV1=SYS2,LOG=LOG,LOGLEV1=SYS2,LT=LT,
//             RETPD=2,JOB=NUL
//CRRTTU    EXEC PGM=CRRTTU
//STEPLIB   DD DSN=&LIB..LOADLIB,DISP=SHR
//SYSUDUMP  DD  SYSOUT=A
//PRINT     DD  SYSOUT=A
//LOGLEDGR  DD DSN=&LEV1..CRR.IMSDS.&LOG.LEDGR,DISP=SHR
//TRALEDGR  DD DSN=&LEV1..CRR.IMSDS.TRALEDGR,DISP=SHR
//LOGIN     DD DSN=&LOGLEV1..CRR.IMS&LT..&JOB,UNIT=(TAPE,,DEFER),
//             VOL=(,RETAIN,,,SER=(INPVOL)),DISP=(OLD,PASS),
//             DCB=(RECFM=VBS,BLKSIZE=3968,LRECL=3964,BUFNO=2)
//LOGOUT    DD DSN=&LOGLEV1..CRR.IMS&LT..&JOB,UNIT=TAPE,DISP=(,PASS),
//             VOL=(,RETAIN),LABEL=RETPD=&RETPD,
//             DCB=(RECFM=VBS,BLKSIZE=3968,LRECL=3964,BUFNO=2)
```

CRRBAR - BACKUP RESET

1. GENERAL

1.1 This procedure is used to reset the date-time and mode stored in the data base ledger for an individual data base. This procedure is intended to be used when the most recent backup is physically damaged and cannot be used by DBD recovery. All dates used are in the format YYDDD space HHMM. This field is 10 bytes in length.

2. USING CRRBAR

```
// EXEC CRRBAR
//SYSIN DD *
DBDNAME  DM CDT          PM  PDT
```

where:

DBDNAME - (col. 1) data base DBD name

DM - (col. 10) current backup mode

CDT - - (col. 13) current backup date-time

PM - (col. 24) previous backup mode, which is to replace the current backup mode

PDT - (col. 27) previous backup date-time, which is to replace the current backup date-time

3. EXAMPLE OF USE

```
// EXEC CRRBAR
//SYSIN DD *
PSIDDBD1 IC 76174 0325 IC 76172 0346
```

This execution Will change the backup date-time from 76174 0325 to 76172 0346 in the data base ledger entry for the data base PSIDDBD1.

4. CATALOGED PROCEDURE LISTING

```
//CRRBAR    PROC LEV1=SYS2,LIBRARY='SYS2.CRR.LOADLIB'
//EXEC      EXEC PGM=CRRBAR
//STEPLIB   DD DSN=&LIBRARY,DISP=SHR
//JOBLEDGR  DD DSN=&LEV1..CRR.IMSDS.JOBLEDGR,DISP=SHR
//DBDLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBDLEDGR,DISP=SHR
//LOGLEDGR  DD DSN=&LEV1..CRR.IMSDS.LOGLEDGR,DISP=SHR
//JDBLEDGR  DD DSN=&LEV1..CRR.IMSDS.JDBLEDGR,DISP=SHR
//DBJLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBJLEDGR,DISP=SHR
//TRALEDGR  DD DSN=&LEV1..CRR.IMSDS.TRALEDGR,DISP=SHR
//MLGLEDGR  DD DSN=&LEV1..CRR.IMSDS.MLGLEDGR,DISP=SHR
//DBSLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBSLEDGR,DISP=SHR
//SYSUDUMP  DD SYSOUT=A
```

CRRJAR - JOB/DATA BASE STATUS RESET

1. GENERAL

1.1 This procedure is used to reset the status of a job and all data bases it references to 'COMPLETE' and 'CLOSED' respectively. This procedure is intended to be used only when all normal procedures for restoring these statuses have failed and the condition of all data bases referenced by the job are valid.

2. USING CRRJAR

```
// EXEC CRRJAR
//SYSIN DD *
JOBNAME  DATE-TIME  CD
```

where:

JOBNAME - (col. 1) name of the job for which statuses are to be reset

DATE-TIME - (col. 10) YYDDD-HHMM date and time which match the date-time on the job ledger entry A space is required between YYDDD and HHMM.

CD - (col. 21) two character code to be supplied by the local DBA

3. EXAMPLE OF USE

```
// EXEC CRRJAR
//SYSIN DD *
PCCWY020 76158 1430 WP
```

This execution will reset the job and data base statuses for the job PCCWY020.

4. CATALOGED PROCEDURE LISTING

```
//CRRJAR    PROC LEV1=SYS2,LIBRARY='SYS2.CRR.LOADLIB'
//EXEC      EXEC PGM=CRRJAR
//STEPLIB   DD DSN=&LIBRARY,DISP=SHR
//JOBLEDGR  DD DSN=&LEV1..CRR.IMSDS.JOBLEDGR,DISP=SHR
//DBDLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBDLEDGR,DISP=SHR
//LOGLEDGR  DD DSN=&LEV1..CRR.IMSDS.LOGLEDGR,DISP=SHR
//JDBLEDGR  DD DSN=&LEV1..CRR.IMSDS.JDBLEDGR,DISP=SHR
//DBJLEDGR  DD DSN=&LEV1..CRR.IMSDS.DBJLEDGR,DISP=SHR
//TRALEDGR  DD DSN=&LEV1..CRR.IMSDS.TRALEDGR,DISP=SHR
//MLGLEDGR  DD DSN=&LEV1..CRR.IMSDS.MLGLEDGR,DISP=SHR
```

CRRRECV - RECOVERY JCL GENERATION

1. GENERAL

1.1 This procedure invokes the program CRRRJP which will generate recovery JCL for the given DBD names.

2. USING CRRRECV

```
// EXEC CRRRECV
//EXEC.SYSIN DD *
DBDNAME
```

DBDNAME - (col. 1) a maximum of eight characters is allowed. Any number of input cards may be submitted in a single execution.

3. EXAMPLE OF USE

```
// EXEC CRRRECV
//EXEC.SYSIN DD *
DBDNAME1
DBDNAME2
DBDNAME3
```

This execution will generate the required recovery JCL for the data bases DBDNAME1, DBDNAME2, and DBDNAME3.

4. CATALOGED PROCEDURE LISTING

```
//CRRRECV  PROC LEV1=SYS2,MOD=RJP,LIB='SYS2.CRR.LOADLIB'
// EXEC PGM=IEFBR14
//LOGIN    DD DSN=&&LOGIN,UNIT=SYSDA,DISP=(NEW,PASS),SPACE=(TRK,10)
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//EXEC     EXEC PGM=CRR&MOD
//STEPLIB  DD DSN=&LIB,DISP=SHR
//JOBLEDGR DD DSN=&LEV1..CRR.IMSDS.JOBLEDGR,DISP=SHR
//DBDLEDGR DD DSN=&LEV1..CRR.IMSDS.DBDLEDGR,DISP=SHR
//LOGLEDGR DD DSN=&LEV1..CRR.IMSDS.LOGLEDGR,DISP=SHR
//JDBLEDGR DD DSN=&LEV1..CRR.IMSDS.LDBLEDGR,DISP=SHR
//DBJLEDGR DD DSN=&LEV1..CRR.IMSDS.DBJLEDGR,DISP=SHR
//TRALEDGR DD DSN=&LEV1..CRR.IMSDS.TRALEDGR,DISP=SHR
//MLGLEDGR DD DSN=&LEV1..CRR.IMSDS.MLGLEDGR,DISP=SHR
//DBSLEDGR DD DSN=&LEV1..CRR.IMSDS.DBSLEDGR,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//JCL      DD SYSOUT=P
//LOGIN    DD DSN=&&LOGIN,DISP=OLD
```