IBM

System/360
Information Management
System

IBM

System/360 Information Management System

Education Guide

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Address comments concerning the contents of this publication to IBM Corporation, DPD Education Development, Education Center, Poughkeepsie, New York 12602.

CONTENTS

Section 1
Introduction

INTRODUCTION

This Education Guide is intended for use in the Information Management System/360 classes.

CONCEPTS AND FACILITIES
APPLICATION PROGRAMMING
IMPLEMENTATION AND INTERNALS

Section 4 of this Education Guide consists of 6 modules, as can be seen from the general outline in Section 3. The first five of these modules are intended for use in the Concepts and Facilities course with no hands on. Depending on facilities available, it may be desirable to demonstrate IMS/360 at the end of the course.

The IMS/360 Application Programming course also makes use of the first five modules of Section 4 and includes hands on experience for the student. If it is desired to teach application programmers for a batch only environment, module 4 of Section 4 (ONLINE PROCESSING) may be dropped from the course.

The IMS/360 Implementation and Internals course consists of all material covered in the Application Programming course plus the last module of Section 4 (Implementation and Internals) with additional hands on experience.

The first week of the Application Programming course and the Implementation and Internals course can be a combined class. At the end of the first week, the Application Programming course would be complete and the Implementation and Internals course would continue the next week.

Considerations for generating a system to be used for hands on classes are presented in Section 5 - Instructor Materials.

IMS/360 COURSES

COURSES	SUGGESTED TIME	MODULES OF SECTION 4	NOTES
CONCEPTS AND FACILITIES	2 DAYS	1-2-3-4-5	NO HANDS ON
APPLICATION PROGRAMMING	4 OR 5 DAYS	1-2-3-4-5	HANDS ON
IMPLEMENTATION AND INTERNALS	7 TO 10 DAYS	1-2-3-4-5-6	HANDS ON

MODULES	TITLE
1	OBJECTIVES AND PURPOSE OF IMS/360
2	AN OVERVIEW OF IMS/360
3	DATA LANGUAGE/I
4	ONLINE PROCESSING
5	FUNCTIONAL DESCRIPTION
6	IMPLEMENTATION AND INTERNALS

NOTE: IN ADDITION TO THE ABOVE THREE COURSES, MODULES 1 AND 2 CAN BE USED AS AN OVERVIEW IN A SURVEY COURSE.

IF IT IS DESIRED TO TEACH APPLICATION PROGRAMMERS WHO WILL NOT BE USING TELEPROCESSING, MODULE 4 CAN BE DROPPED FROM THE COURSE.

Section 2
Course Description

IBM®

Course Description

SYSTEM/360 INFORMATION MANAGEMENT SYSTEM - IMPLEMENTATION AND INTERNALS

Course Code - U3672

Duration - 8 student days

Audience - System analysts and programmers who will be responsible for installing IMS/360

Prerequisites - Successful completion of S/360 OS System Control for Programmers (H3682), S/360 OS Data Management Coding (H3667), and S/360 OS Advanced Coding (H3670). In addition to these courses a working knowledge of COBOL, PL/I, or Assembler Language is required.

Objectives - Upon successful completion of the course, the student should be able to:

1. List the advantages of using IMS/360 and describe the difference between traditional data processing and the data base approach.

2. Trace the flow of information and control through the system from the time a request is entered at a terminal until the response is received at the terminal.

3. Write a program specification block and a batch application program using all Data Language/I call functions.

4. Write a message processing program to handle input from a terminal, interaction with a data base, and output to a terminal.

5. Describe the major functional areas such as System Definition, System Log, Checkpoint/Restart, and Security Maintenance and their significance.

6. Generate an IMS/360 system for batch or online processing for PCP (batch only), MFT-II, or MVT; write a data base description; create a data base; use the utilities for processing the log tape; prepare control cards for the security maintenance program, and trace the flow of information and control through the major control blocks of IMS/360.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Address comments concerning the contents of this publication to IBM Corporation, DPD Education Development, Education Center, Poughkeepsie, New York 12602.

6/69

Material Requirements

Student Materials

Title	Form No.	Abstract Ref.
IBM System/360 Operating System COBOL Language	C28-6516	*
Information Management System/360 Application Description Manual	H20-0524	None
Information Management System/360 Program Description Manual	***	None
Information Management System/360 System Operations Manual	***	None
Information Management System/360 Machine Operations Manual	***	None

Instructor Materials (in addition to the above)

Information Management System/360 Implementation and Internals Education Guide	R20-4142	See Below
Information Management System/360 System Manual	***	None

* IBM System/360 Bibliography (A22-6822)
*** To be released

Abstract -

R20-4142 Education Guide
8½" x 11" Looseleaf Instructor Outline, 400 pages
(Red Cover)

This guide contains a detailed course outline which can be used for teaching three Information Management System/360 courses - Concepts and Facilities, Application Programming, and Implementation and Internals.

Included in the guide are class exercises and paper masters which may be used in preparing overhead projection transparencies.

SYSTEM/360 INFORMATION MANAGEMENT SYSTEM - APPLICATION PROGRAMMING

Course Code - U3671

Duration - 5 student days

Audience - Application programmers who will be responsible for writing programs in COBOL, PL/I, or Assembler Language using Data Language/I to interact with a Data Base.

Prerequisites - The student should have actual programming experience in COBOL, PL/I, or Assembler Language

Objectives - Upon successful completion of the course, the student should be able to:

1. List the advantages of using IMS/360 and describe the difference between traditional data processing and the data base approach.

2. Trace the flow of information and control through the system from the time a request is entered at a terminal until the response is received at the terminal.

3. Write a program specification block and a batch application program using all Data Language/I call functions.

4. Write a message processing program to handle input from a terminal, interaction with a data base, and output to a terminal.

5. Describe the major functional areas such as System Definition, System Log, Checkpoint/Restart, and Security Maintenance and their significance.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Address comments concerning the contents of this publication to IBM Corporation, DPD Education Development, Education Center, Poughkeepsie, New York 12602.

Material Requirements

Student Materials

Title	Form No.	Abstract Ref.
IBM System/360 Operating System COBOL Language	C28-6516	*
Information Management System/360 Application Description Manual	H20-0524	None
Information Management System/360 Program Description Manual	***	None

Instructor Materials (in addition to the above)

Title	Form No.	Abstract Ref.
Information Management System/360 Application Programming Education Guide	R20-4142	See Below
Information Management System/360 System Operations Manual	***	None
Information Management System/360 Machine Operations Manual	***	None

* IBM System/360 Bibliography (A22-6822)
*** To be released

Abstract -

R20-4142 Education Guide
8½" x 11" Looseleaf Instructor Outline, 400 pages
(Red Cover)

This guide contains a detailed course outline which can be used for teaching three Information Management System/360 courses - Concepts and Facilities, Application Programming, and Implementation and Internals.

Included in the guide are class exercises and paper masters which may be used in preparing overhead projection transparencies.

IBM

Course Description

SYSTEM/360 INFORMATION MANAGEMENT SYSTEM - APPLICATION PROGRAMMING

Course Code - U3670

Duration - 2 student days

Audience - Data processing managers who are responsible for implementing IMS/360 and data processing managers who require an understanding of the system facilities in order to evaluate IMS/360 in relationship to other alternatives.

Prerequisites - Successful completion of S/360 for Data Processing Management (T3602) or equivalent experience.

Objectives - Upon successful completion of the course, the student should be able to:

1. List the advantages of using IMS/360 and describe the difference between traditional data processing and the data base approach.

2. Trace the flow of information and control through the system from the time a request is entered at a terminal until the response is received at the terminal.

3. Define a file in terms of a hierarchical file structure.

4. Distinguish the difference between a batch processing program and a message processing program.

5. Describe the major functional areas such as System Definition, System Log, Checkpoint/Restart, and Security Maintenance and their significance.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Address comments concerning the contents of this publication to IBM Corporation, DPD Education Development, Education Center, Poughkeepsie, New York 12602.

6/69

Material Requirements

Student Materials

Title	Form No.	Abstract Ref.
Information Management System/360 Application Description Manual	H20-0524	None
Information Management System/360 Program Description Manual	***	None

Instructor Materials (in addition to the above)

Information Management System/360 Concepts and Facilities Education Guide	R20-4142	See Below
Information Management System/360 System Operations Manual	***	None
Information Management System/360 Machine Operations Manual	***	None

* IBM System/360 Bibliography (A22-6822)
*** To be released

Abstract -

R20-4142 Education Guide
8½" x 11" Looseleaf Instructor Outline
(Red Cover)

This guide contains a detailed course outline which can be used for teaching three Information Management System/360 courses - Concepts and Facilities, Application Programming, and Implementation and Internals.

Included in the guide are class exercises and paper masters which may be used in preparing overhead projection transparencies.

Section 3
General Course Outline

GENERAL COURSE OUTLINE

Objectives and Purpose of IMS/360

A. What is IMS/360
B. What is a Data Base
C. Primary Objectives of IMS/360
D. A Typical Company's File Organization
E. The Traditional Approach
F. The Data Base Approach
G. Conceptual View of IMS/360
H. IMS/360 is an Extension of OS/360

An Overview of IMS/360

A. Information Management System/360
B. Requirements of an Information System
C. Major Features of IMS/360
D. Data Base Facilities
E. Data Communication Facility
F. OS/360 Extension
G. Data Language/I
H. Logical Data Structure
I. Procedure for Using Data Base Facility
J. Data Storage Techniques
K. Using the Data Communication Facility
L. System Organization and Data Flow

Data Language/I

A. General Description
B. Data Language/I vs OS/360 Data Management
C. Data Language/I Structures - General
D. Dependent Segments
E. Data Language/I Structure - Rules
F. Segment Sensitivity
G. Key Length and Representation
H. Storage Technique
I. Data Base Description
J. Program Specification Block
K. Application Program Definition
L. Data Language/I Calls
M. Physical Storage
N. Loading the Application Program

Online Processing (Message Processing)

A. The Online Environment
B. Preparing the Application Program
C. Calling for an Input Message
D. Outputting a Response
E. The Message Format
F. Message Processing Region Simulation
G. An Input Message Editor

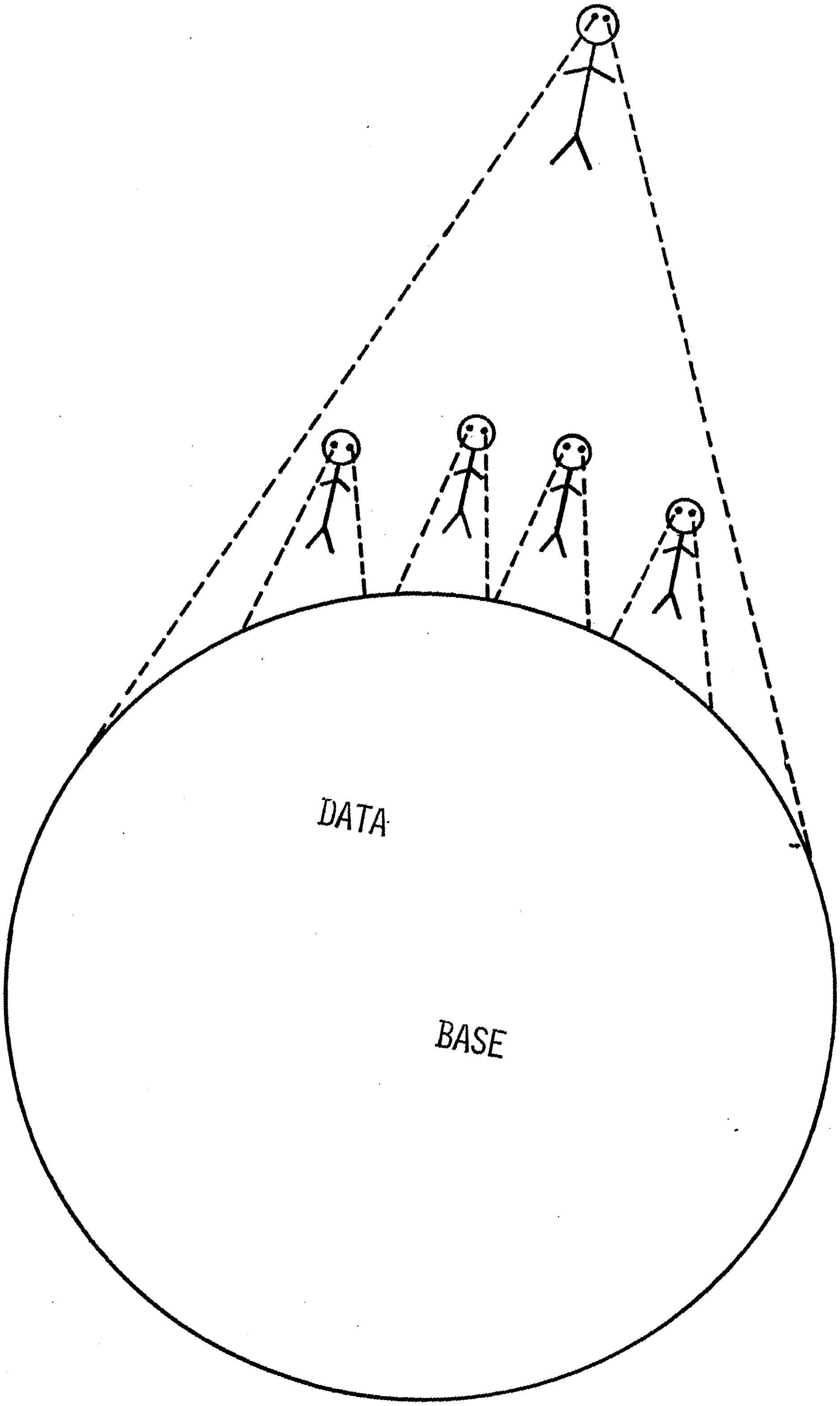
Functional Description

A. System Definition
B. IMS/360 System Log
C. Checkpoint, Restart, Data Base Dump and Recovery
D. Security Maintenance

Implementation and Internals

A. IMS/360 Initialization and System Flow
B. Communications Control
C. Switched Communications Networks
D. System Generation
E. Security Maintenance
F. Command Language Facilities
G. Checkpoint, Restart, Data Base Dump and Recovery
H. System Log
I. Message Queue Space Allocation
J. Estimating DASD Space for Data Bases
K. Estimating Core Storage Requirements for IMS/360

Section 4
Detailed Course Outline

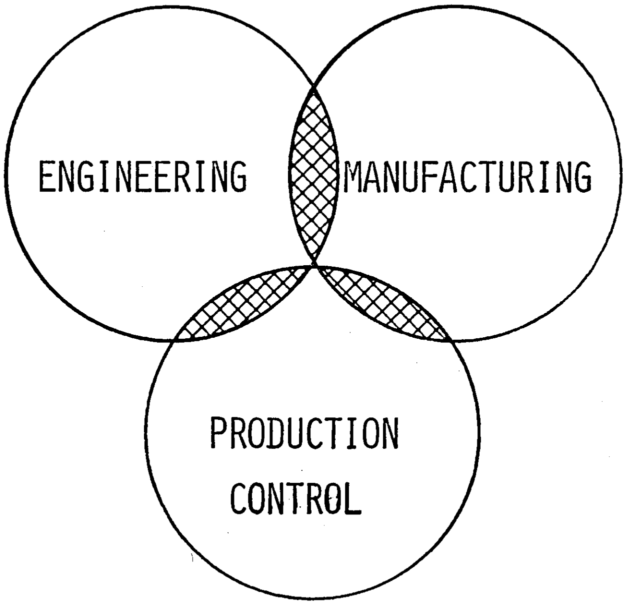


4.1.5

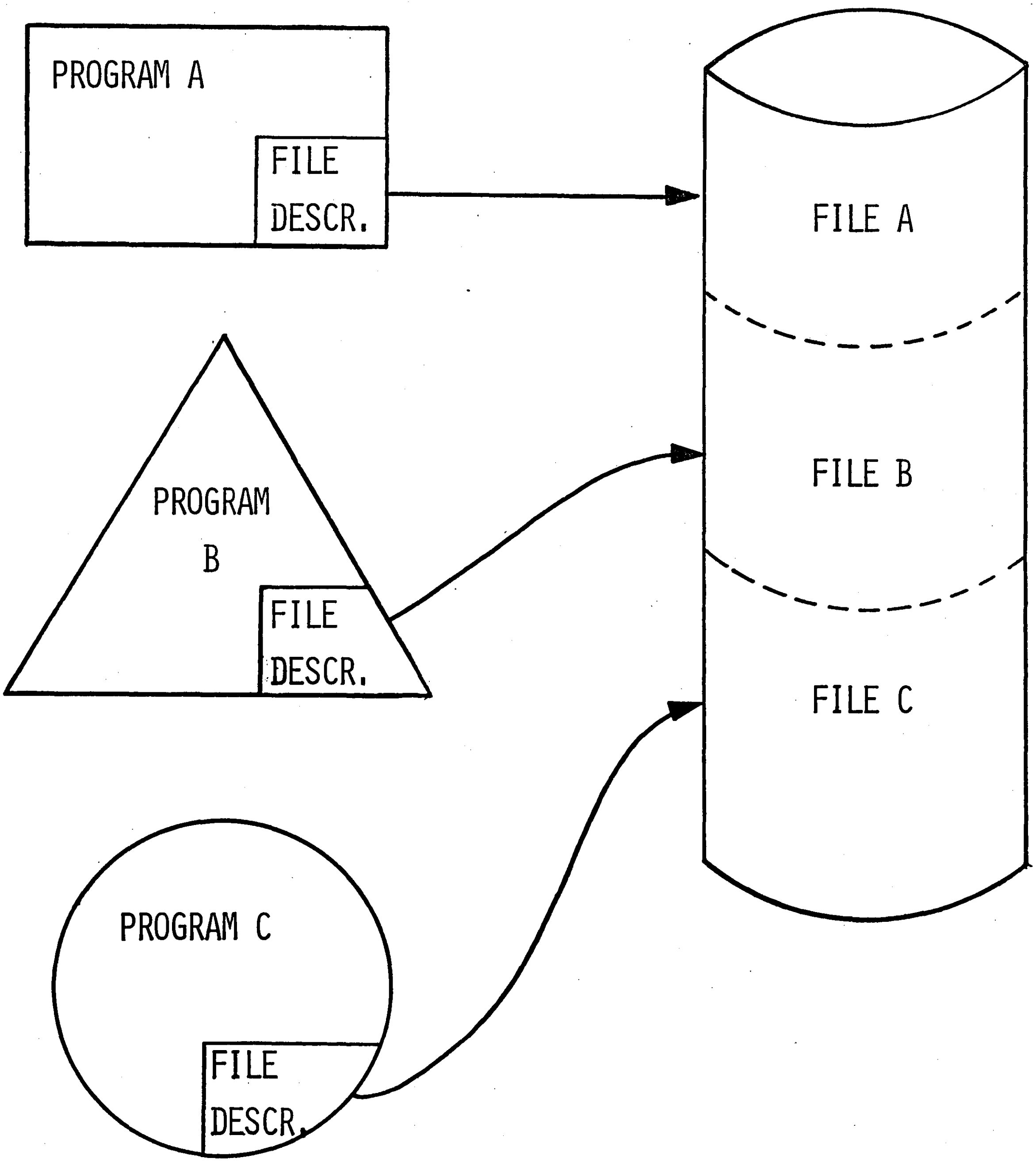
PRIMARY OBJECTIVES OF IMS/360

- ELIMINATION OF REDUNDANT DATA
- REDUCTION IN PROGRAM MAINTENANCE
- PROVIDE ON-LINE MAINTENANCE OF A DATA BASE

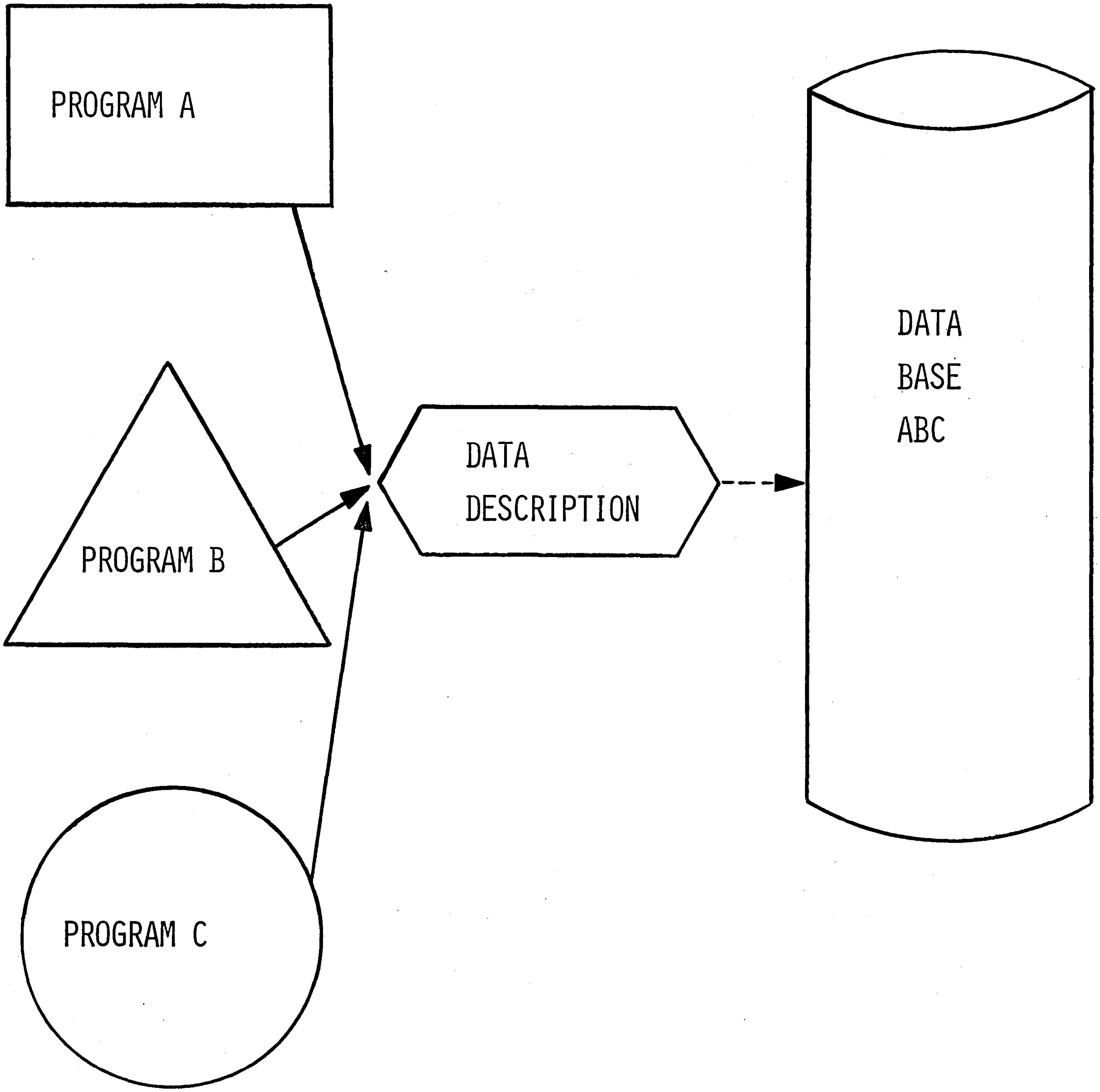
COMPANY DATA BASE



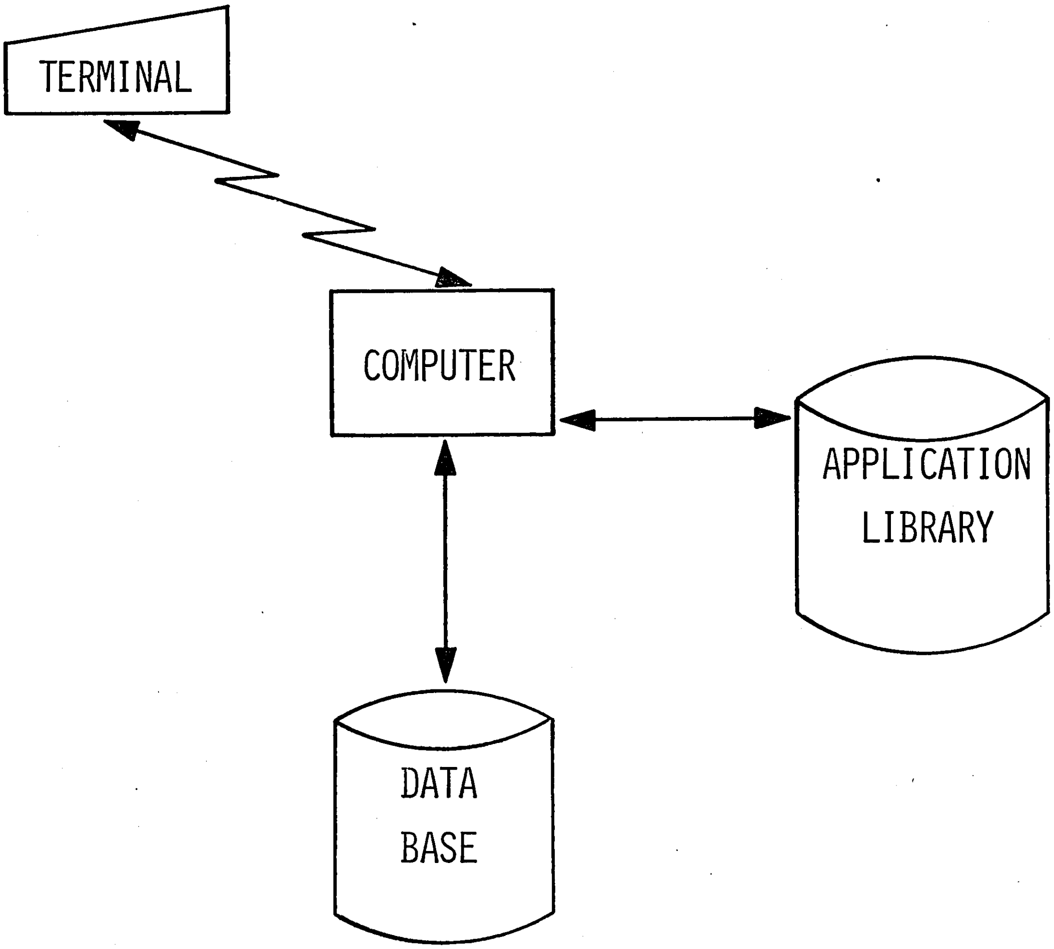
TRADITIONAL APPROACH



DATA BASE APPROACH



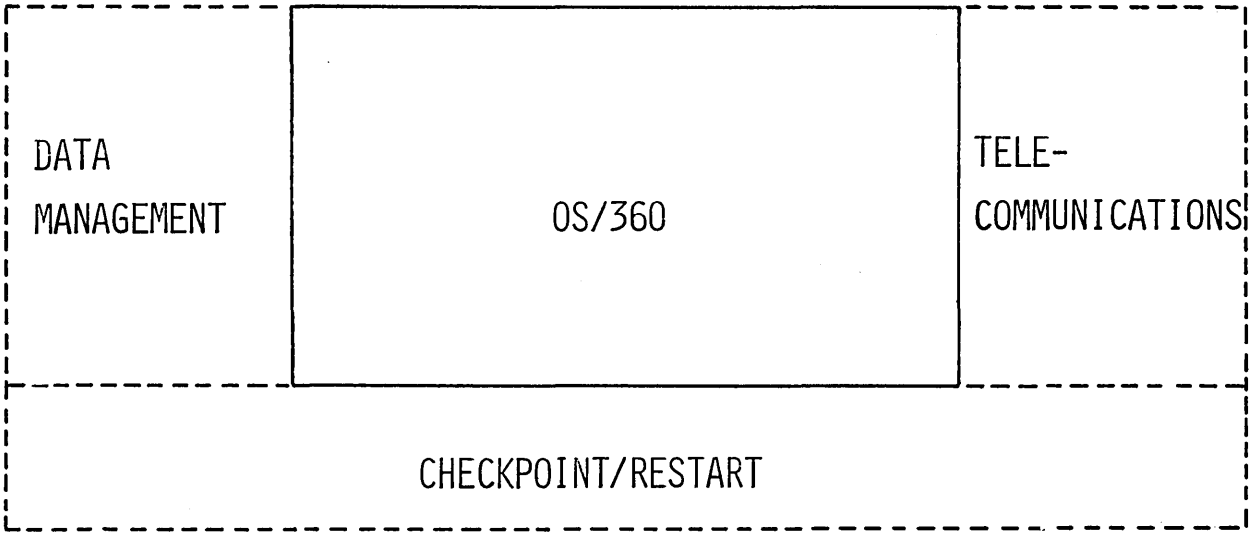
IMS/360



DATA BASE FACILITY

DATA COMMUNICATIONS FACILITY

AN EXTENSION OF OS/360



DATA MANAGEMENT

OS/360

TELE-COMMUNICATIONS

CHECKPOINT/RESTART

INFORMATION MANAGEMENT SYSTEM/360

IS A GENERAL PURPOSE DATA BASE/DATA

COMMUNICATIONS SYSTEM

INFORMATION SYSTEM REQUIREMENTS

EVOLUTIONARY GROWTH

- FROM BATCH TO ONLINE
- FROM APPLICATION DATA FILES TO LARGE DATA BASES

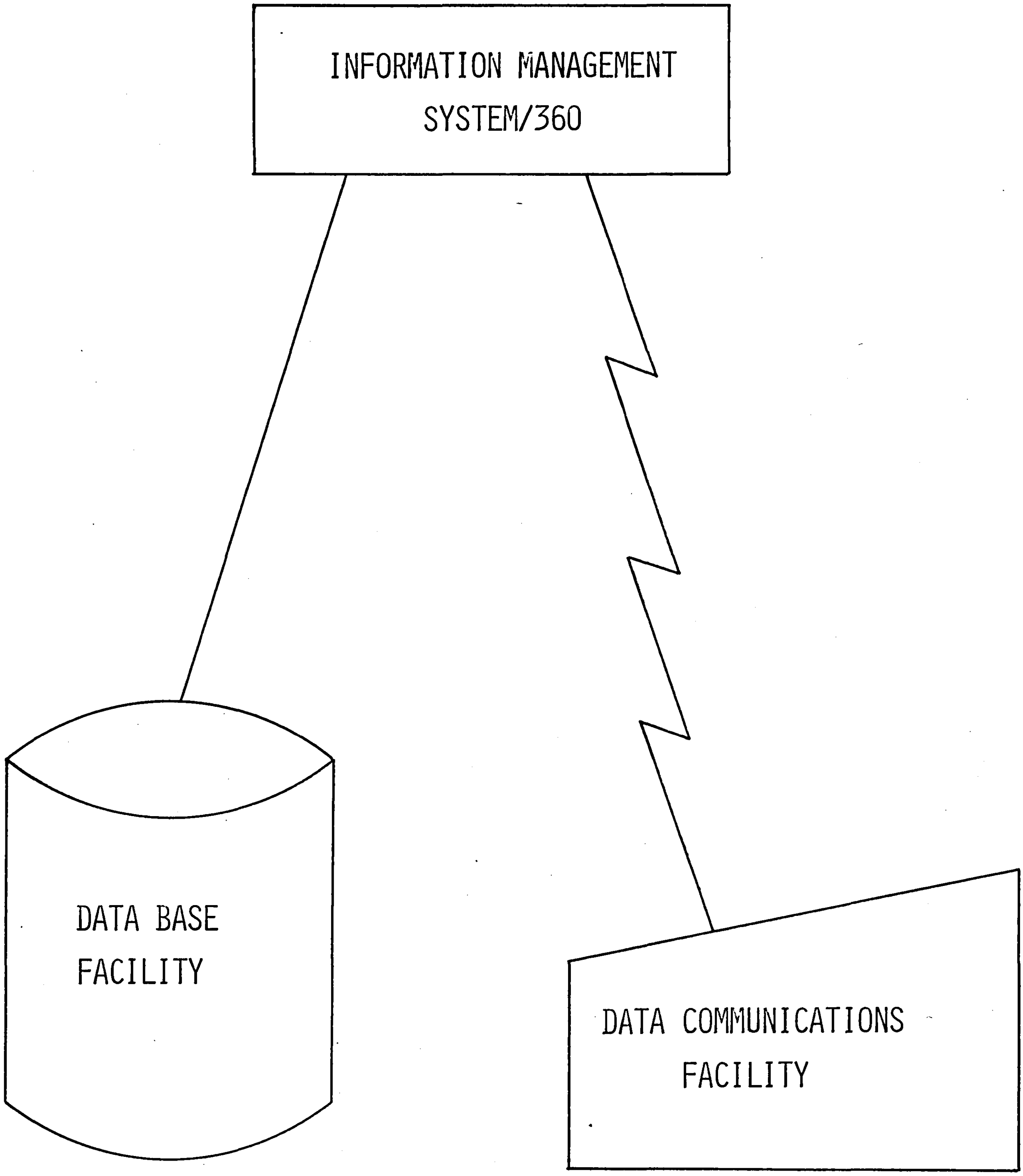
SUPPORT BATCH AND ONLINE CONCURRENTLY

LARGE VOLUME WITH RAPID RESPONSE

HIGH DEGREE OF DATA INDEPENDENCE

APPLICATIONS IN HIGH LEVEL LANGUAGE

- IN BATCH AND TELEPROCESSING ENVIRONMENT


INFORMATION MANAGEMENT SYSTEM/360
DATA BASE FACILITY
DATA COMMUNICATIONS FACILITY

DATA BASE FACILITY

DATA LANGUAGE/I

- CREATION, MAINTENANCE, AND GROWTH
- VARIABLE LENGTH INFORMATION
- DATA AND PROGRAM INDEPENDENCE
- BATCH AND ONLINE CONCURRENTLY OR INDEPENDENTLY
- SUPPORTS HIGH LEVEL LANGUAGES

DATA COMMUNICATIONS FACILITIES

1. BATCH TO ONLINE
   SIMPLIFIES CONVERSION

2. HIGH LEVEL LANGUAGES
   COBOL AND PL/I

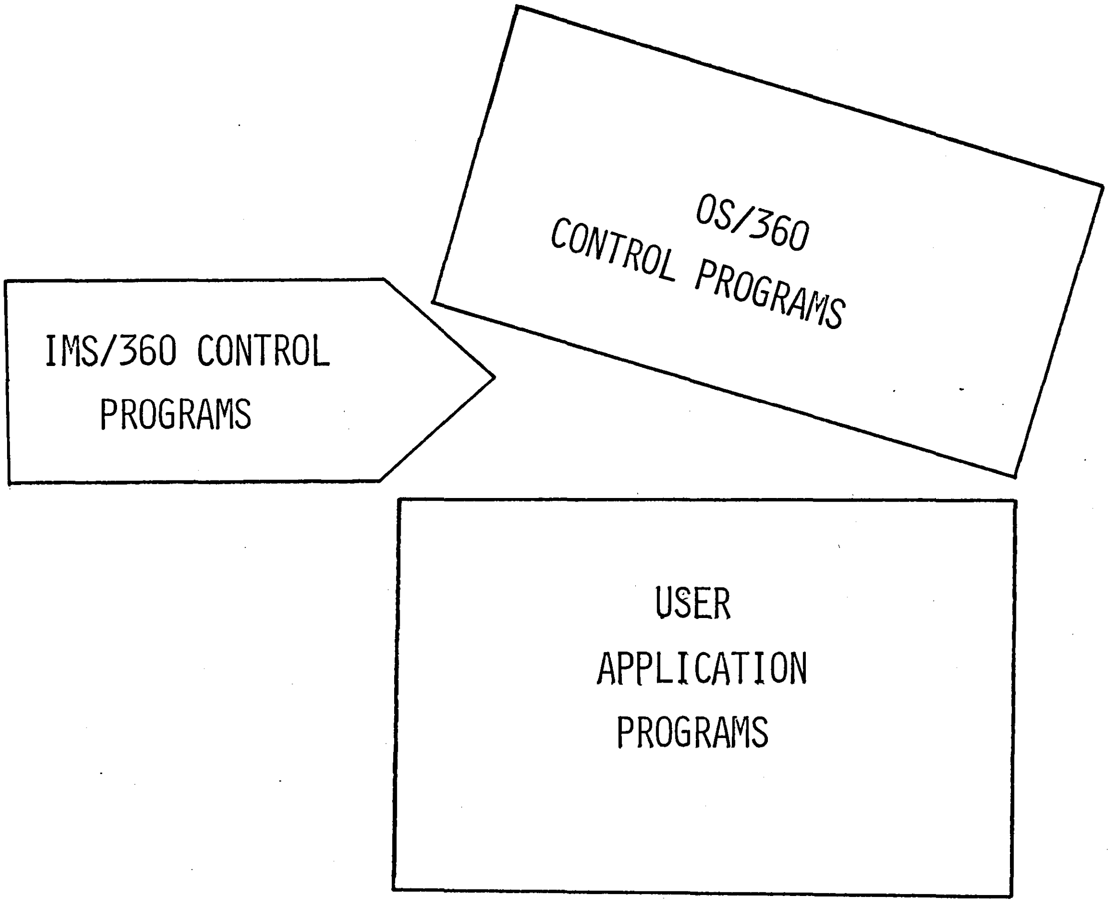
3. CHECKPOINT AND RESTART
   EXTENSIVE DATA BASE RECOVERY

4. SECURITY
   FOR MESSAGE ENTRY

5. LOGICAL TERMINAL OPERATION
   FOR MAINTENANCE AND CHANGE
   MULTIPLE USERS OF ONE PHYSICAL TERMINAL

6. MASTER TERMINAL CONTROL
   SUPPORTS DYNAMIC SYSTEM CHANGES
   CHECKPOINT/RESTART
   OTHER TERMINAL CONTROL

7. STATISTICAL AND ACCOUNTING INFORMATION
   DETAIL REPORTS PROVIDED



A SECOND LEVEL OF CONTROL IS ADDED BETWEEN THE USER APPLICATION PROGRAM AND OS/360 CONTROL PROGRAM

DATA BASE FACILITIES

DATA LANGUAGE/I

LOGICAL DATA STRUCTURES

DATA BASE OPERATION

PHYSICAL DATA STRUCTURES

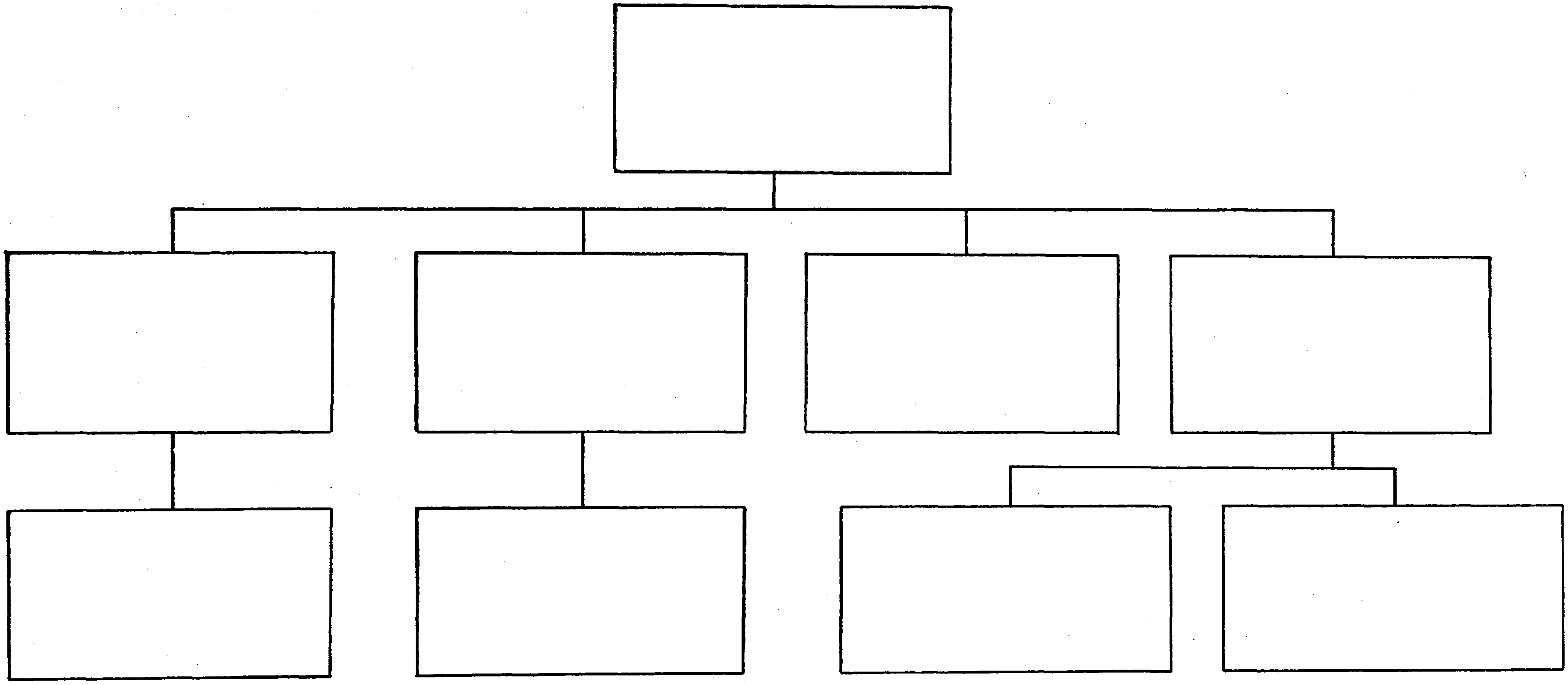
LOGICAL DATA STRUCTURE

HIERARCHICAL

TRADITIONAL FILE STRUCTURE

EMPLOYEE	SECURITY	AUTHORIZATION	EDUCATION	RELATED EXPERIENCE	PREVIOUS POSITION	PAYROLL	CREDIT UNION	BANK DEPOSIT
BANK CUSTOMER	DEMAND DEPOSIT	STATEMENT INFO.	LOAN	PAYMENT HISTORY	ADDRESS	TRUST	TRUST BALANCES	TRUST TRANS. HISTORY
DRAWING MASTER	CONFIGURATION	NEXT ASSEMBLY	FABRICATION PART	FABRICATION QUANTITY	DESCRIPTION	ENGRG. ORDER HISTORY	FABRICATION HISTORY	CONFIGURATION HISTORY

LOGICAL DATA STRUCTURE



EMPLOYEE

SECURITY EDUCATION PREVIOUS POSIT, PAYROLL

AUTHORIZATION RELATED EXPERIENCE CREDIT UNION BANK DEPOSIT

CUSTOMER

DEMAND DEPOSIT

LOAN

ADDRESS

TRUST

STATEMENT HISTORY

PAYMENT HISTORY

TRUST BALANCE

TRANSACTION HISTORY

DWGMSTR

CONFIG FABPART DESCRIPT EOHIST

NEXTASBY FABQTY FABHIST CONHIST

A DATA BASE CONSISTS OF 1 TO N DATA BASE RECORDS

A DATA BASE RECORD CONSISTS OF 1 TO N SEGMENTS

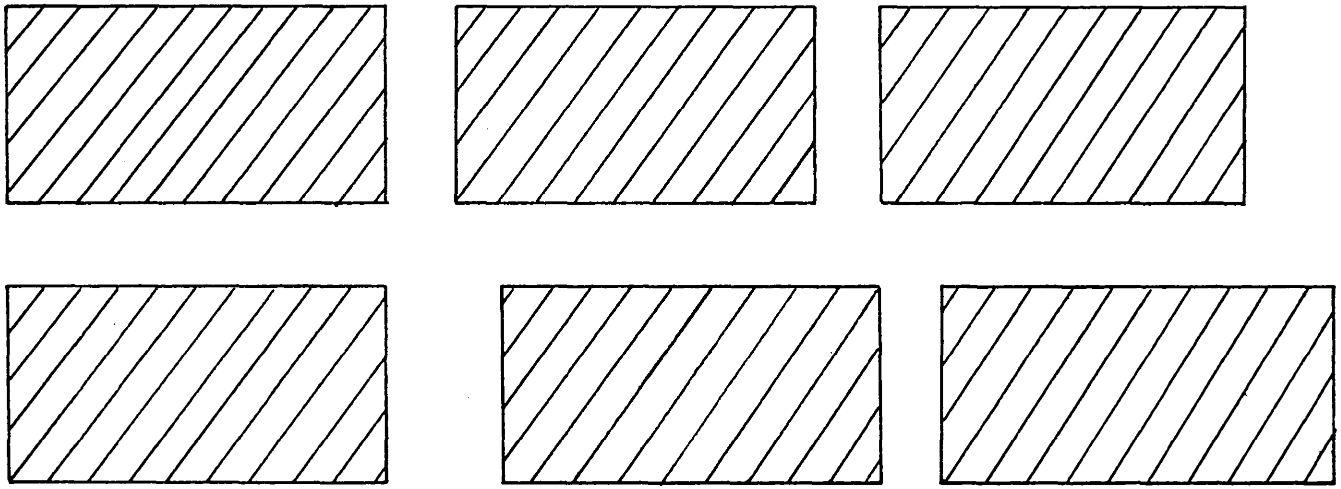
MAXIMUM OF 255 SEGMENT NAMES

MAXIMUM OF 15 SEGMENT LEVELS

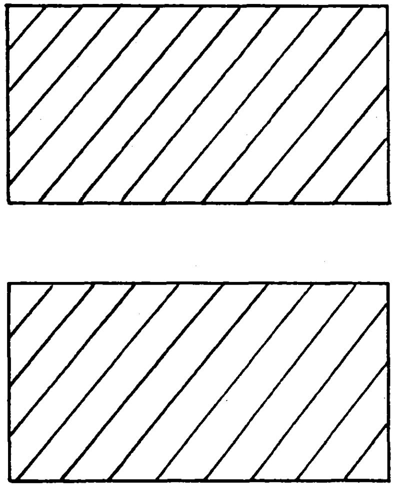
1 ROOT SEGMENT PER DATA BASE RECORD

DEPENDENT SEGMENTS -- 0 TO N PER PARENT

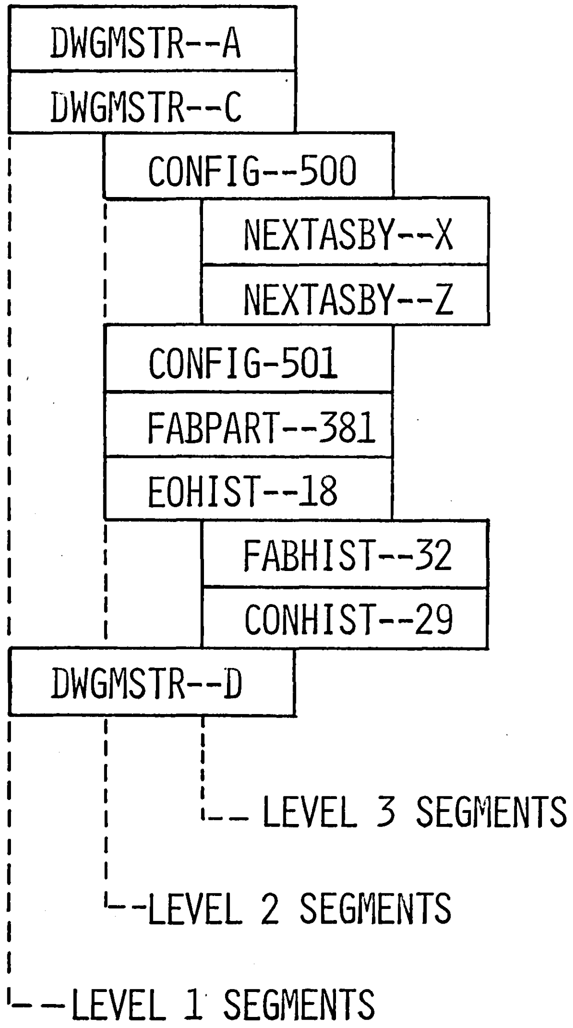
SENSITIVITY OF USER ONE



SENSITIVITY OF USER TWO



ORDER OF PHYSICAL STORAGE



DATA BASE OPERATION

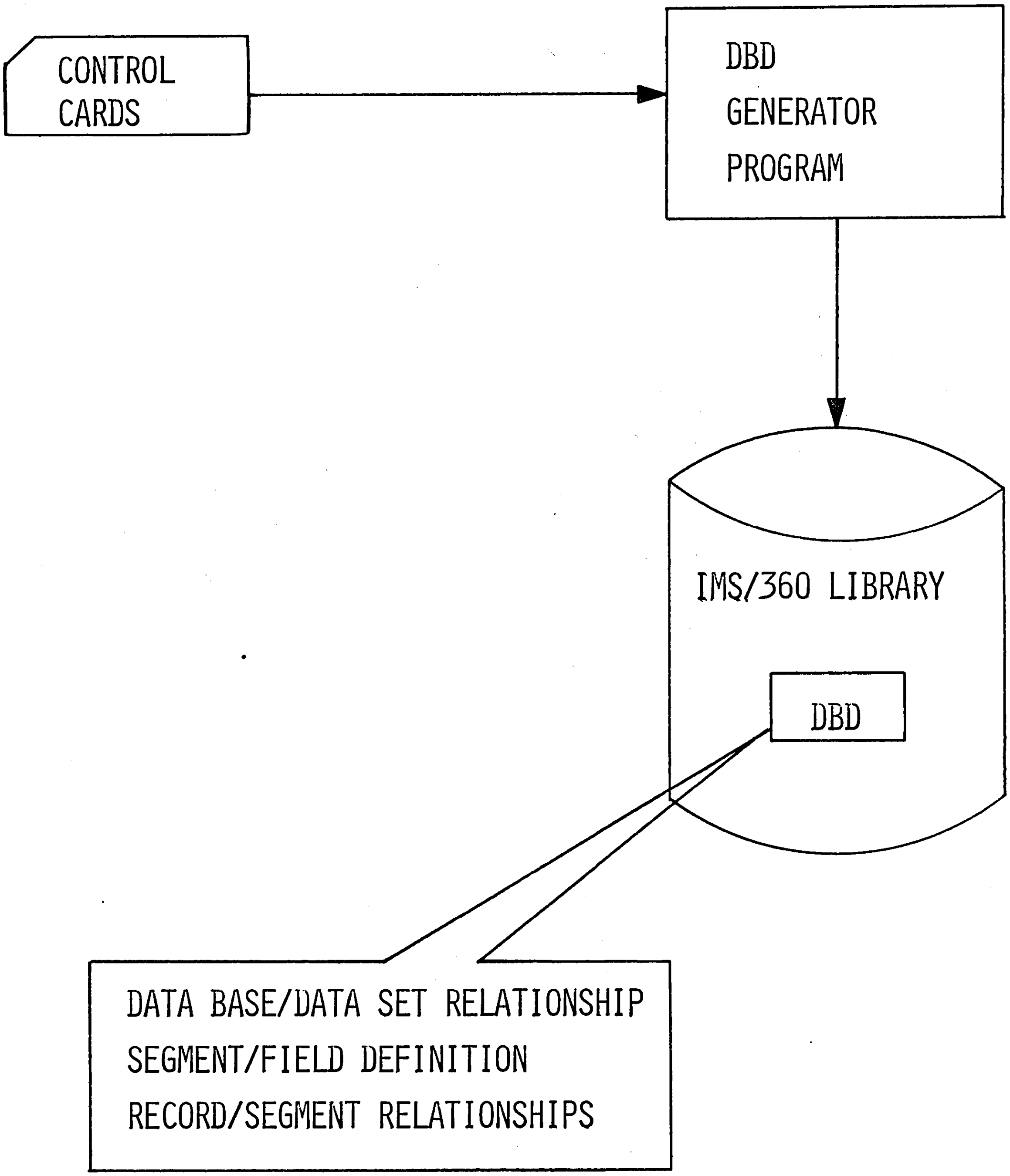
DATA BASE DESCRIPTION

PROGRAM DESCRIPTION

DATA BASE CREATION

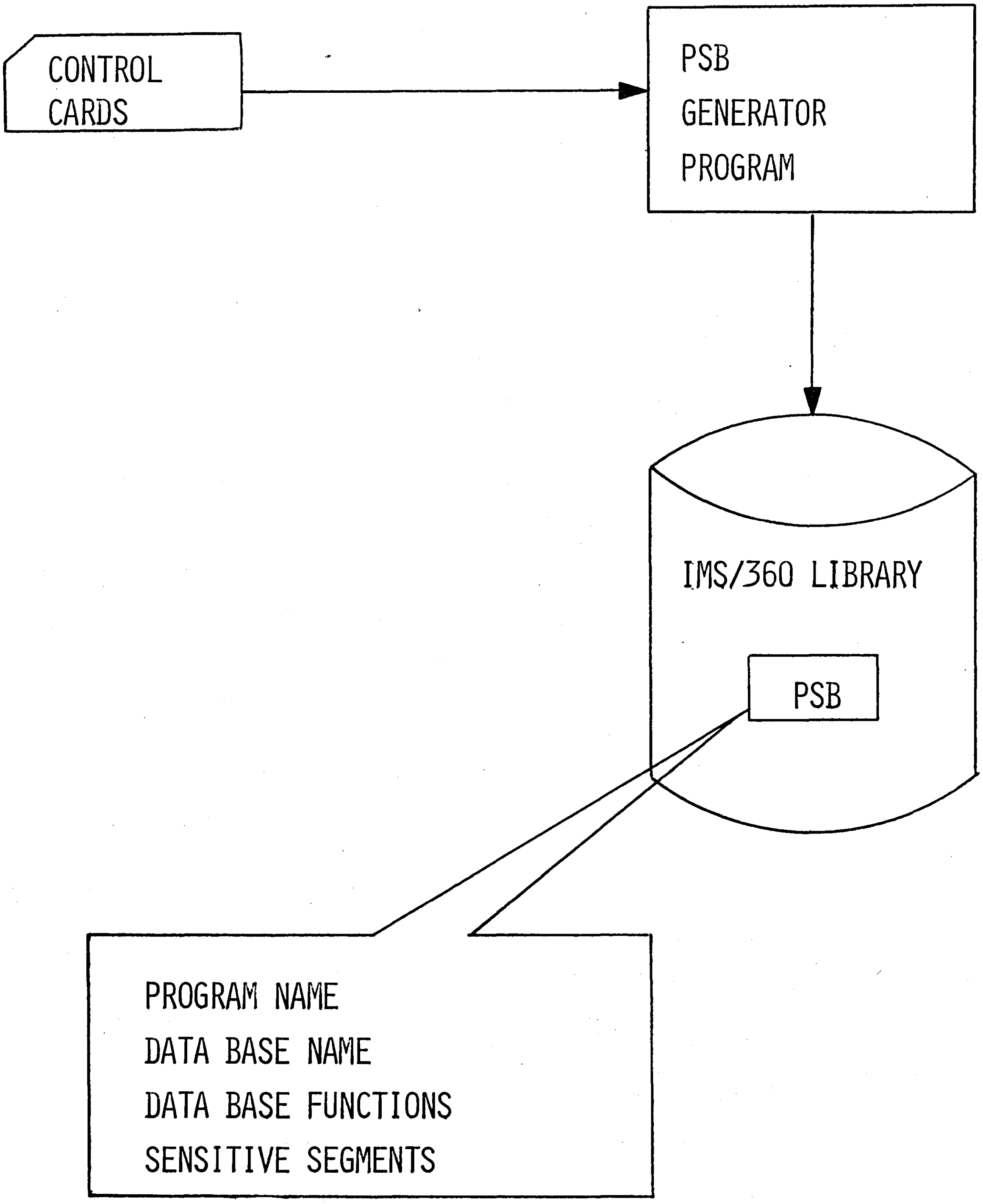
DATA BASE PROCESSING

DATA BASE DESCRIPTION

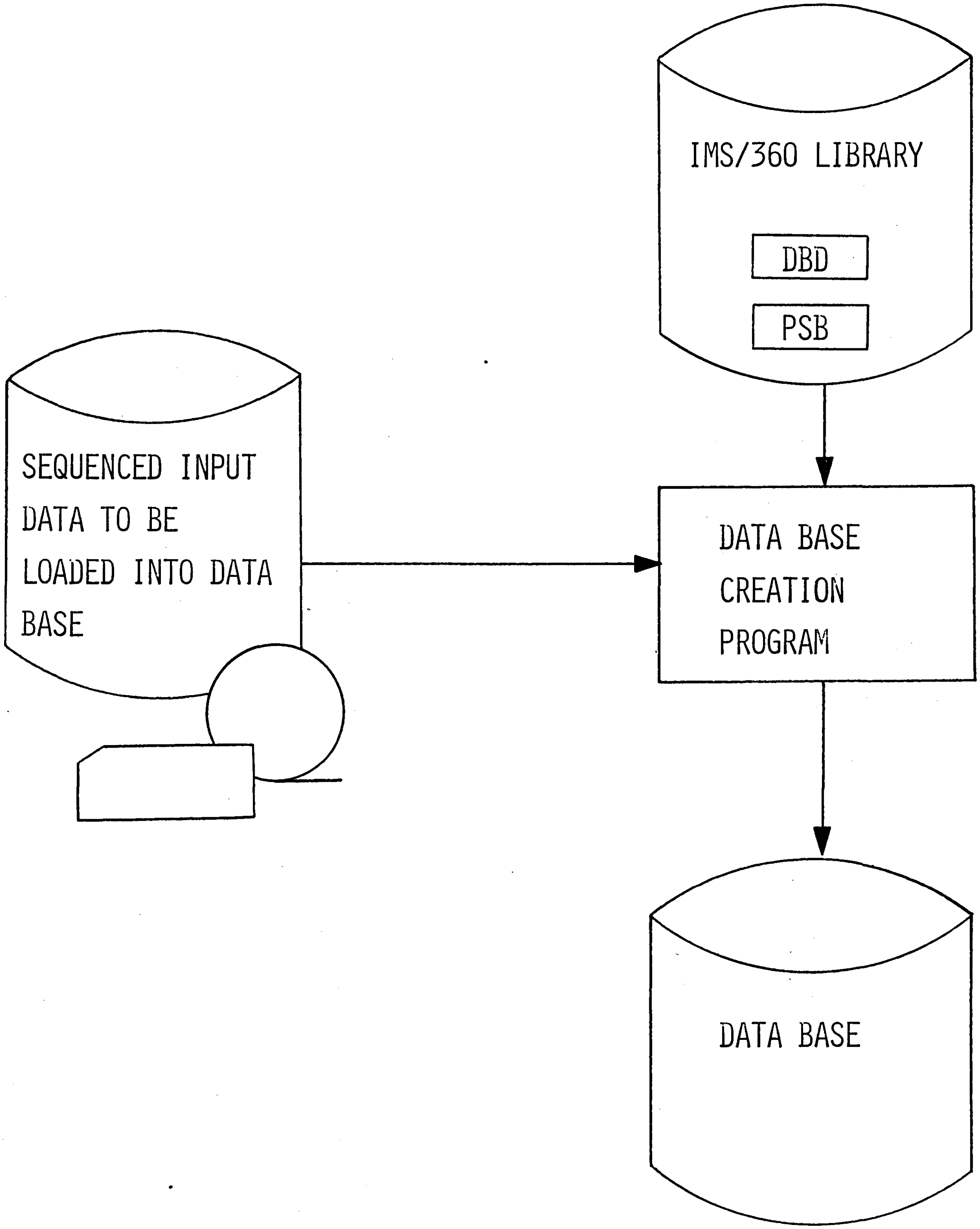


DATA BASE/DATA SET RELATIONSHIP
SEGMENT/FIELD DEFINITION
RECORD/SEGMENT RELATIONSHIPS

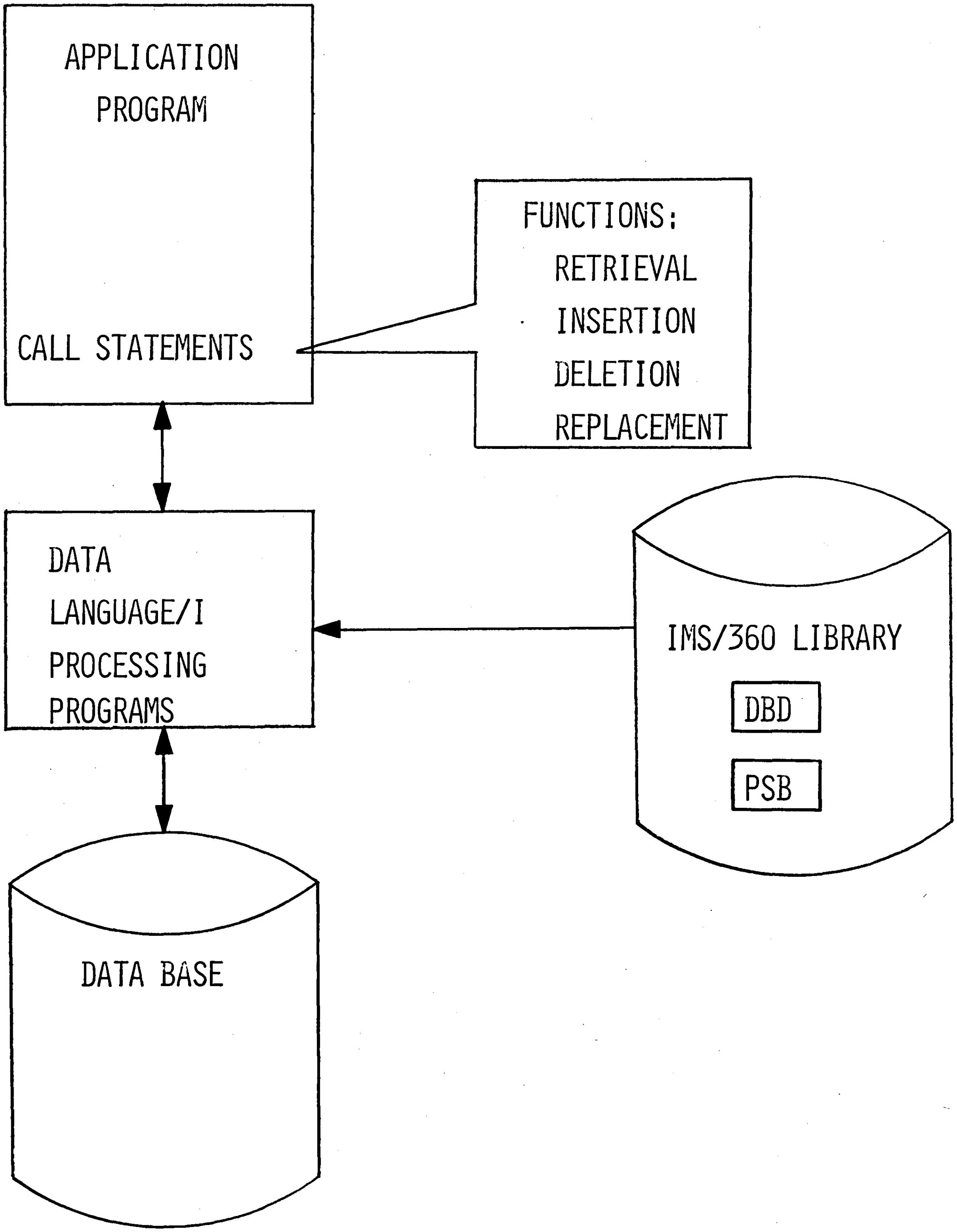
PROGRAM DESCRIPTION



DATA BASE CREATION


IMS/360 LIBRARY
DBD
PSB
SEQUENCED INPUT DATA TO BE LOADED INTO DATA BASE
DATA BASE CREATION PROGRAM
DATA BASE

DATA BASE PROCESSING



CALL STATEMENTS

FUNCTIONS (AT THE SEGMENT LEVEL)

- GET UNIQUE (HOLD)
- GET NEXT (HOLD)
- GET NEXT WITHIN PARENT (HOLD)
- INSERT
- DELETE
- REPLACE

SEGMENT SEARCH ARGUMENTS

- SEGNAME (KEY FIELD NAME=FIELD VALUE)

DATA LANGUAGE/I CALLS

```
GU ROOT (KEYFLD=0816)
   DEP1 (KEYFLD1=276)
```

```
ISRT ROOT (KEYFLD=0912)
     DEP1 (KEYFLD1=314)
     DEP2
```

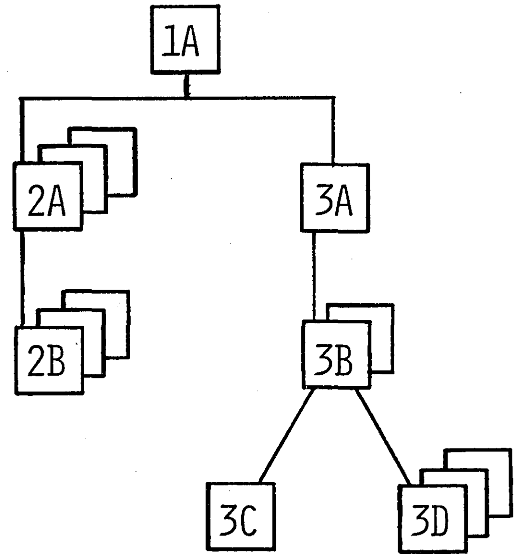
PHYSICAL DATA STRUCTURES

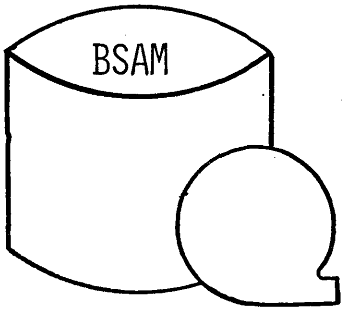
HSAM

HISAM

HSAM

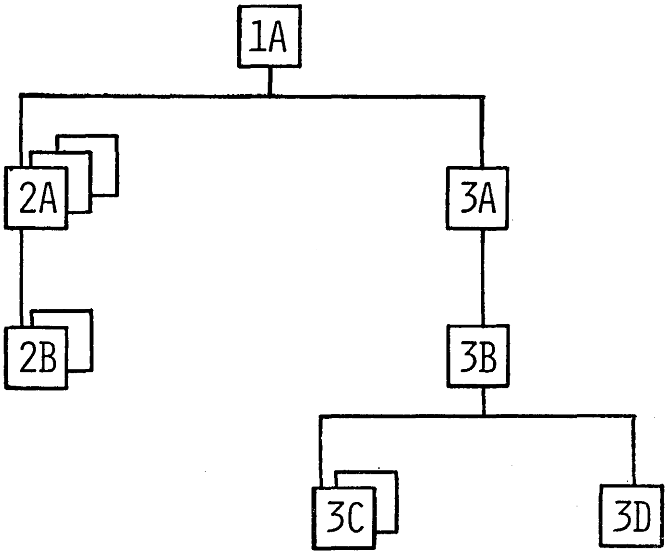
DATA BASE
RCD STRUCTURE

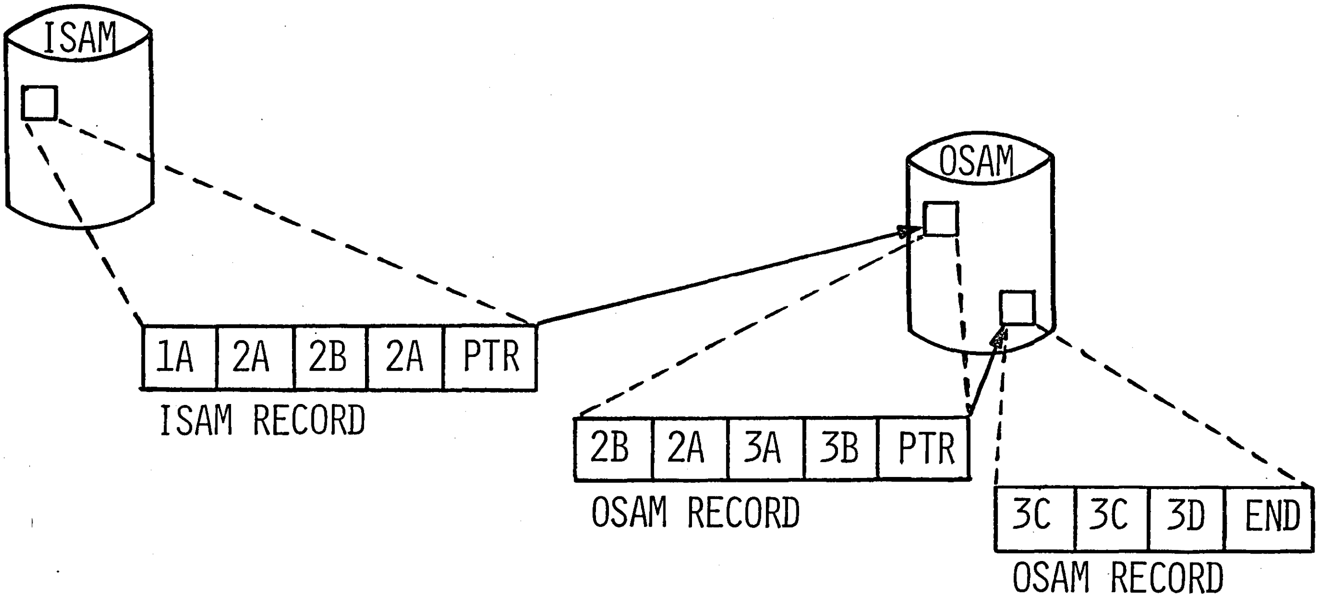




BSAM RCD, NO, 1	1A	2A	2B	2A	2B	2B
BSAM RCD, NO, 2	2A	3A	3B	3C	3D	3B
BSAM RCD, NO, 3	3D	3D	1A	2A	2B	2B

HISAM - SINGLE DATA SET GROUP

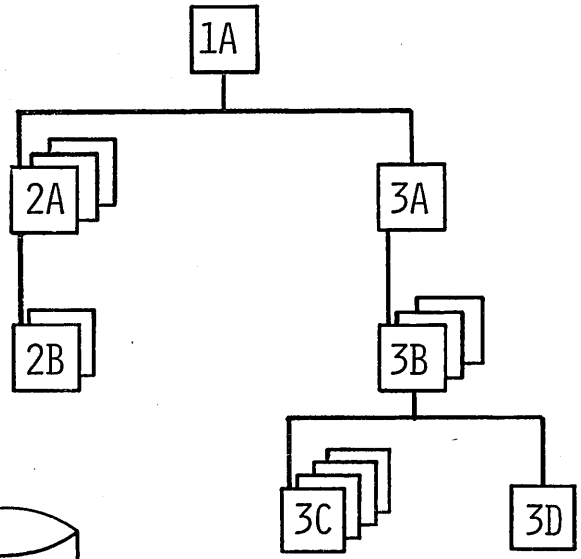


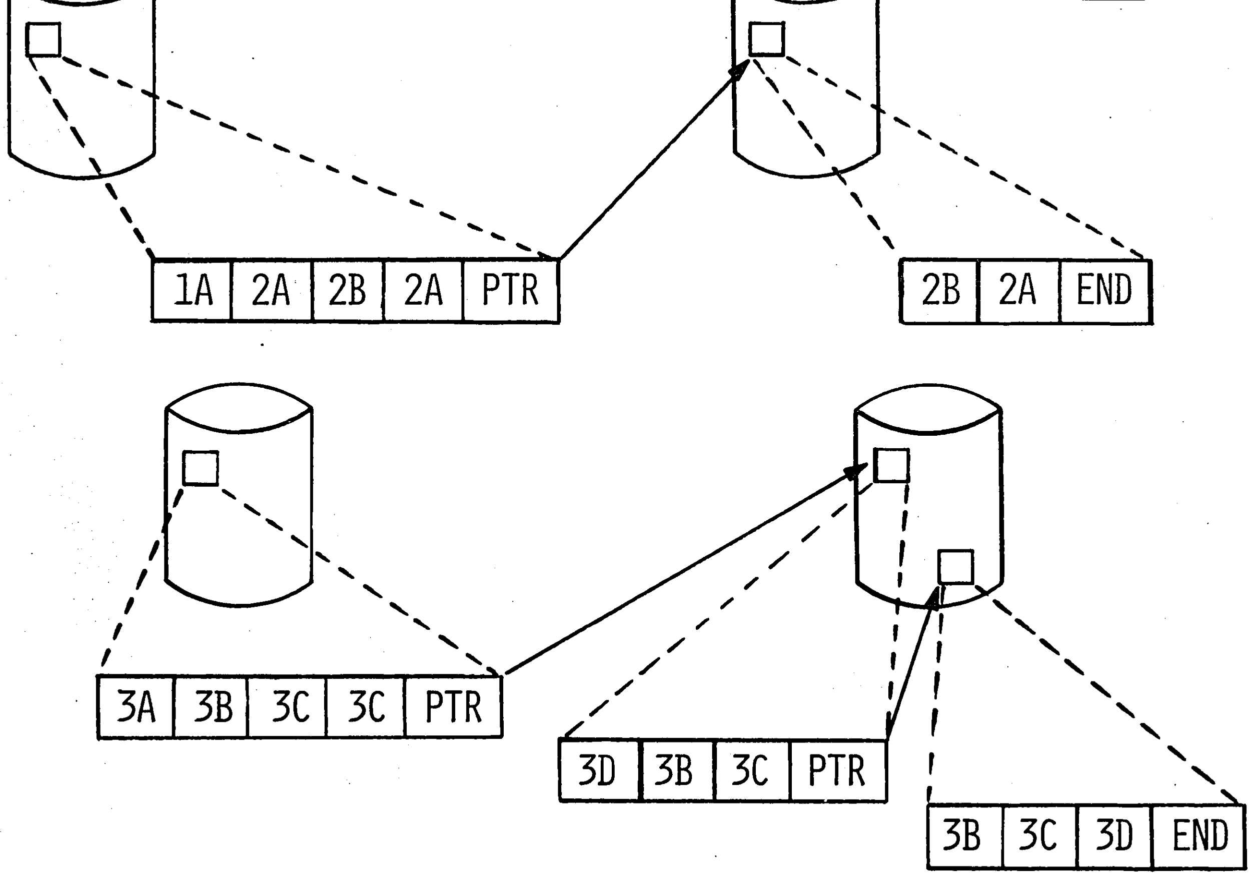


HISAM - MULTIPLE DATA SET GROUP

TWO DATA SET GROUPS:

1. PRIMARY DATA SET GROUP
   1A, 2A, 2B SEGMENTS
2. SECONDARY DATA SET GROUP
   3A, 3B, 3C, 3D SEGMENTS





DATA COMMUNICATION FACILITIES

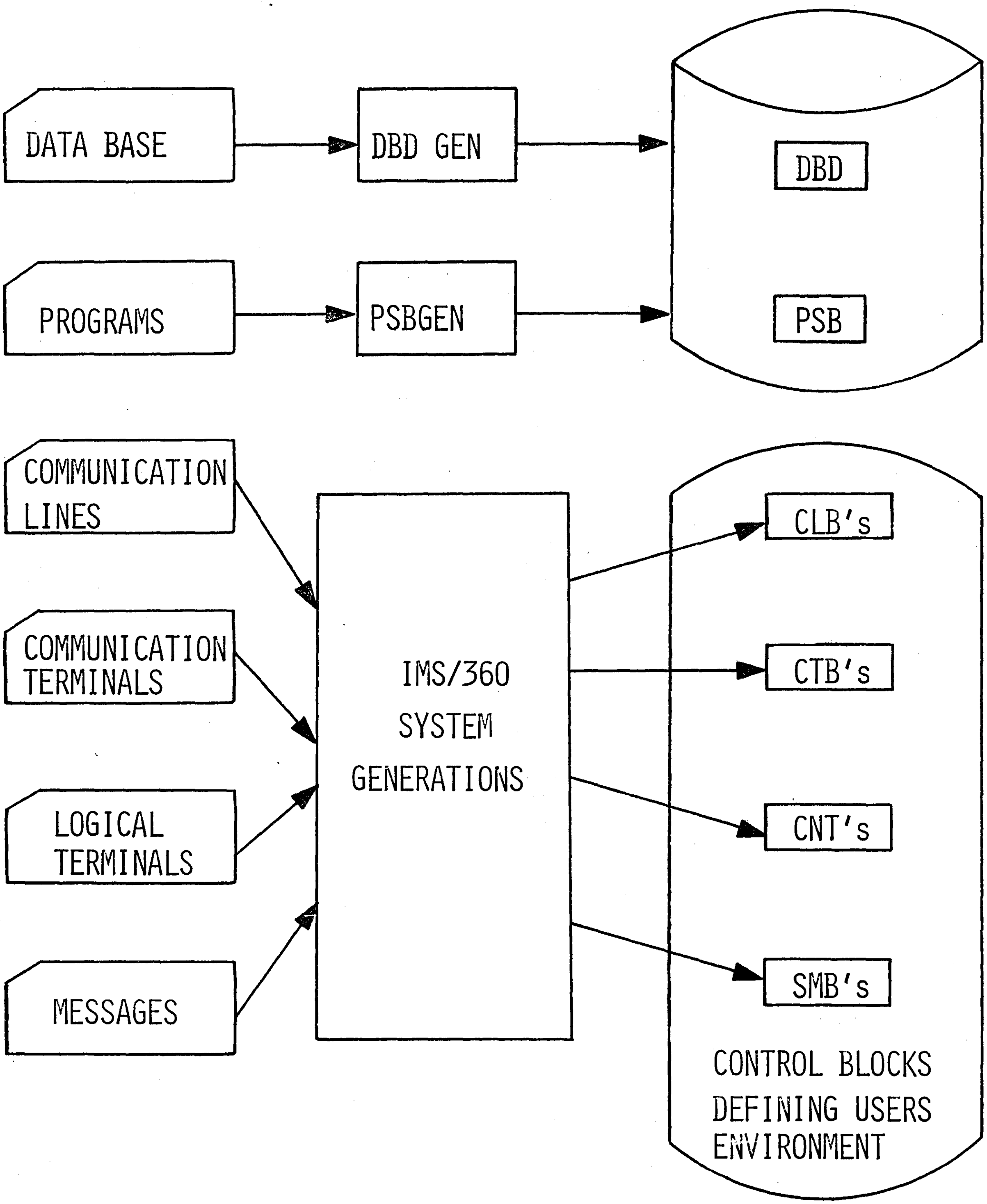
PREDEFINITION

SECURITY AND PRIORITY

TERMINAL COMMAND LANGUAGE

RECOVERY

PREDEFINITION



INPUT FORMAT

TRANSACTION CODE		SECURITY CODE		TEXT

EXAMPLES OF TRANSACTIONS

TRANSACTION: SALARY (PASSWD) 281635

RESPONSE: SALARY FOR A.J. SMITH EMP. NO. 281635 is $1,000

TRANSACTION: INV 84312

RESPONSE: PART NUMBER 84312 HAS A TOTAL OF 261 UNITS

TRANSACTION: DDNO C.A. BRAWLEY

RESPONSE: C.A. BRAWLEY'S DEMAND DEPOSIT ACCOUNT
NO. 15 285 16 8401

MESSAGE PROCESSING PRIORITY SCHEME

15 PRIORITY LEVELS

	1	2	3
NORMAL PRIORITY	③	3	③
LIMIT PRIORITY	12	⑫	12
LIMIT COUNT	10	10	10
QUEUE COUNT	4	10	0

MASTER TERMINAL COMMAND LANGUAGE

/ASSIGN

/CHANGE

/CHECKPOINT

/DBDUMP

/DBLOG

/DBNOLOG

/DBRECOVERY

/DISPLAY

/ERESTART

/NRESTART

/PSTOP

/PURGE

/START

/STOP

/DELETE

REMOTE TERMINAL COMMAND LANGUAGE

/BROADCAST

/EXCLUSIVE

/END

/LOCK

/UNLOCK

/TEST

/IAM

/SET

/RESET

RECOVERY

IMS/360 CONTROL BLOCKS

QUEUES

DATA BASES

SECURITY MATRICES

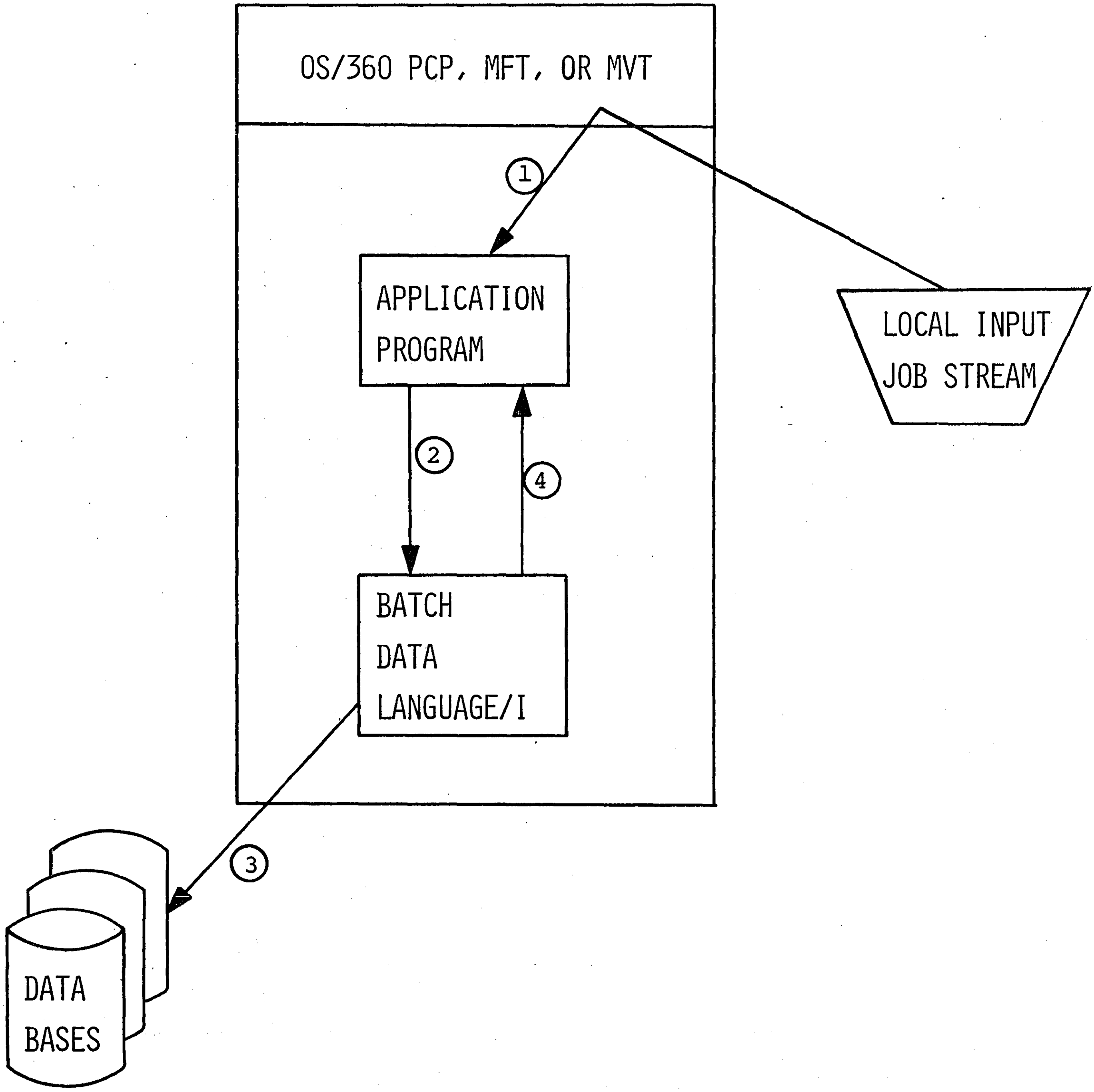
TRANSACTION CODE

		PAYINQ	RECREP	INVINQ	ABCXYZ	XYZ123
SOURCE TERMINAL	REC101	0	1	1	0	1
	SAL102	1	1	0	0	0
	INV109	0	0	1	1	0
	PDXIMQ	1	0	0	0	0

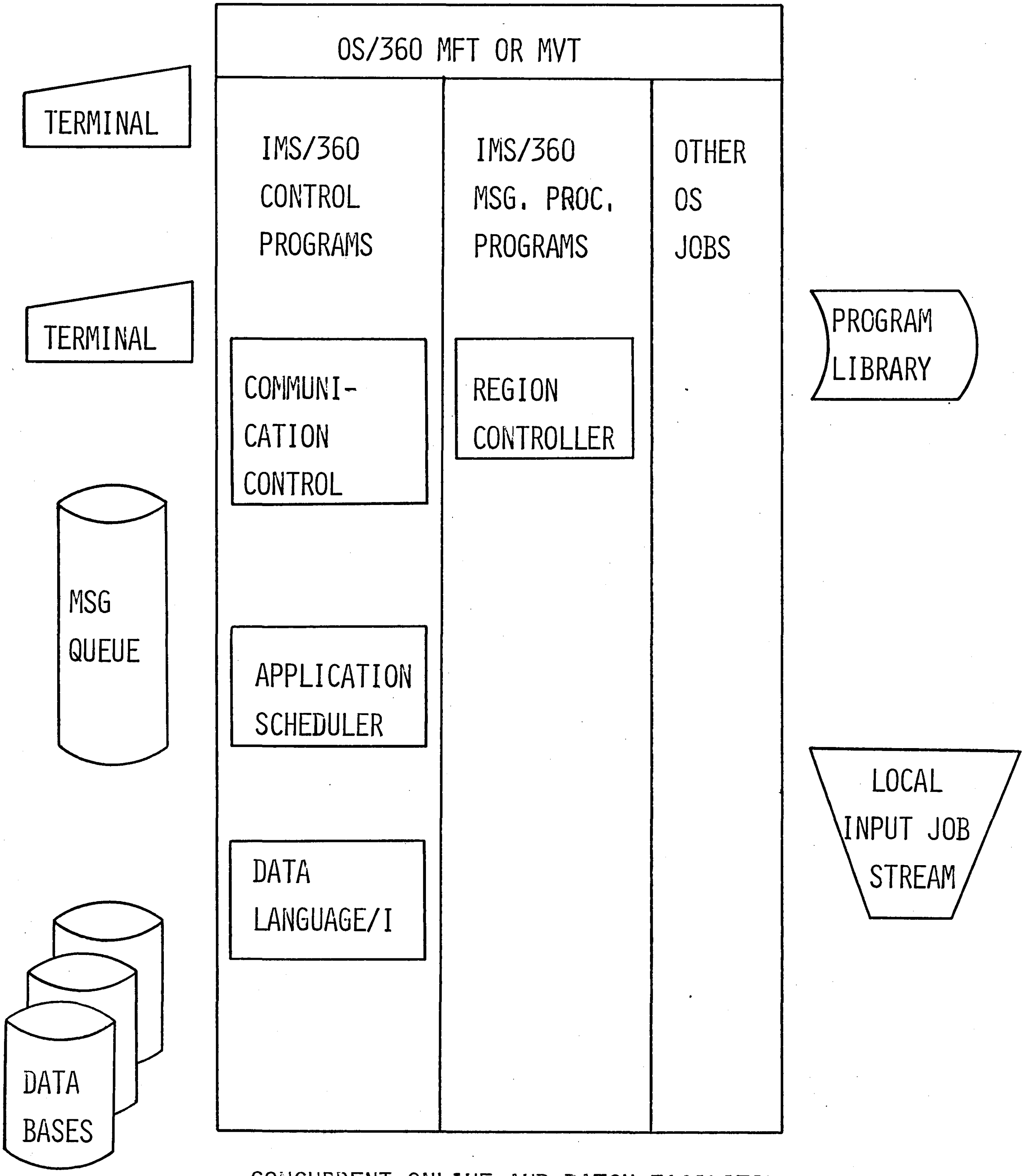
TRANSACTION CODE

		PAYINQ	XYZZ44	A123456	ABCXYZ	STOPPAY
PASSWORD	ABCXYZ12	1	0	1	0	0
	PMX12122	1	1	1	1	1
	ABOOXYOZ	0	0	0	1	0

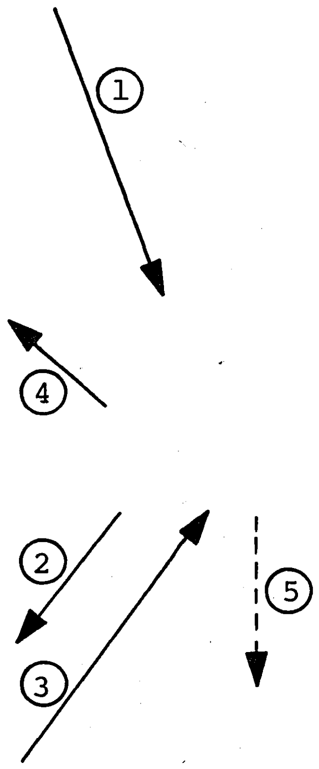
BATCH ONLY PROCESSING

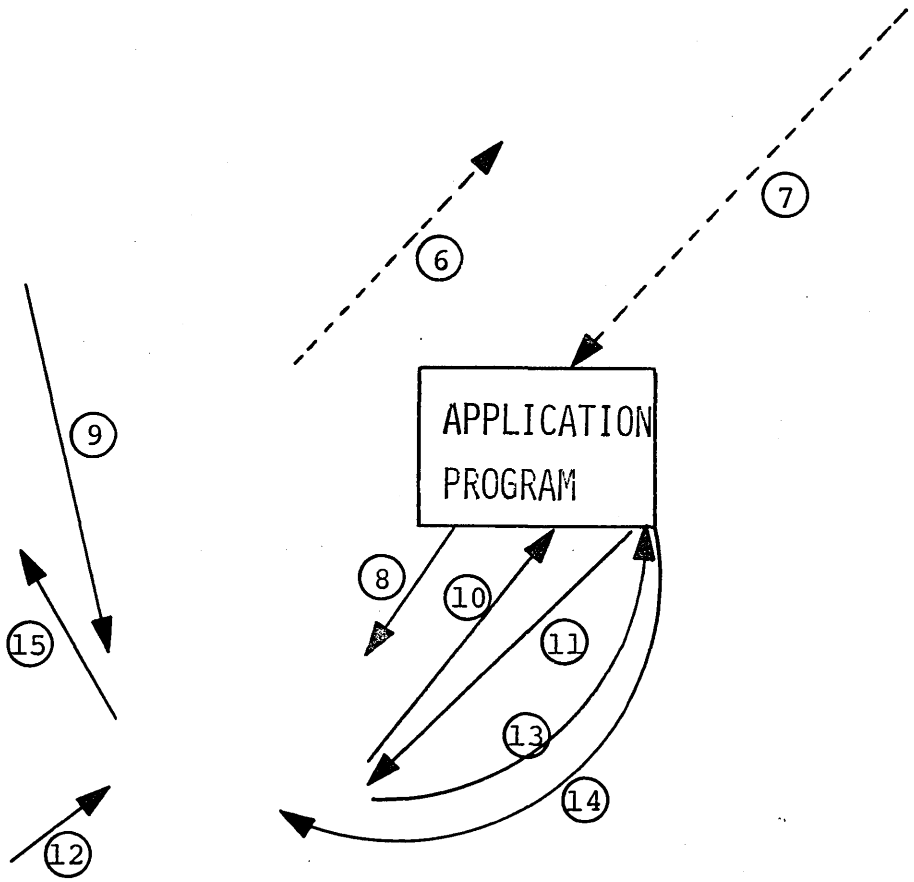


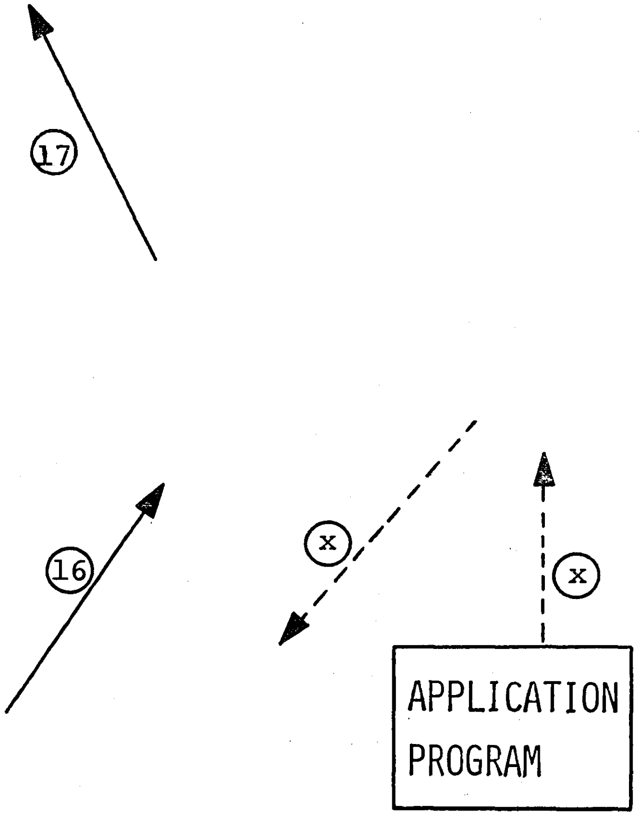
IMS/360 SYSTEM ORGANIZATION

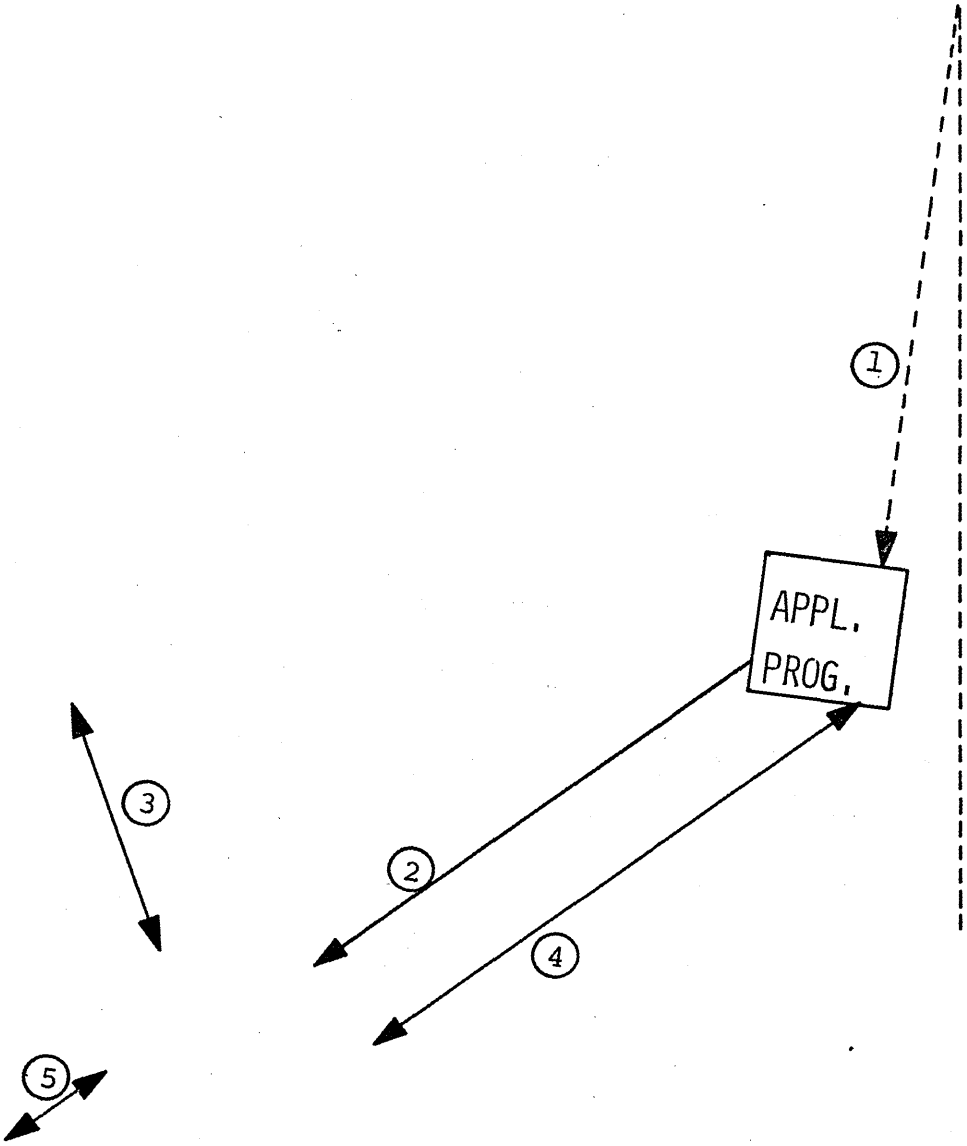


CONCURRENT ONLINE AND BATCH FACILITIES

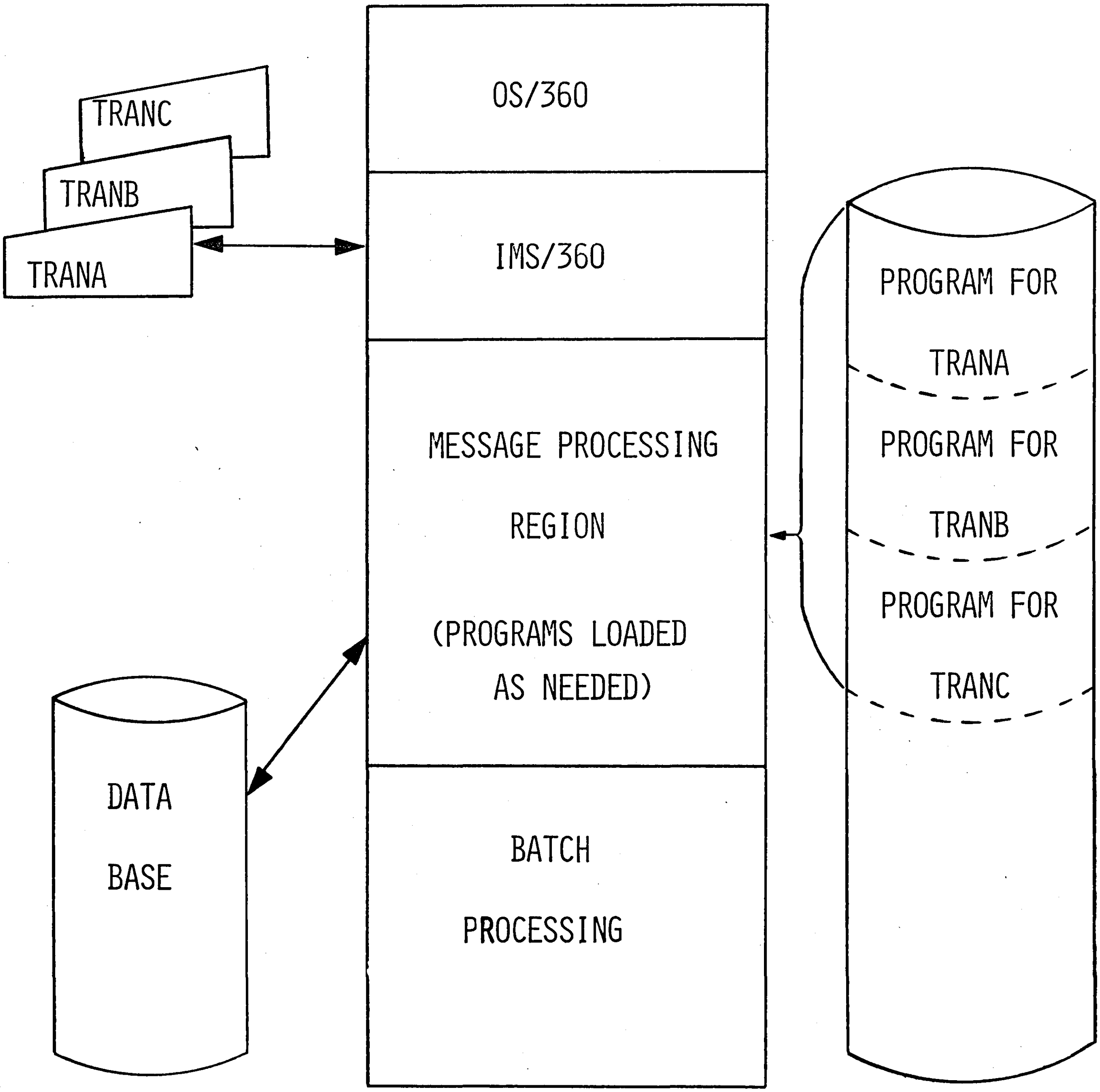

1
4
2
3
5


APPLICATION PROGRAM
6
7
8
9
10
11
12
13
14
15

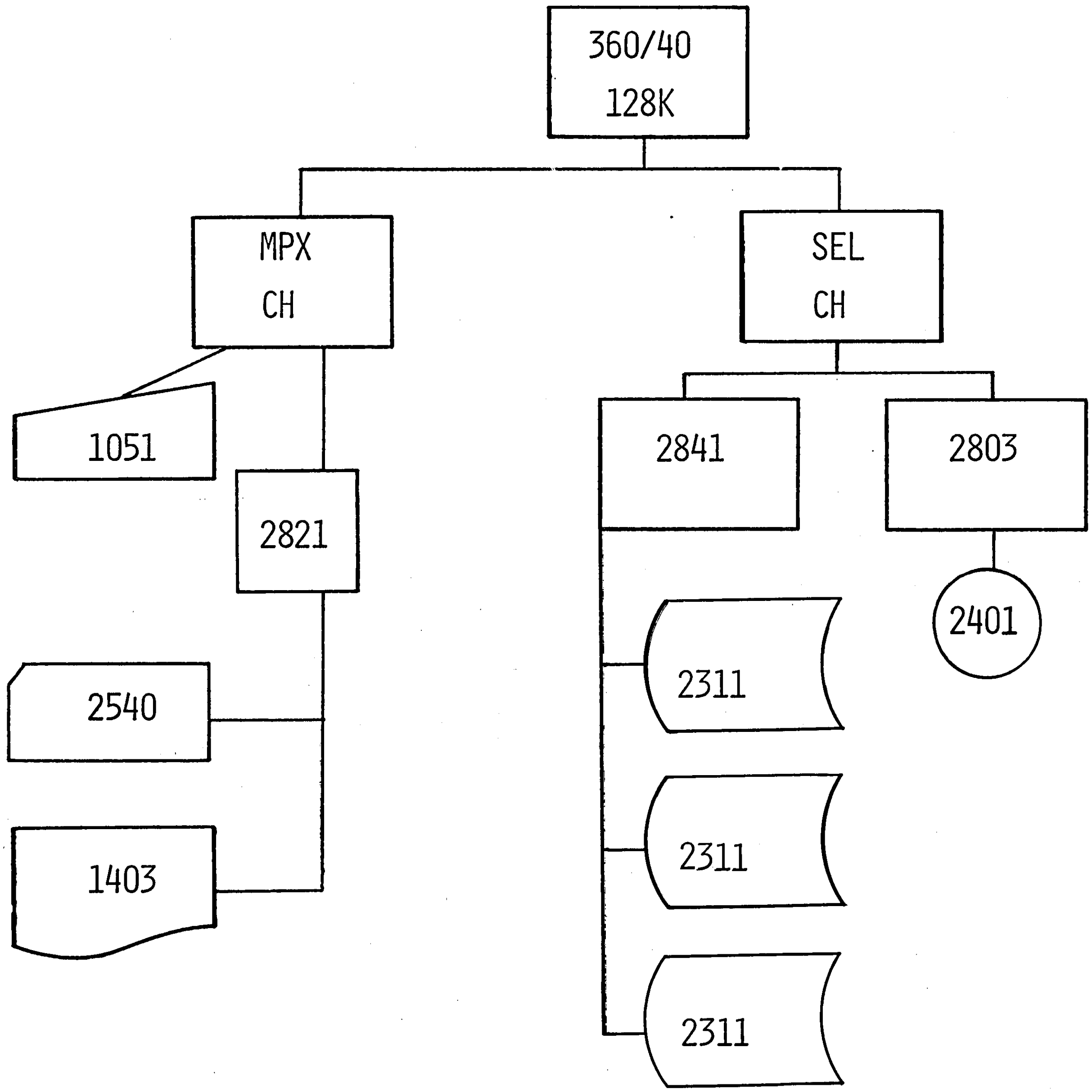

17
16
x
x
APPLICATION PROGRAM


1
APPL.
PROG.
2
3
4
5

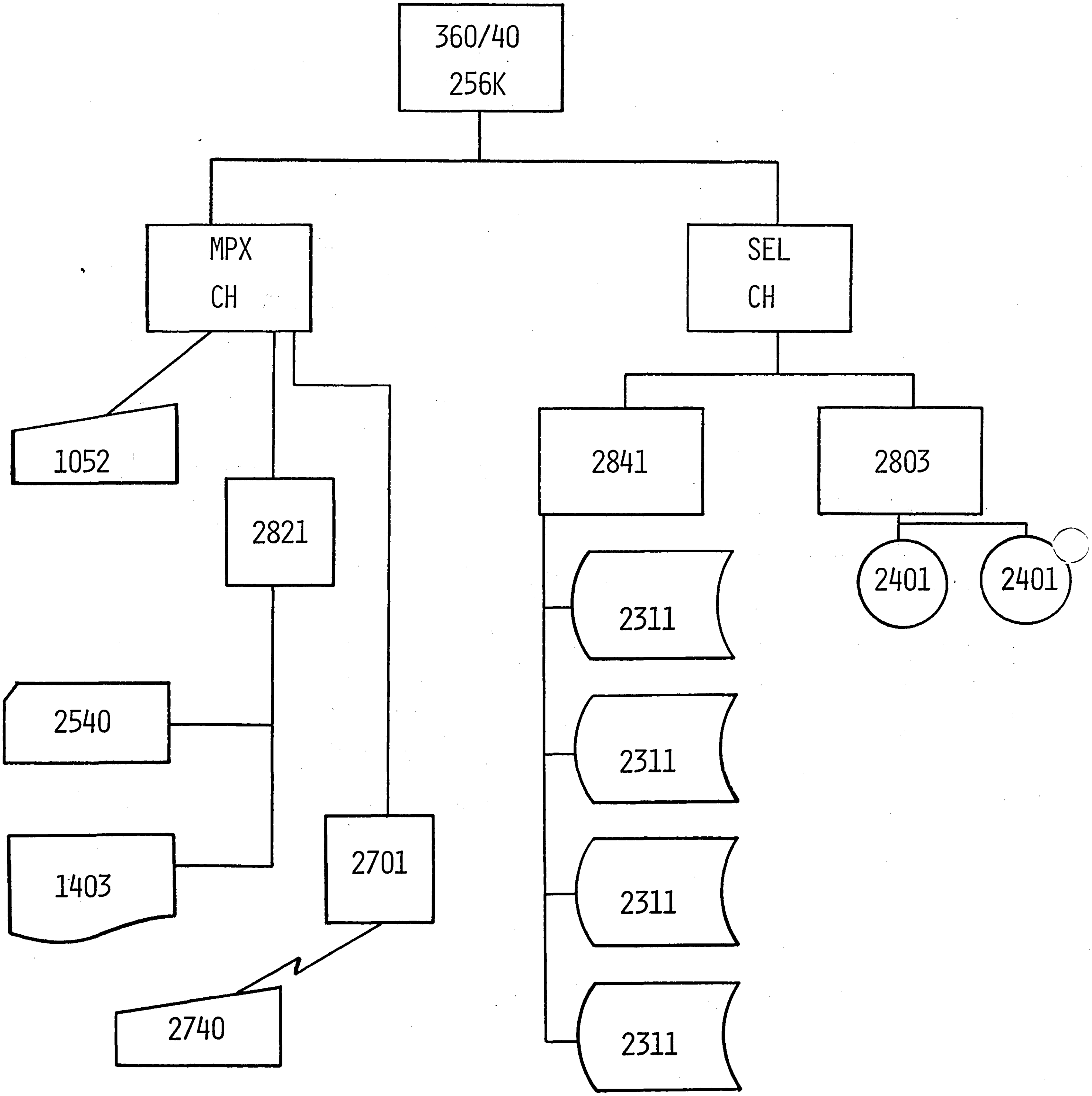
IMS/360 SYSTEM ORGANIZATION

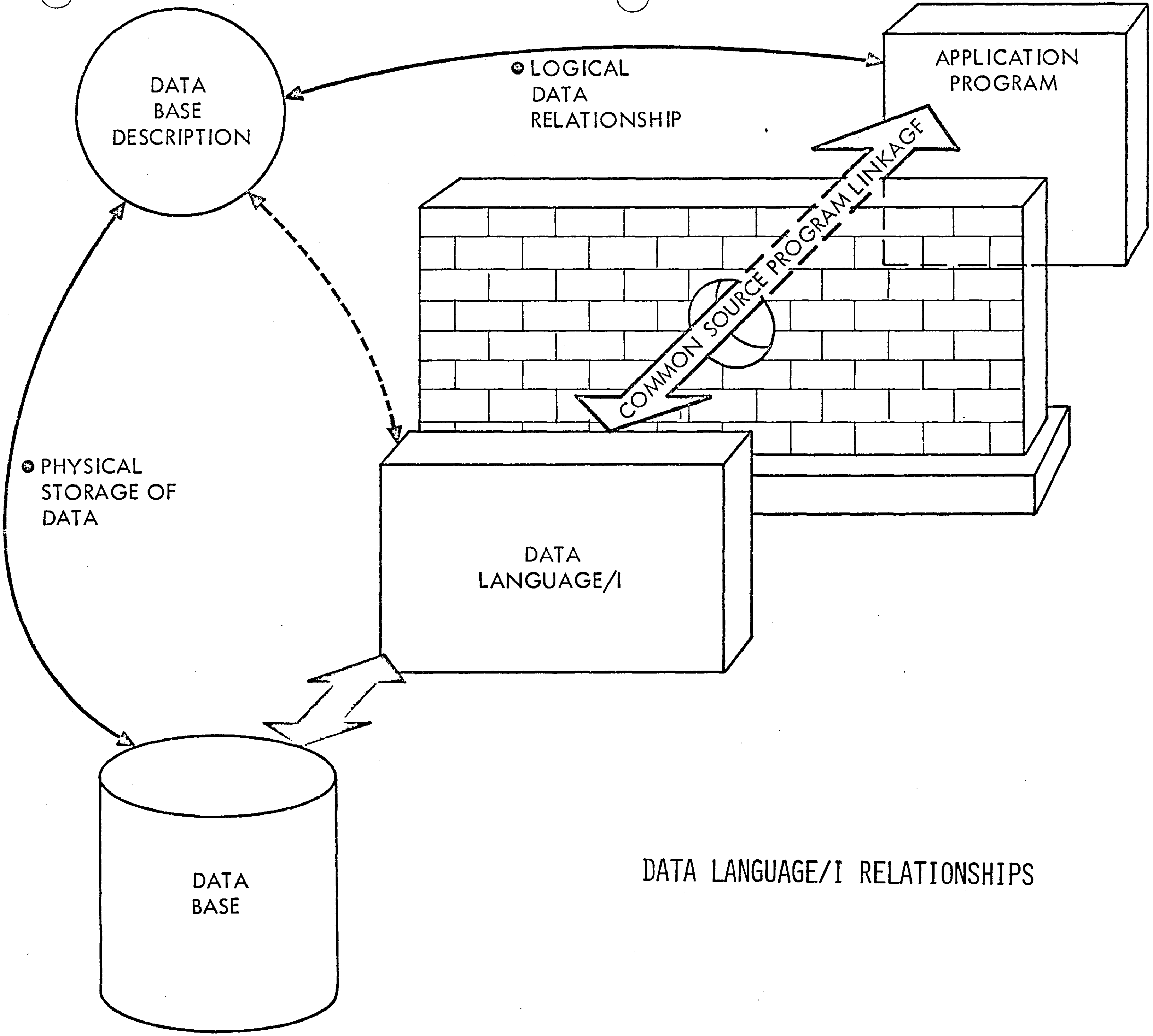


MINIMUM BATCH CONFIGURATION

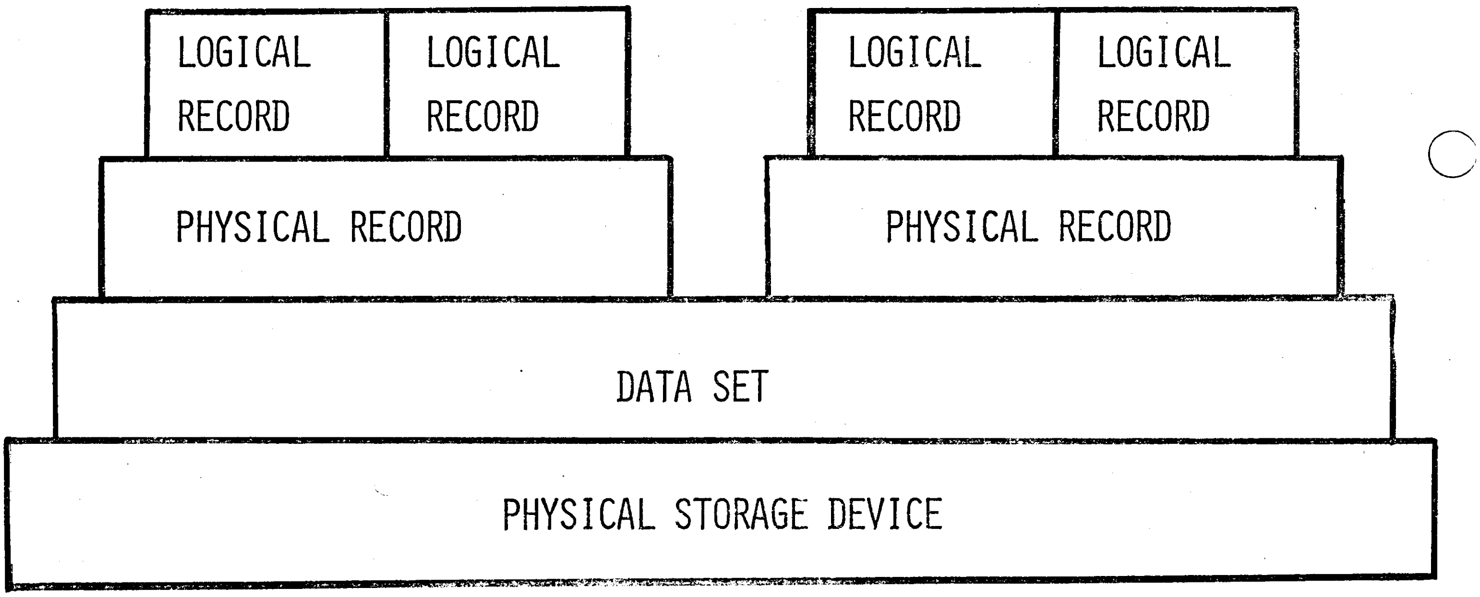


MINIMUM TELEPROCESSING CONFIGURATION

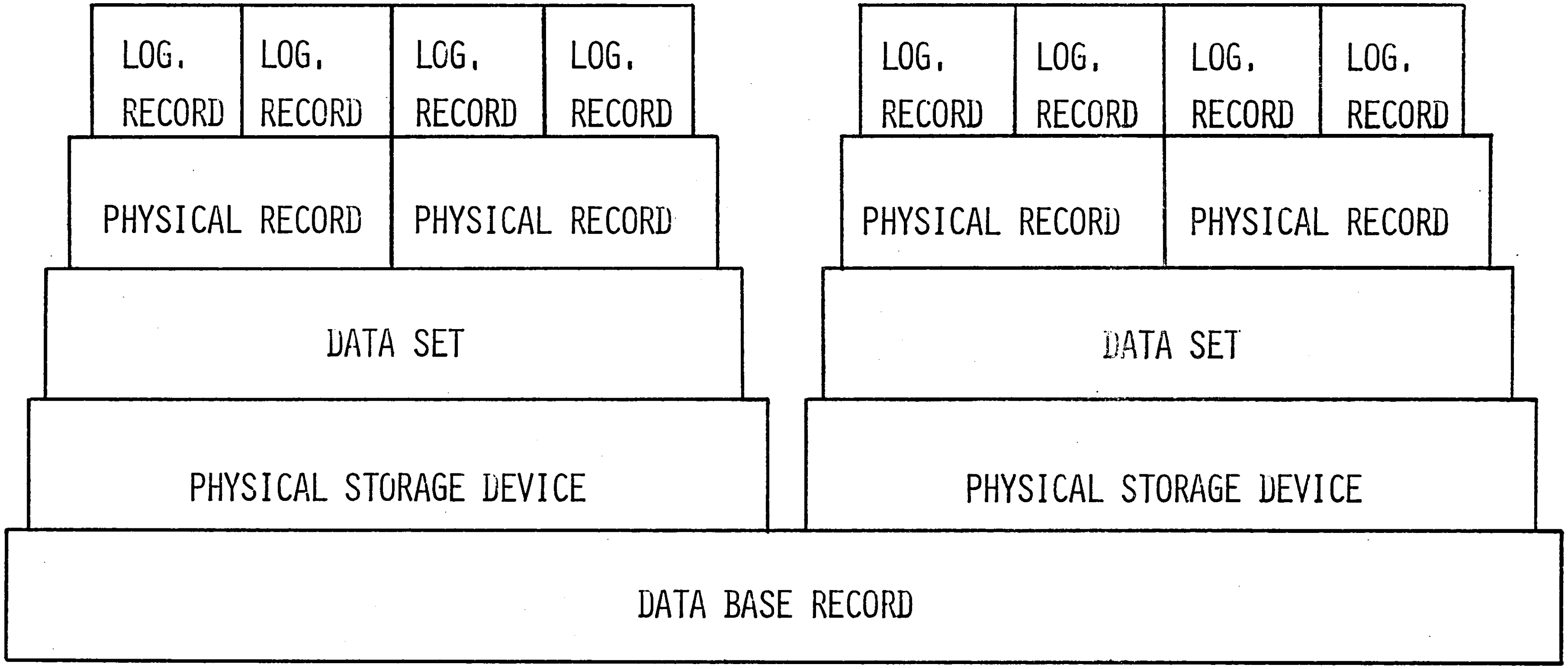



DATA BASE DESCRIPTION
• LOGICAL DATA RELATIONSHIP
APPLICATION PROGRAM
COMMON SOURCE PROGRAM LINKAGF
• PHYSICAL STORAGE OF DATA
DATA LANGUAGE/I
DATA BASE

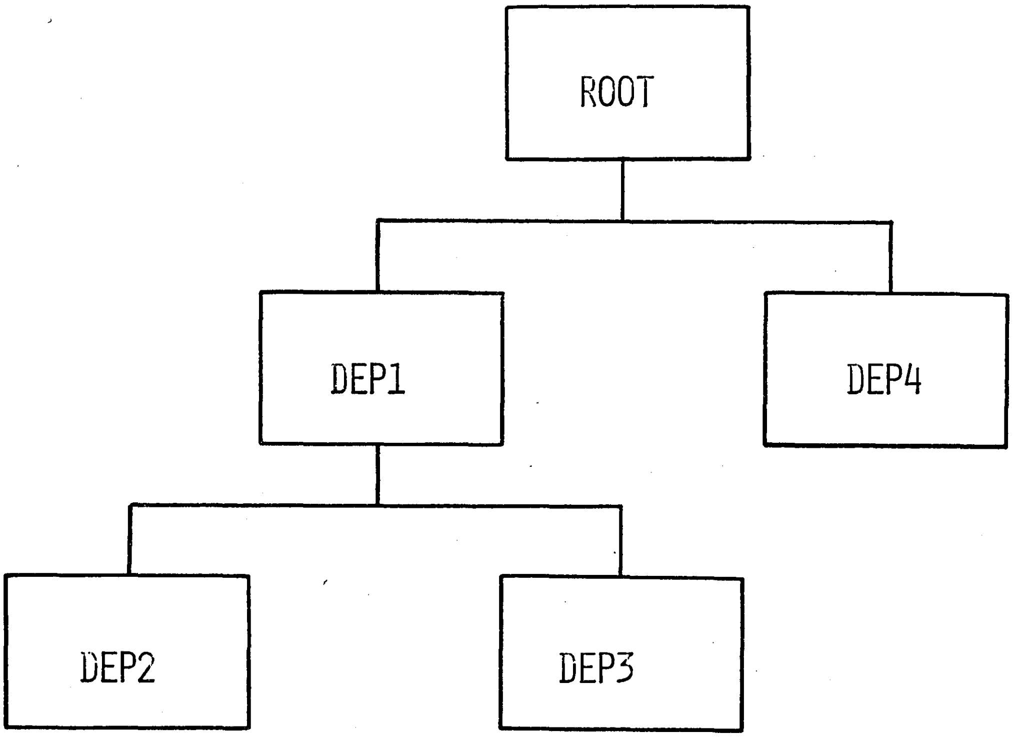
DATA LANGUAGE/I RELATIONSHIPS



OS/360 DATA MANAGEMENT

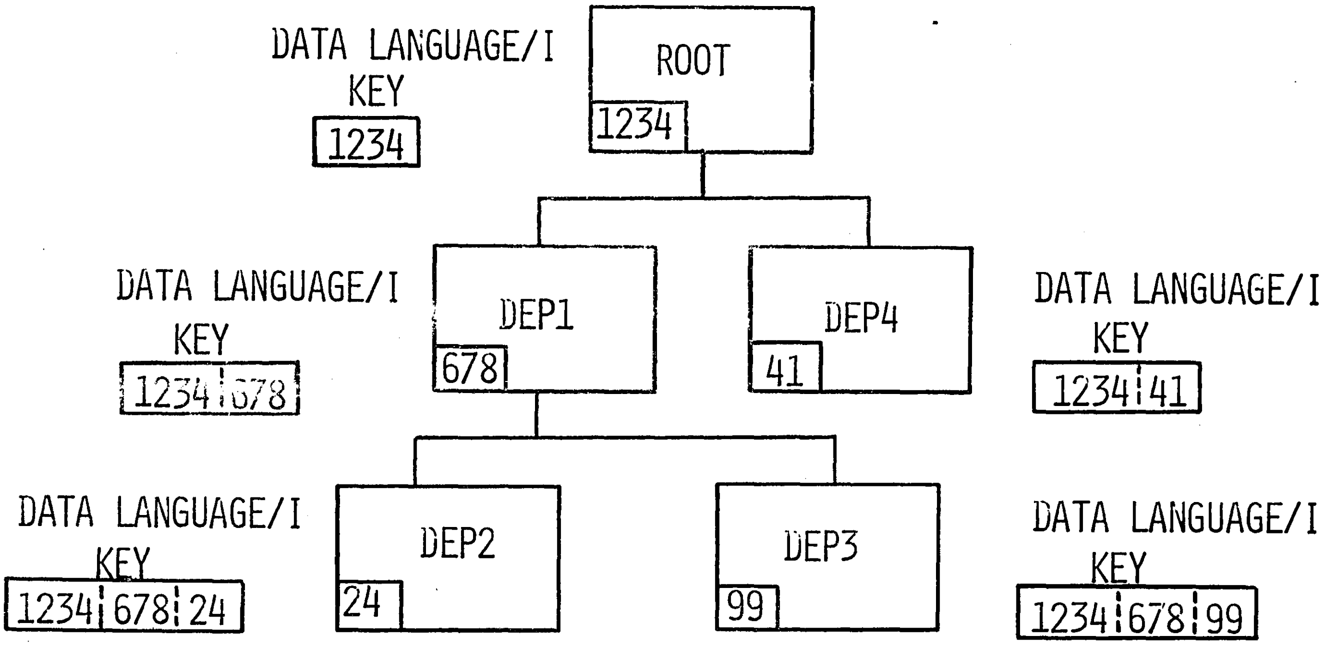

LOG. RECORD
LOG. RECORD
LOG. RECORD
LOG. RECORD
LOG. RECORD
LOG. RECORD
LOG. RECORD
LOG. RECORD
PHYSICAL RECORD
PHYSICAL RECORD
PHYSICAL RECORD
PHYSICAL RECORD
DATA SET
DATA SET
PHYSICAL STORAGE DEVICE
PHYSICAL STORAGE DEVICE
DATA BASE RECORD

HIERARCHICAL STRUCTURE

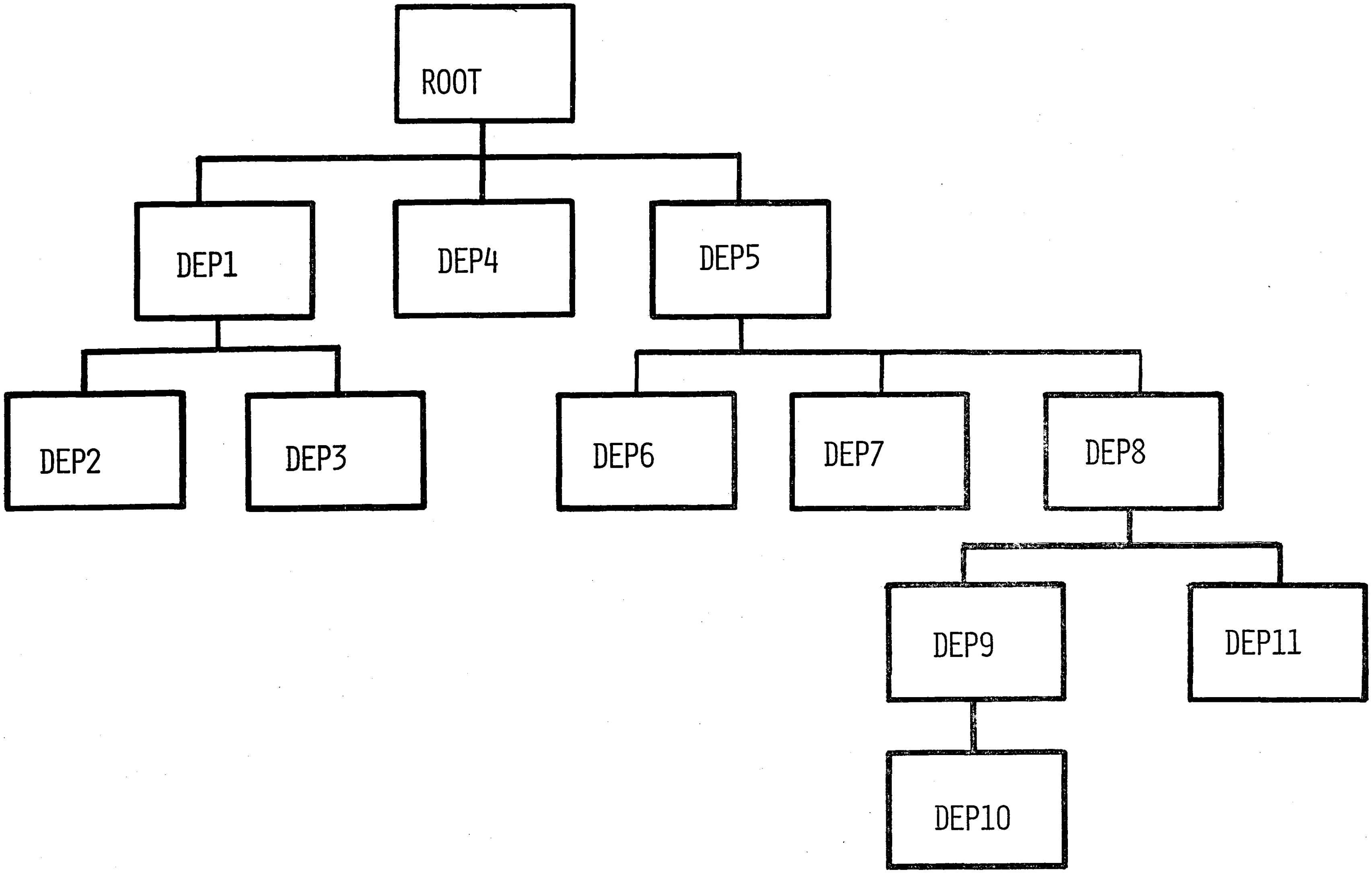


ROOT	DEP1	DEP2	DEP3	DEP4
PART NO.	DESCRIPTION	QUAN. ON HAND	UNIT PRICE	VENDOR

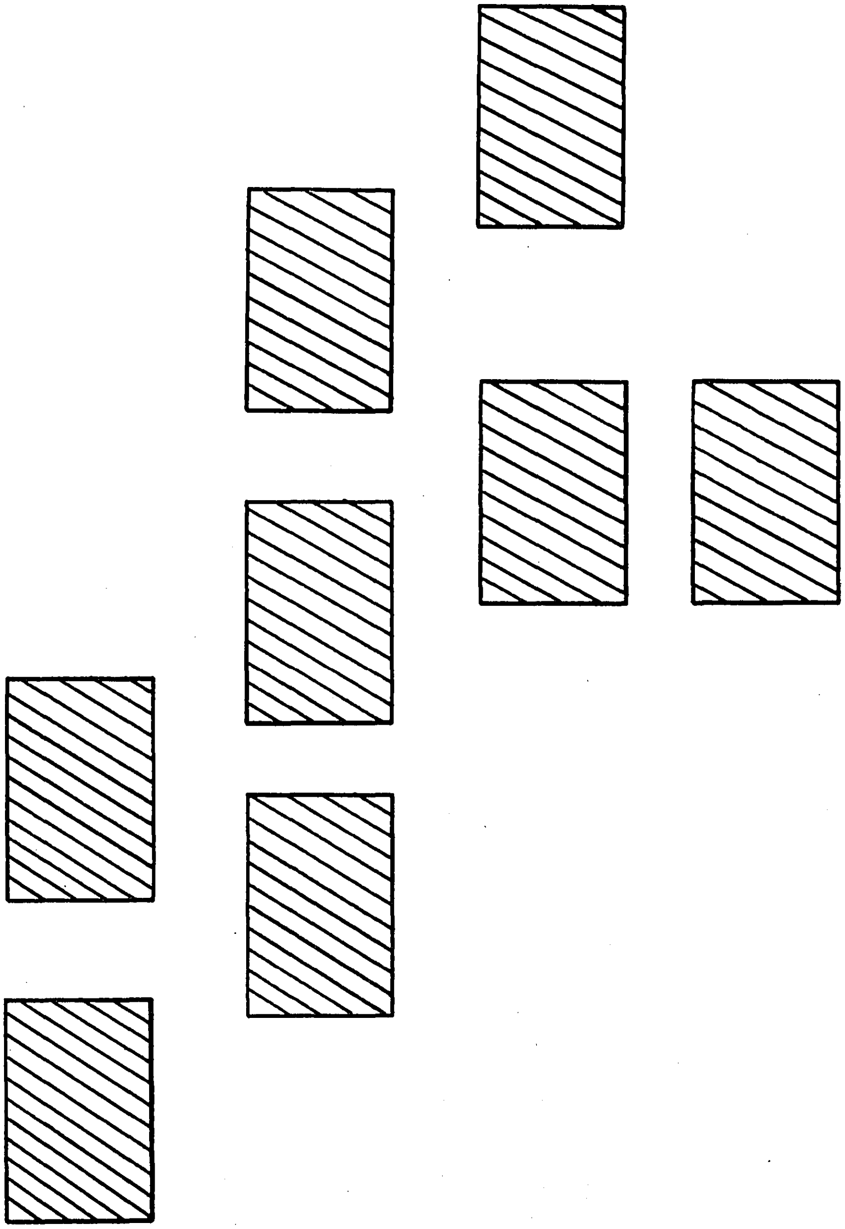
CONCATENATED KEYS

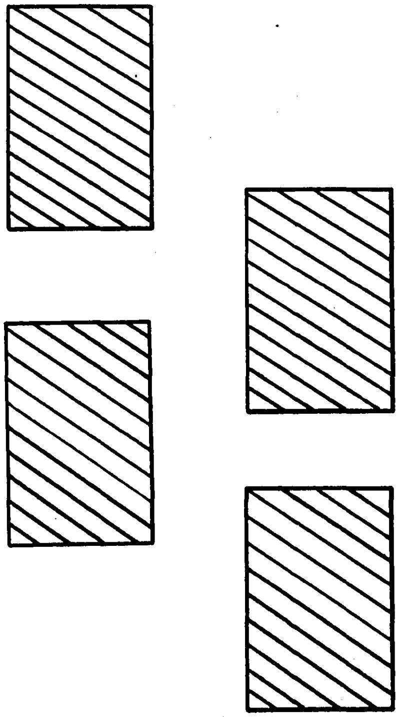


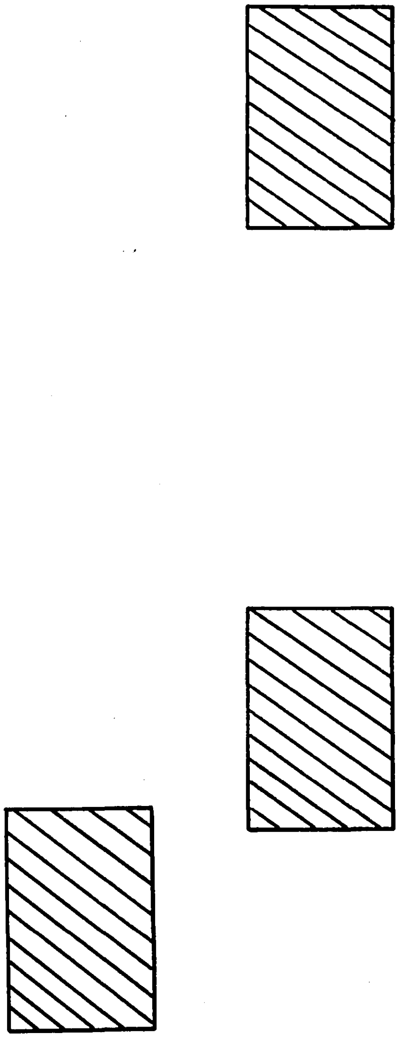
LOGICAL HIERARCHY



TOP TO BOTTOM - LEFT TO RIGHT





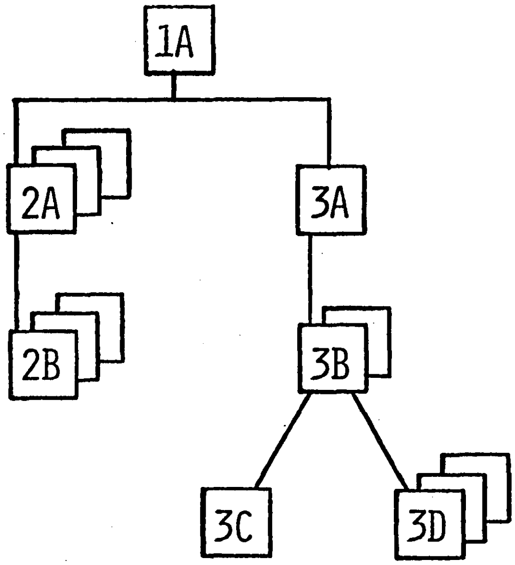


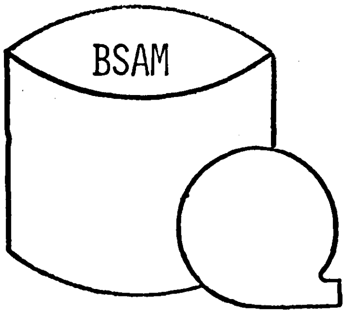
THREE LEVEL HIERARCHY

		KEY LENGTH
ROOT	002	3
DEP1	0004	4
DEP2	02	2
DEP2	07	2
DEP2	24	2
DEP1	0008	4
DEP1	0025	4
DEP1	0911	4
DEP1	0912	4
DEP2	03	2
DEP2	05	2
DEP2	07	2
DEP2	24	2
DEP3	001	3
DEP3	002	3
DEP1	0925	4
DEP4	451	3
DEP4	455	3

HSAM

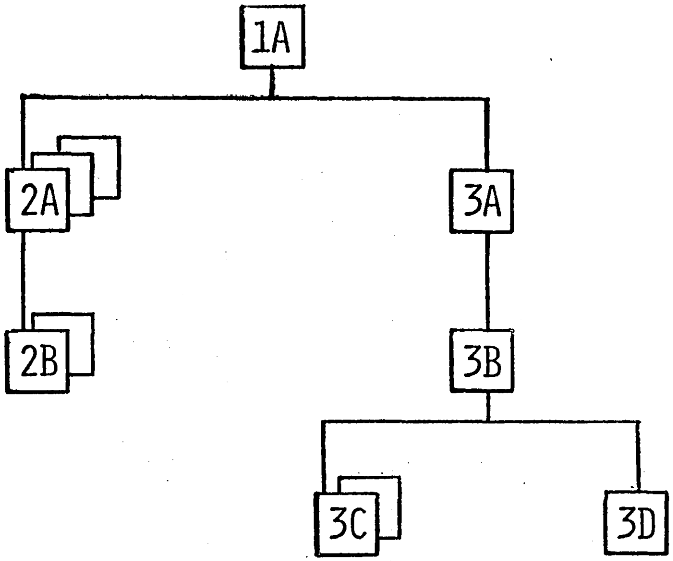
DATA BASE
RCD STRUCTURE

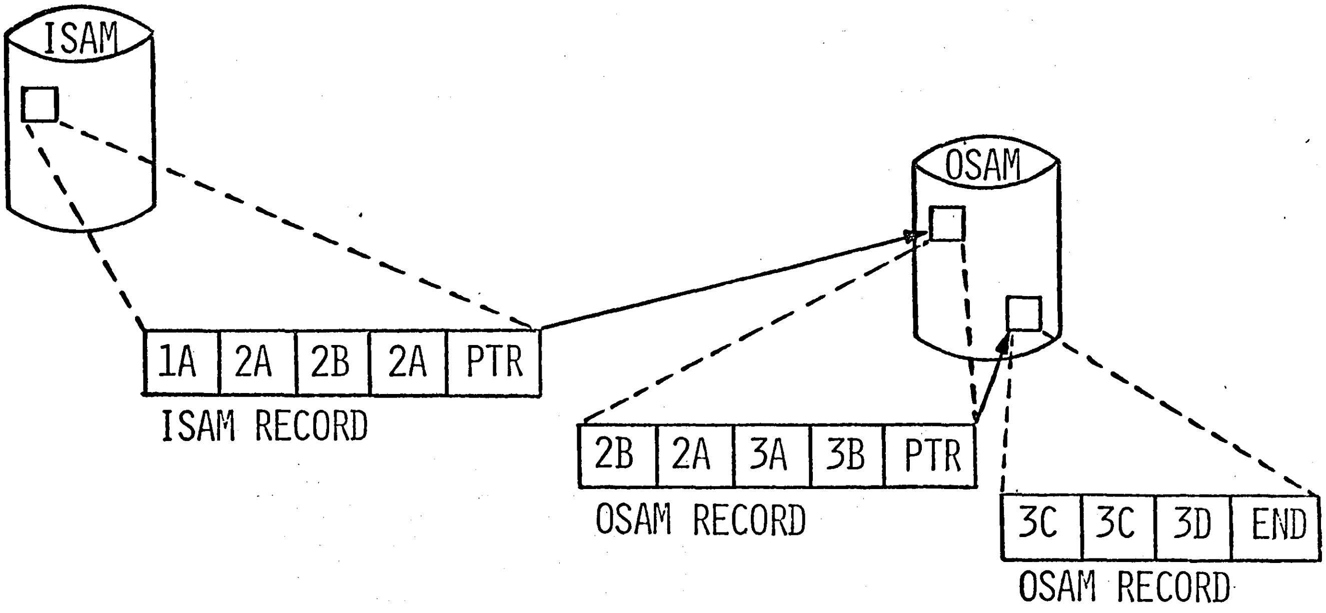




BSAM RCD. NO. 1	1A	2A	2B	2A	2B	2B
BSAM RCD. NO. 2	2A	3A	3B	3C	3D	3B
BSAM RCD. NO. 3	3D	3D	1A	2A	2B	2B

HISAM - SINGLE DATA SET GROUP

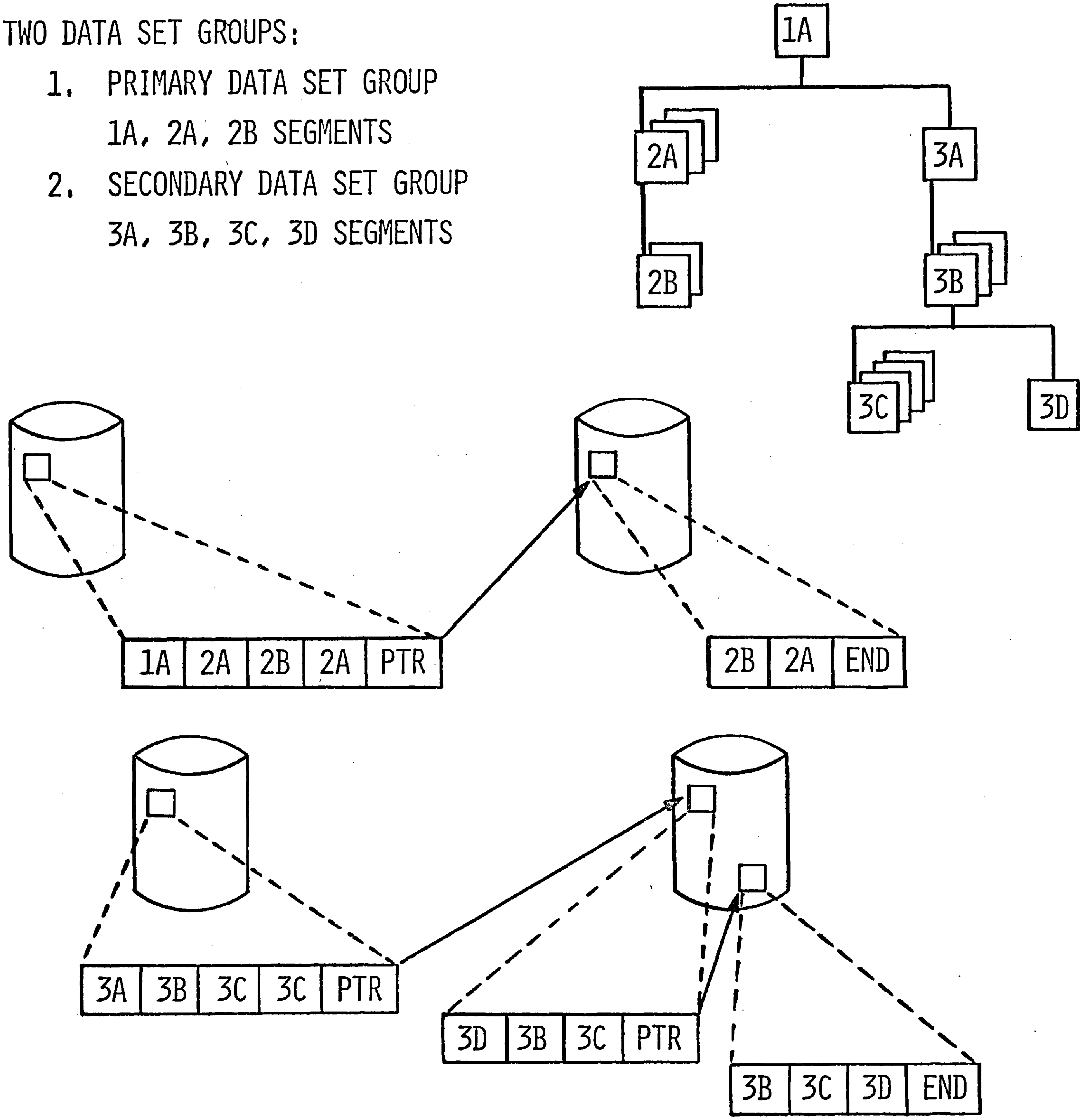




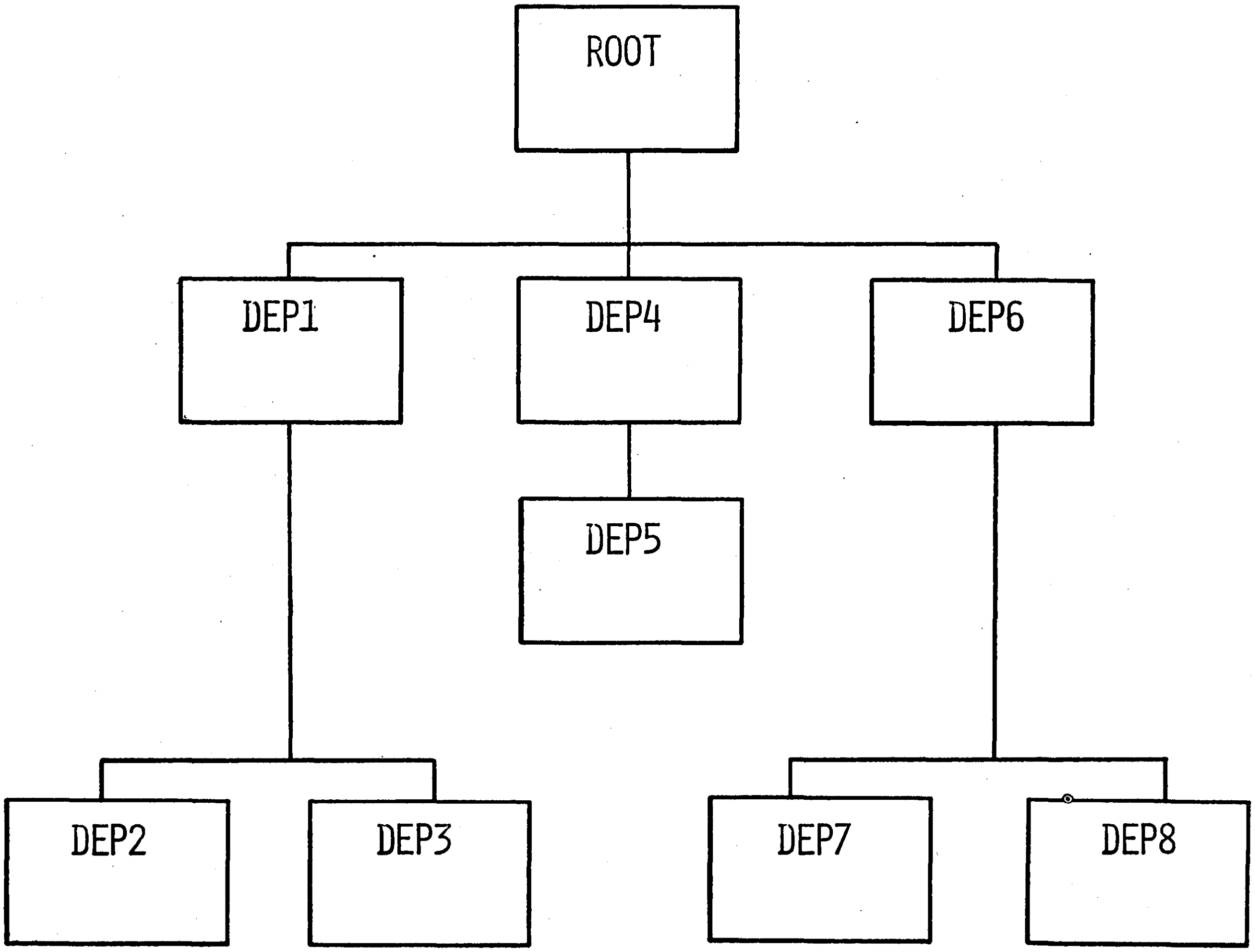
HISAM - MULTIPLE DATA SET GROUP

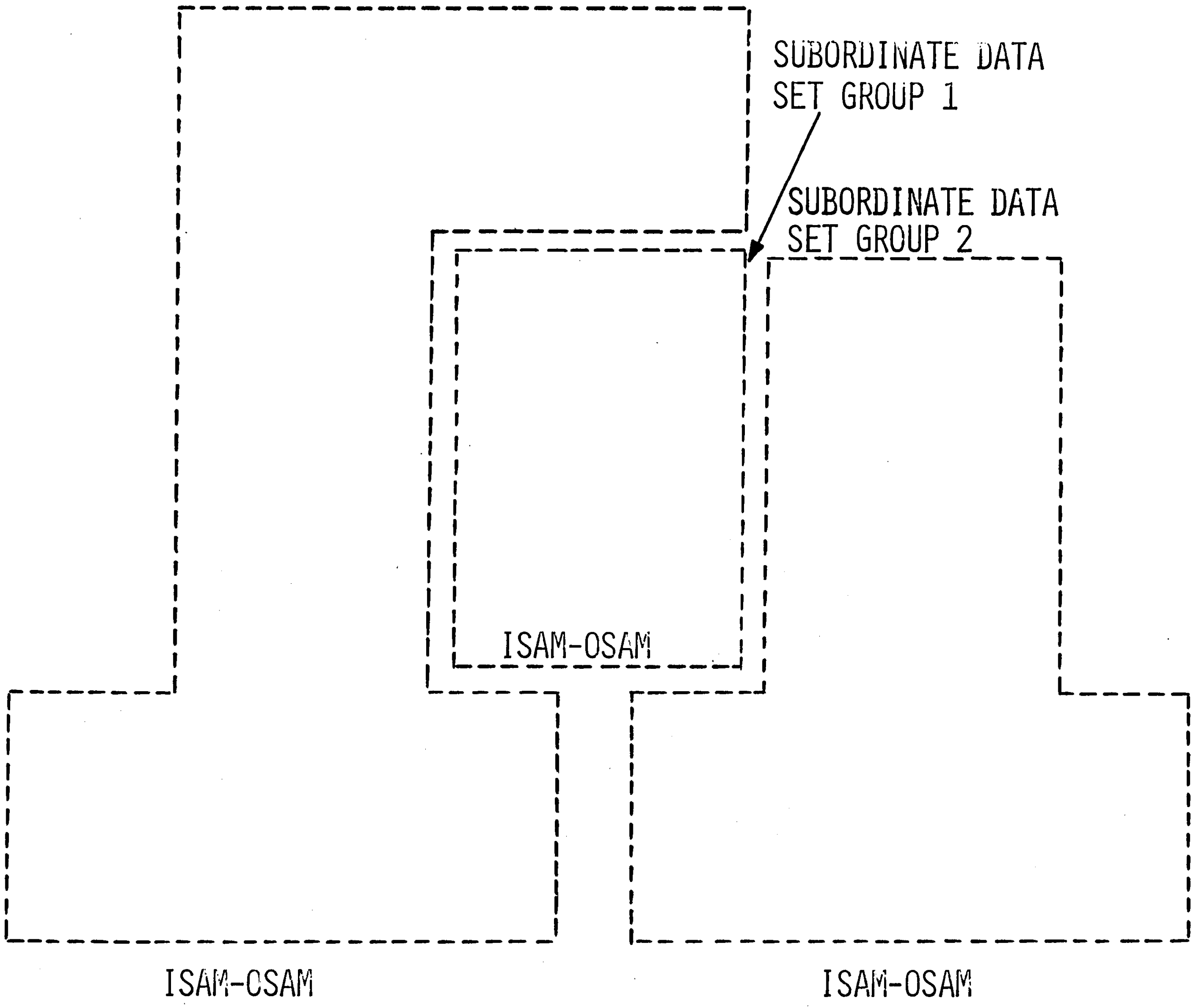
TWO DATA SET GROUPS:

1. PRIMARY DATA SET GROUP
   1A, 2A, 2B SEGMENTS
2. SECONDARY DATA SET GROUP
   3A, 3B, 3C, 3D SEGMENTS



MULTIPLE DATA SET GROUPS




SUBORDINATE DATA
SET GROUP 1
SUBORDINATE DATA
SET GROUP 2
ISAM-OSAM
ISAM-OSAM
ISAM-OSAM

001

02 003 04

006

004 005 05 06

ROOT 001	DEP1 02	DEP2 004	DEP3 005	

PRIME DATA SET

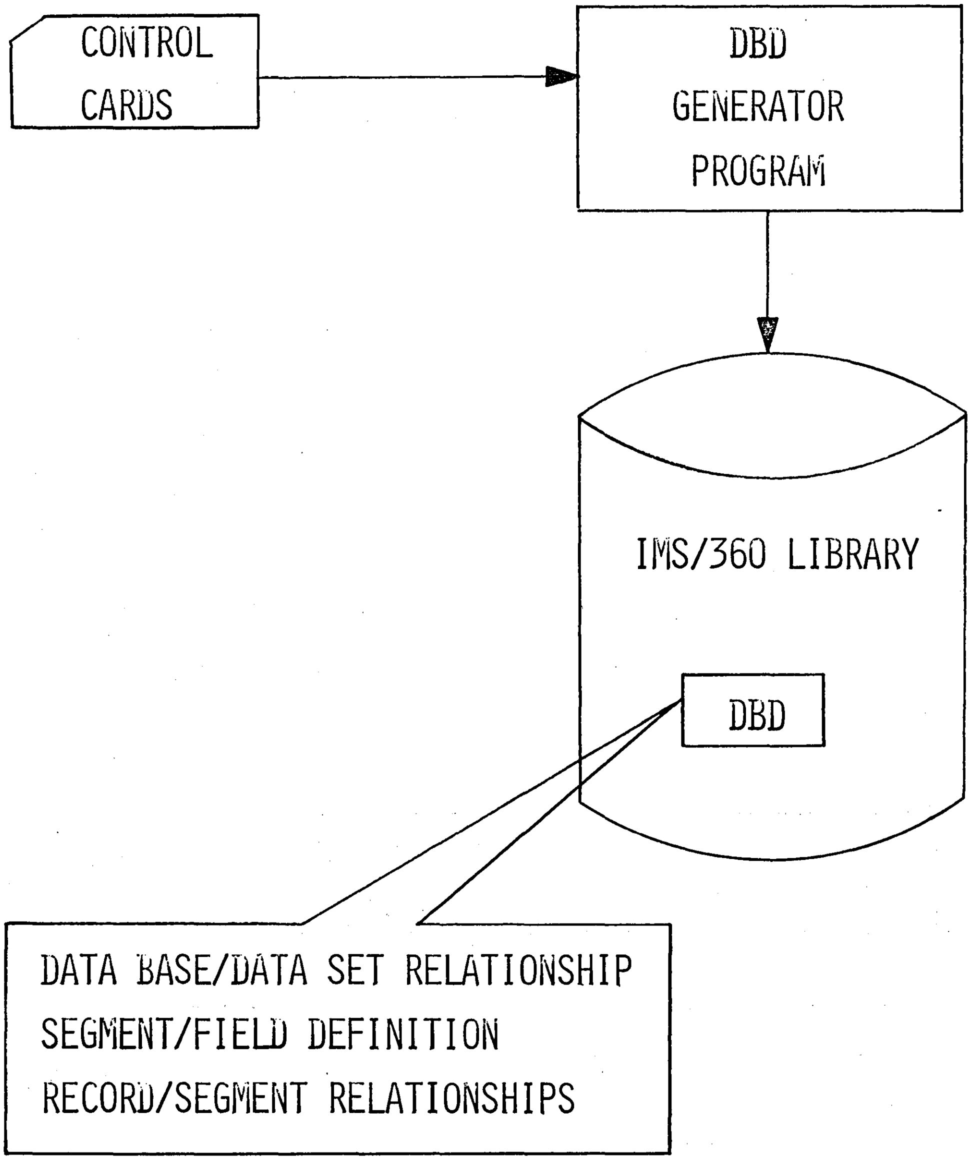
ROOT KEY 001	DEP4 003	DEP5 006	

SUBORDINATE DATA SET GROUP 1

ROOT KEY 001	DEP6 04	DEP7 05	DEP8 06	

SUBORDINATE DATA SET GROUP 2

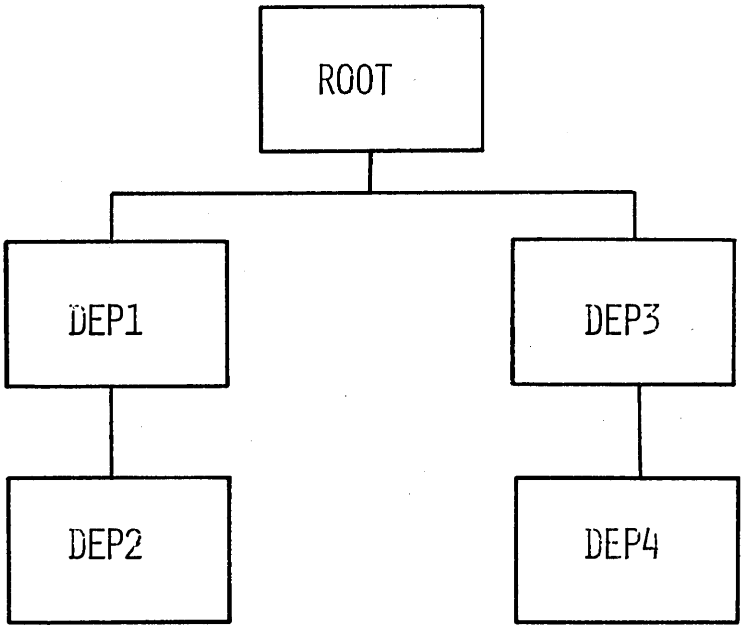
DATA BASE DESCRIPTION



DBD INPUT CARDS

1	[PRINT	NOGEN]
2	DBD	NAME=,ACCESS=
3	DMAN	DD1=,DEV1=,[DD2=],[DLIOF=]
4	SEGM	NAME=,PARENT=,BYTES=,FREQ=
5	FLDK	NAME=,TYPE=,BYTES=,START=
	[FLD	NAME=,TYPE=,BYTES=,START=]
6	DBDGEN	
7	FINISH	
8	END	

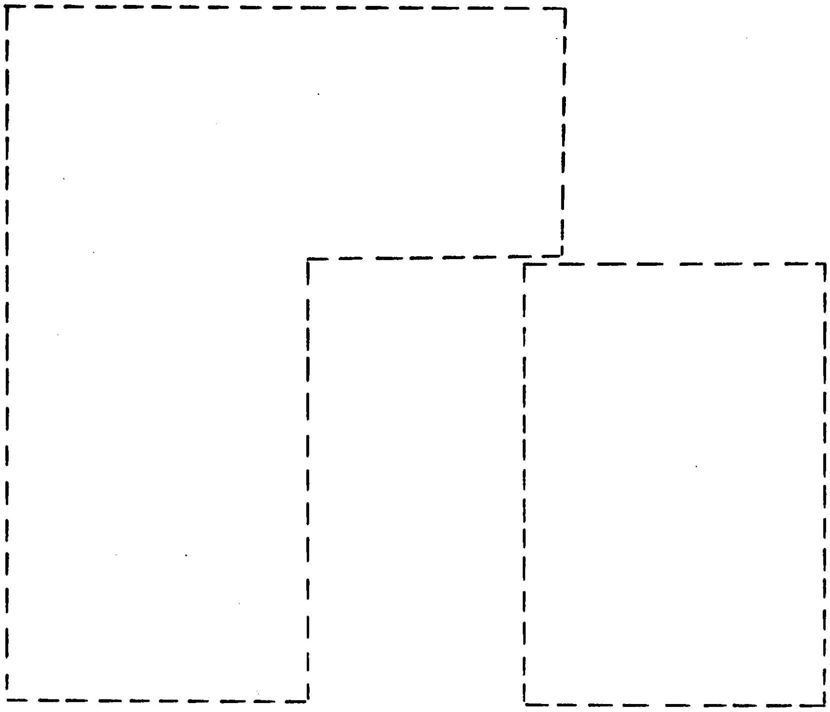
EXAMPLE OF DBD GENERATION



```
PRINT     NOGEN

SEGM      NAME=ROOT,PARENT=0,BYTES=90,FREQ=500
FLDK      NAME=KEY,TYPE=C,BYTES=6,START=1
FLD       NAME=FIELD,TYPE=C,BYTES=84,START=7
SEGM      NAME=DEP1,PARENT=ROOT,BYTES=91,FREQ=1
FLDK      NAME=KEY1,TYPE=C,BYTES=4,START=19
SEGM      NAME=DEP2,PARENT=DEP1,BYTES=300,FREQ=1
FLDK      NAME=KEY2,TYPE=C,BYTES=8,START=1
FLD       NAME=FIELD3,TYPE=P,BYTES=253,START=7

SEGM      NAME=DEP3,PARENT=ROOT,BYTES=91,FREQ=1
FLDK      NAME=KEY3,TYPE=C,BYTES=6,START=14
SEGM      NAME=DEP4,PARENT=DEP3,BYTES=259,FREQ=1
FLDK      NAME=KEY4,TYPE=C,BYTES=3,START=1
DBDGEN
FINISH
END
```

```
DBD        NAME=AB,ACCESS=SEQ
DMAN       DD1=DBP,DEV1=TAPE,DD2=DBP1
```

```
DBD     NAME=AB,ACCESS=INDEX
DMAN    DD1=DBP,DEV1=2311,DLIOF=OVFL1
```



```
DBD      NAME=AB,ACCESS=INDEX
DMAN     DD1=ABC,DEV1=2311,DLIOF=OVFL1
```

```
DMAN     DD1=DB1,DEV1=2311,DLIOF=OVFL4
```

PSB INPUT

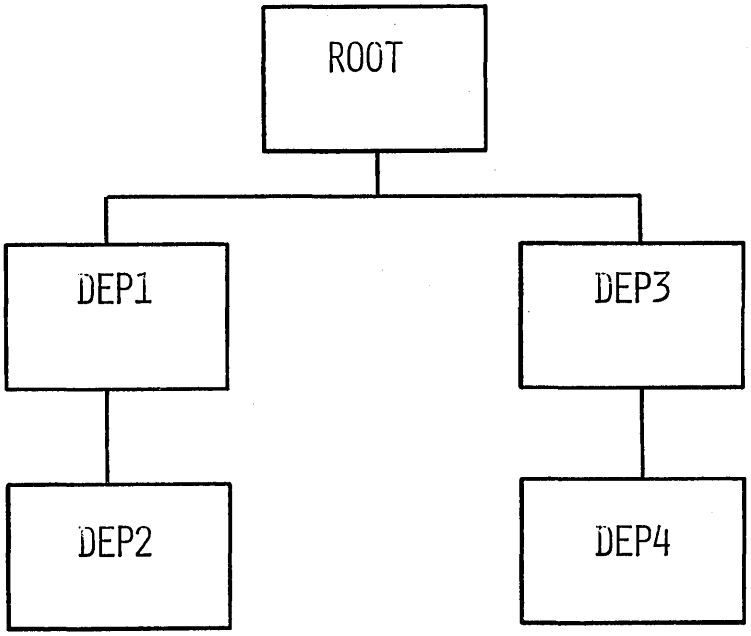
```
PCB          TYPE=DB,
             DBDNAME=NAME,
             PROCOPT=X,
             KEYLEN=LENGTH
```

```
SENSEG       NAME OF SENSITIVE SEGMENT,
             PARENT OF THIS SENSITIVE SEGMENT
```

```
PSBGEN       LANG=COMPILER LANGUAGE,
             PSBNAME=NAME OF THIS PSB
```

```
END
```

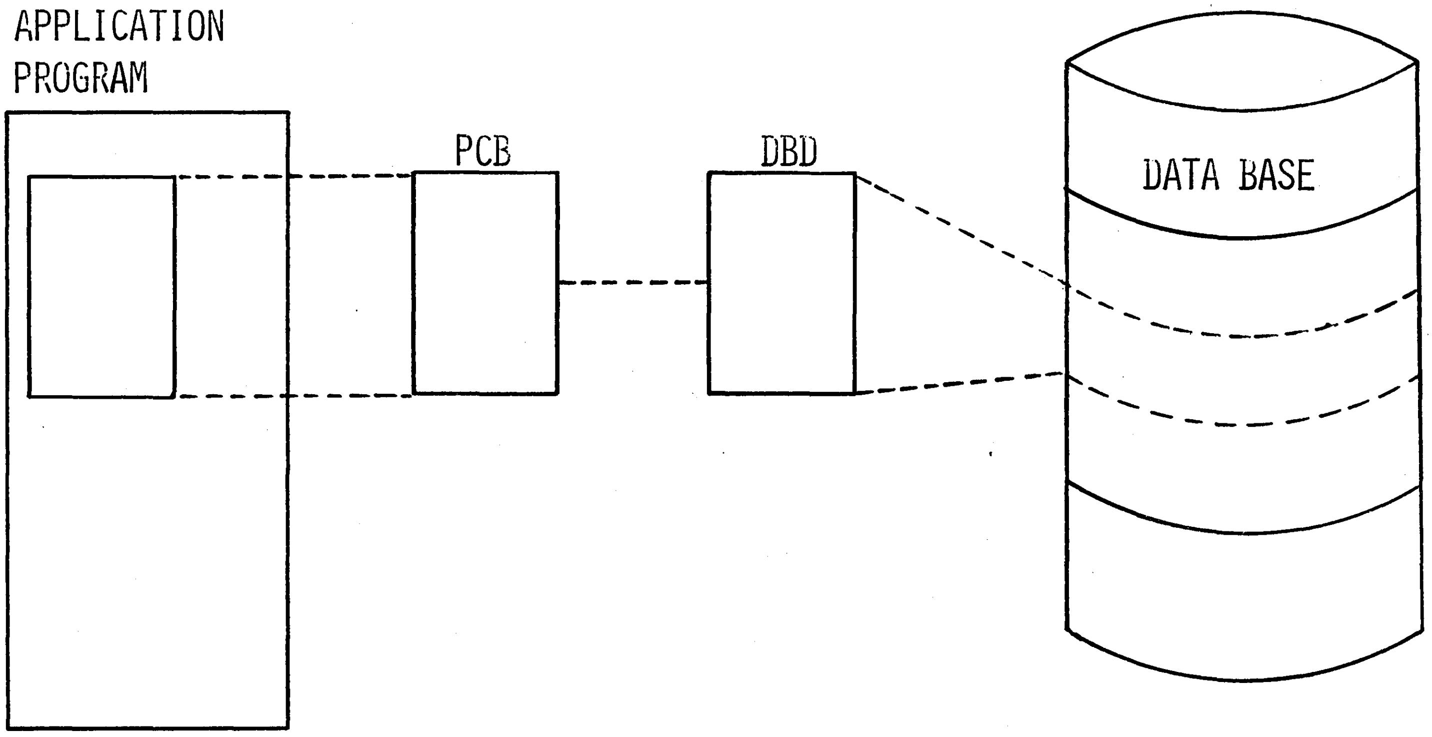
EXAMPLES OF PSB



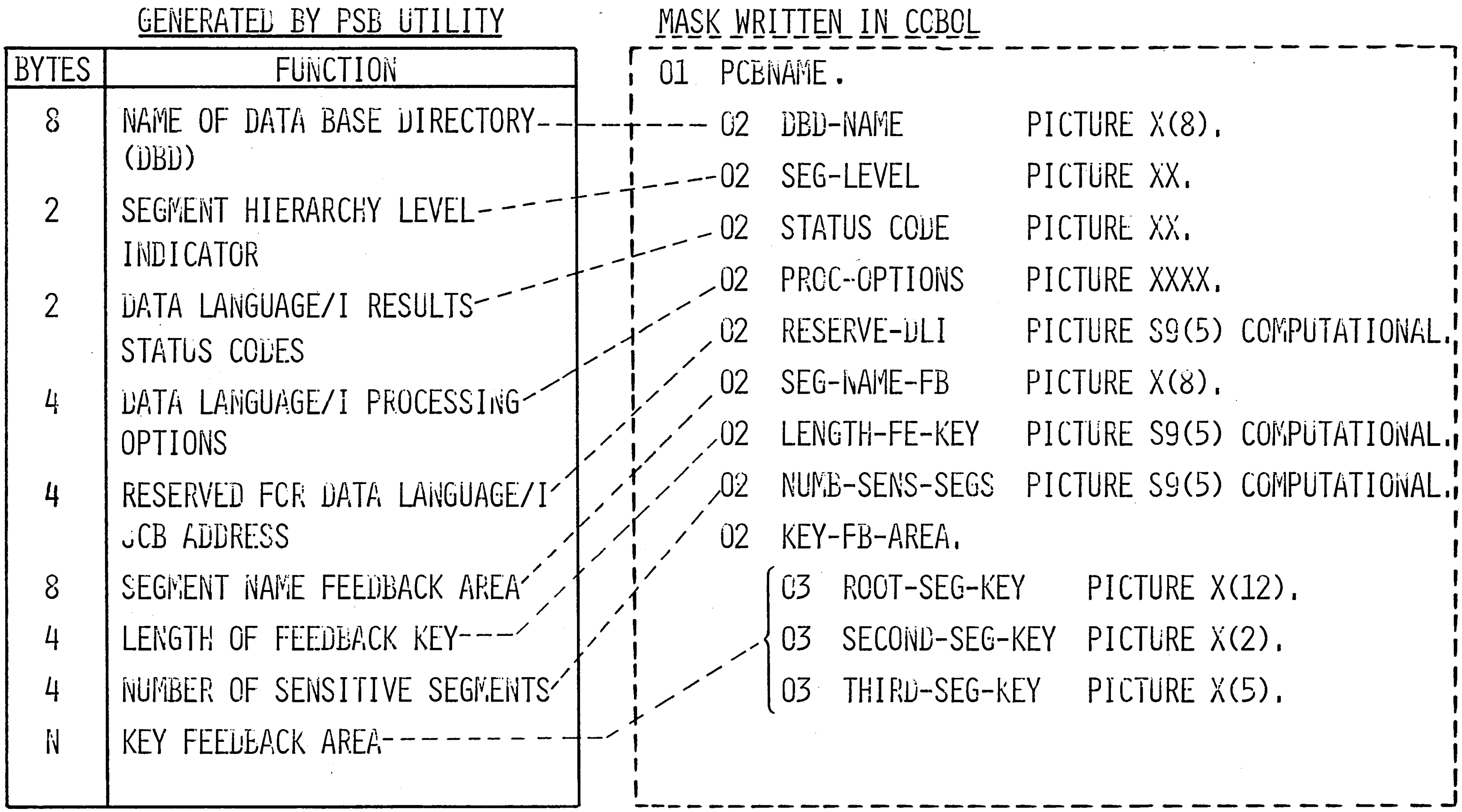
```
PCB       TYPE=DB,DBDNAME=AB,PROCOPT=A,KEYLEN=18
SENSEG    ROOT
SENSEG    DEP1,ROOT
SENSEG    DEP2,DEP1
SENSEG    DEP3,ROOT
SENSEG    DEP4,DEP3
PSBGEN    LANG=COBOL,PSBNAME=DISBURSE
END
```

```
PCB       TYPE=DB,DBDNAME=AB,PROCOPT=G,KEYLEN=15
SENSEG    ROOT
SENSEG    DEP3,ROOT
SENSEG    DEP4,DEP3
PSBGEN    LANG=PL/I,PSBNAME=AUDIT
END
```

THE PCB IS A FILTER



ONE-TO-ONE RELATIONSHIP IN COBOL



GENERATED BY PSB UTILITY

BYTES	FUNCTION
8	NAME OF DATA BASE DIRECTORY (DBD)
2	SEGMENT HIERARCHY LEVEL INDICATOR
2	DATA LANGUAGE/I RESULTS STATUS CODES
4	DATA LANGUAGE/I PROCESSING OPTIONS
4	RESERVED FOR DATA LANGUAGE/I JCB ADDRESS
8	SEGMENT NAME FEEDBACK AREA
4	LENGTH OF FEEDBACK KEY
4	NUMBER OF SENSITIVE SEGMENTS
N	KEY FEEDBACK AREA

MASK WRITTEN IN COBOL

```
01  PCBNAME.
    02  DBD-NAME         PICTURE X(8).
    02  SEG-LEVEL        PICTURE XX.
    02  STATUS CODE      PICTURE XX.
    02  PROC-OPTIONS     PICTURE XXXX.
    02  RESERVE-DLI      PICTURE S9(5) COMPUTATIONAL.
    02  SEG-NAME-FB      PICTURE X(8).
    02  LENGTH-FB-KEY    PICTURE S9(5) COMPUTATIONAL.
    02  NUMB-SENS-SEGS   PICTURE S9(5) COMPUTATIONAL.
    02  KEY-FB-AREA.
       {03  ROOT-SEG-KEY     PICTURE X(12).
       {03  SECOND-SEG-KEY   PICTURE X(2).
       {03  THIRD-SEG-KEY    PICTURE X(5).
```

ONE-TO-ONE RELATIONSHIP IN PL/I

GENERATED BY PSB UTILITY

BYTES	
8	NAME OF DATA BASE DIRECTORY (DBD)
2	SEGMENT HIERARCHY LEVEL INDICATOR
2	DATA LANGUAGE/I RESULTS STATUS CODES
4	DATA LANGUAGE/I PROCESSING OPTIONS TO BE USED
4	RESERVED FOR DATA LANGUAGE/I JCB ADDRESS
8	SEGMENT NAME FEEDBACK AREA
4	LENGTH OF FEEDBACK KEY
4	NUMBER OF SENSITIVE SEGMENTS
N	KEY FEEDBACK AREA

MASK WRITTEN IN PL/I

```
DECLARE  1  SAMPLE PCB,
            2  DBD_NAME          CHARACTER (8),
            2  SEG_LEVEL         CHARACTER (2),
            2  STATUS_CODE       CHARACTER (2),
            2  PROC_OPTIONS      CHARACTER (4),
            2  RESERVE_DLI       FIXED BINARY (31,0),
            2  SEG_NAME_FB       CHARACTER (8),
            2  LENGTH_FB_KEY     FIXED BINARY (31,0),
            2  NUMB_SENS_SEGS    FIXED BINARY (31,0),
            2  KEY_FB_AREA,
               3 ROOT_SEG_KEY       CHARACTER (12),
               3 SECOND_SEG_KEY     CHARACTER (2),
               3 THIRD_SEG_KEY      CHARACTER (5);
```

DATA LANGUAGE/I CALLS

COBOL

```
ENTER LINKAGE .
    CALL 'CBLTDLI' USING
          CALL-FUNCTION,
          PCBNAME,
          USER-IO-AREA,
          SSA1,---,SSAN .
ENTER COBOL .
```

PL/I

```
CALL PLITDLI (PARM_COUNT,
     CALL_FUNCTION,
     PCBNAME,
     USER_IO_AREA,
     SSA1,---,SSAN);
```

SEGMENT SEARCH ARGUMENT

	SEGMENT NAME	(	FIELD NAME	RELATIONAL OPER	COMPARATIVE VALUE	)
CHARACTERS	◄----8---►	1	◄--8---►	◄----2----►	◄-1 TO 255---►	
				B=,B>,B<,¬= , =>,=<		

```
01     SSA1
       02 SEG-NAME          PICTURE X (8) VALUE 'PARTBBBB'
       02 SEG-QUAL          PICTURE X VALUE '('.
       02 SEG-KEYNAME       PICTURE X (8) VALUE 'PARTKEYB'.
       02 SEG-OPERATOR      PICTURE XX VALUE 'B='.
       02 SEG-KEY VALUE     PICTURE X (6) VALUE 'AB7024'.
       02 SEG-END-CHAR      PICTURE X VALUE ')'.

           PARTBBBB(PARTKEYBB=AB7024)
```

SSA EXAMPLES

```
PARTbbbbb

PARTbbbb(PARTKEYbb=AB6023)

PARTbbbb(PARTKEYbb=AB7024)
INVNTORY(KLOCbbbbb=BIN324)
VENDHIST(KEYNAMEbb=SMITHBRD)

PARTbbbb(PARTKEYbb=AB7024)
INVNTORY(BUILDINGb=BLD21)
VENDHISTb
```

DATA LANGUAGE/I FUNCTION CODES

FUNCTION	CODE
GET UNIQUE	GUBB
GET NEXT	GNBB
GET NEXT WITHIN PARENT	GNPB
GET HOLD UNIQUE	GHUB
GET HOLD NEXT	GHNB
GET HOLD NEXT WITHIN PARENT	GHNP
INSERT	ISRT
DELETE	DLET
REPLACE	REPL

GET UNIQUE - STATUS CODES

STATUS CODE	MEANING
'AB'	NO SEGMENT I/O AREA SPECIFIED IN CALL
'AC'	HIERARCHICAL ERROR IN SSA's
'AD'	ILLEGAL FUNCTION PARAMETER
'AG'	FIRST SSA MUST BE FOR A LEVEL ONE SEGMENT
'AH'	SSA's MISSING
'AI'	DATA MANAGEMENT OPEN ERROR
'AJ'	INVALID SSA QUALIFICATION FORMAT
'AK'	INVALID FIELD NAME IN CALL
'AL'	TERMINAL PCB INVALID IN BATCH ONLY PROCESSING REGION
'AM'	CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION
'AO'	I/O ERROR IN ISAM OR BSAM
'AP'	I/O ERROR IN OSAM
'GE'	SEGMENT NOT FOUND

```
PARTBBBB(PARTKEYBB=AB7024)
INVNTORY(KLOCBBBBB=BIN324)
VENDHIST(KEYNAMEBB=SMITHBRO)
```

GET NEXT - STATUS CODES

STATUS CODE	MEANING
'AB'	NO SEGMENT I/O AREA SPECIFIED IN CALL
'AC'	HIERARCHICAL ERROR IN SSA's
'AD'	ILLEGAL FUNCTION PARAMETER
'AI'	DATA MANAGEMENT OPEN ERROR
'AJ'	INVALID SSA QUALIFICATION FORMAT
'AK'	INVALID FIELD NAME IN CALL
'AL'	TERMINAL PCB INVALID IN BATCH ONLY PROCESSING REGION
'AM'	CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION
* 'AN'	GN CALL NOT ALLOWED IMMEDIATELY AFTER ISRT CALL
'AO'	I/O ERROR IN ISAM OR BSAM
'AP'	I/O ERROR IN OSAM
* 'GA'	CROSSED HIERARCHICAL BOUNDARY
* 'GB'	END OF DATA SET
'GE'	SEGMENT NOT FOUND
* 'GK'	DIFFERENT SEGMENT NAME AT SAME LEVEL (RETURNED ON UNQUALIFIED CALLS ONLY)

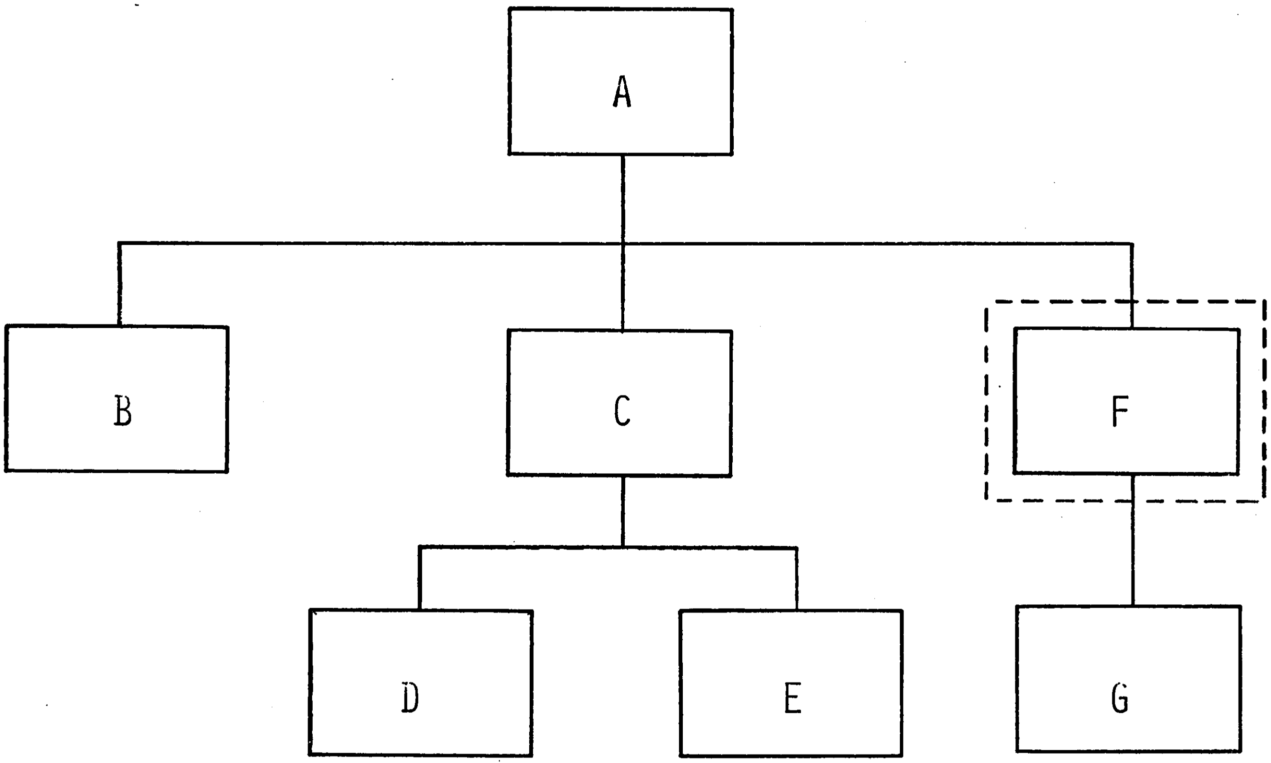
GET NEXT WITHIN PARENT - STATUS CODES

	STATUS CODE	MEANING
	'AB'	NO SEGMENT I/O AREA SPECIFIED IN CALL
	'AC'	HIERARCHICAL ERROR IN SSA's
	'AD'	ILLEGAL FUNCTION PARAMETER
*	'AE'	ON GNP/GHNP CALLS, FIRST LEVEL SSA ILLEGAL
	'AI'	DATA MANAGEMENT OPEN ERROR
	'AJ'	INVALID SSA QUALIFICATION FORMAT
	'AK'	INVALID FIELD NAME IN CALL
	'AL'	TERMINAL PCB IS INVALID IN A BATCH PROCESSING REGION
	'AM'	CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION
	'AN'	GN CALL NOT ALLOWED IMMEDIATELY AFTER ISRT CALL
	'AO'	I/O ERROR IN ISAM OR BSAM
	'AP'	I/O ERROR IN OSAM
	'GA'	CROSSED HIERARCHICAL BOUNDARY INTO HIGHER LEVEL
	'GE'	SEGMENT NOT FOUND
	'GK'	DIFFERENT SEGMENT TYPE AT SAME LEVEL RETURNED (RETURNED ON UNQUALIFIED CALLS ONLY)
*	'GP'	PARENT NOT ESTABLISHED

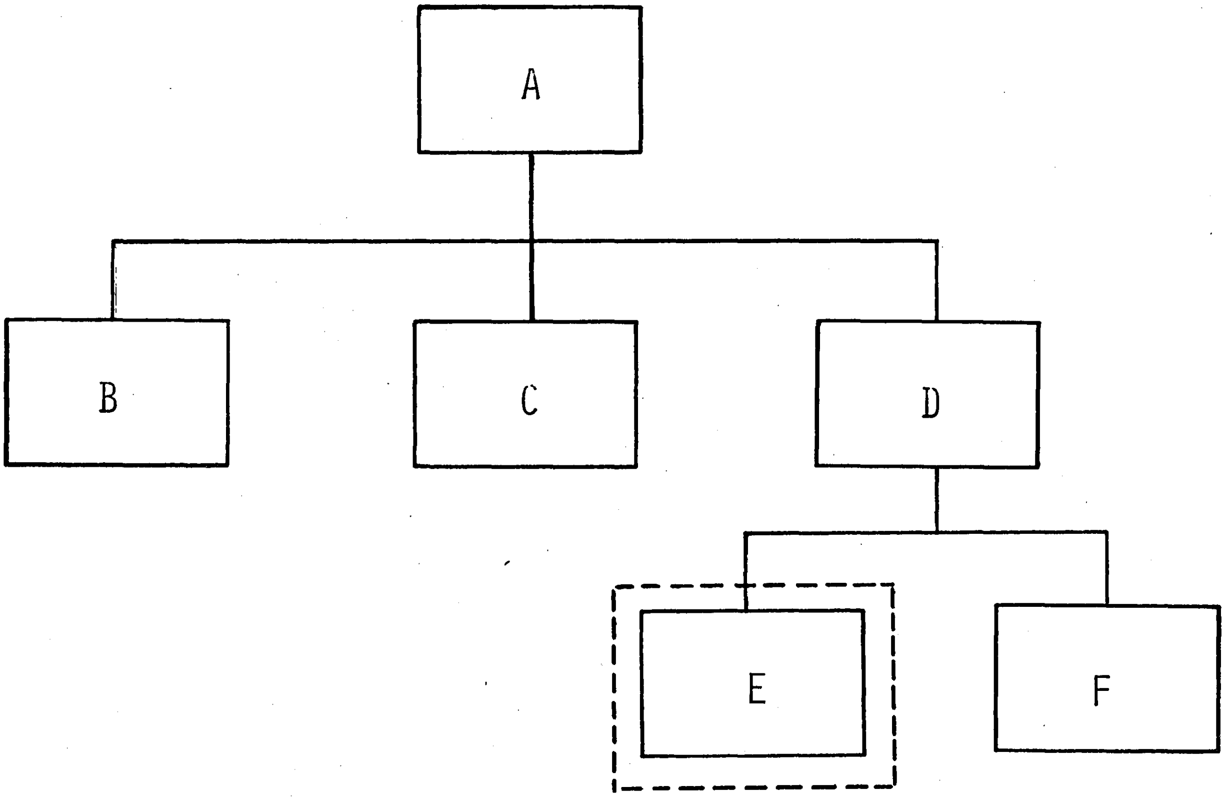
INSERT - STATUS CODE

	STATUS CODE	MEANING
	'AB'	NO SEGMENT I/O AREA SPECIFIED IN CALL
	'AC'	HIERARCHICAL ERROR IN SSA's
	'AD'	THE FIRST SSA MUST BE FOR A LEVEL ONE SEGMENT
	'AH'	SSA's MISSING
	'AI'	DATA MANAGEMENT OPEN ERROR
	'AJ'	INVALID SSA QUALIFICATION FORMAT
*	'AK'	INVALID FIELD NAME IN CALL
	'AL'	TERMINAL PCB IS INVALID IN BATCH ONLY PROCESSING REGION
	'AM'	CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION
	'AO'	I/O ERROR IN ISAM OR BSAM
	'AP'	I/O ERROR IN OSAM
	'GE'	QUALIFYING SEGMENT NOT FOUND
INSERT	'II'	DUPLICATE SEGMENT ALREADY IN DATA BASE
	'LB'	DUPLUCATE SEGMENT ALREADY IN DATA BASE
	'LC'	SEGMENT KEY FIELD CONTENTS OUT OF SEQUENCE
	'LD'	THE PARENT FOR THIS SEGMENT HAS NOT BEEN SUBMITTED
	'LE'	SUBMISSIONS VIOLATE ORDER WITHIN A LEVEL
	'LH'	LEVEL IMMEDIATELY FOLLOWING ROOT HAS BEEN SKIPPED

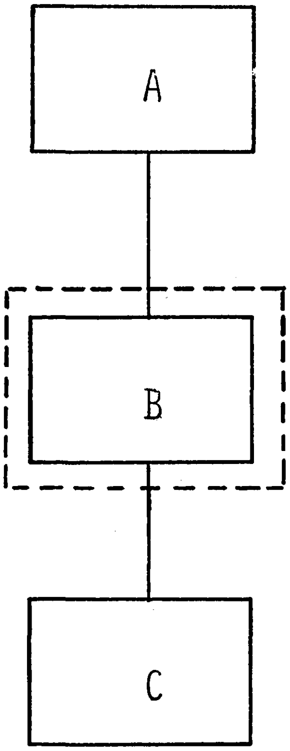
'LD' STATUS CODE



'LE' STATUS CODE



'LH' STATUS CODE



DELETE/REPLACE - STATUS CODES

STATUS CODE	MEANING
'AB'	NO SEGMENT I/O AREA ADDRESS IN CALL
'AD'	ILLEGAL FUNCTION PARAMETER
'AF'	NO SSA's ALLOWED FOR DLET/REPL CALLS
'AI'	DATA MANAGEMENT OPEN ERROR
'AL'	TERMINAL PCB INVALID IN BATCH ONLY REGION
'AO'	I/O ERROR IN ISAM OR BSAM
'AP'	I/O ERROR IN OSAM
'AM'	CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION
'DA'	KEY FIELD CANNOT BE CHANGED ON A DELETE OR REPLACE
'DJ'	NO PRECEDING SUCCESSFUL GET HOLD

DATA BASE WITH THREE RECORDS

ENTRY					KEY FIELD
1	ROOT 001				RUT
2		DEP1 0001			KFLD1
3		DEP1 0002			KFLD1
4		DEP2 0001			KFLD2
5			DEP3 0004		KFLD3
6				DEP4 0008	KFLD4
7			DEP5 0006		KFLD5
8			DEP5 0007		KFLD5
9			DEP6 0007		KFLD6
10		DEP2 0004			KFLD2
11			DEP3 0005		KFLD3
12			DEP5 0007		KFLD5
13		DEP2 0005			KFLD2
14	ROOT 002				RUT
15		DEP1 0012			KFLD1
16		DEP2 0011			KFLD2
17			DEP3 0004		KFLD3
18			DEP5 0007		KFLD5
19	ROOT 005				RUT

		RETRIEVES ENTRY
GU	ROOT(RUT=002)	
	DEP1(KFLD1=0012)	15
GU	ROOT	1
GU	ROOT(RUT=001)	
	DEP2(KFLD2=0004)	
	DEP5(KFLD5=0007)	12
GU	ROOT(RUT=001)	
	DEP5(KFLD5=0008)	AC
GN		1
GN	ROOT	1
GN	ROOT(RUT=001)	
	DEP1	2
GN	ROOT	
	DEP2	4
GN	ROOT(RUT=001)	
	DEP2	
	DEP3(KFLD3=0005)	11

		RETRIEVES ENTRY
GN	ROOT	
	DEP5(KFLD5=0007)	AC
GN	ROOT	
	DEP2	
	DEP3	5
GN	ROOT(RUT=001)	1
GN	ROOT(RUT=001)	
	DEP1(KFLD1=0002)	3
GN	ROOT(RUT=001)	
	DEP2(KFLD2=0001)	
	DEP3(KFLD3=0004)	5
GN	ROOT(RUT=004)	GE
GN	ROOT(RUT=002)	14

		RETRIEVES ENTRY
GN		1
GN		2
GN	ROOT	1
GN		2
GN	ROOT	1
GN	ROOT	14
GN	ROOT	1
GN		2
GN		3
GN		4
GN	ROOT(RUT=001)	
	DEP2(KFLD2=0001)	
	DEP6(KFLD6=0007)	9
GN		10
GN	ROOT(RUT=001)	
	DEP2(KFLD2=0001)	
	DEP5(KFLD5=0007)	8
GN	DEP5	12

		RETRIEVES ENTRY
GN	ROOT(RUT=001)	
	DEP2(KFLD=0005)	13
GN		14
GN	ROOT(RUT=001)	
	DEP2(KFLD2=0004)	
	DEP5(KFLD5=0009)	GE
GN		13
GN	DEP3	5
GN	DEP3	11
GN	DEP3	17
GN	DEP3	GB
GN	ROOT	1
GNP		2
GU	ROOT(RUT=001)	
	DEP2(KFLD2=0004)	10
GNP		11
GNP		12
GNP		GE

		RETRIEVES ENTRY
GU	ROOT(RUT=001)	1
GNP	ROOT(RUT=001)	
	DEP1(KFLD1=0001)	AE
GU	ROOT(RUT=001)	
	DEP1(KFLD1=0001)	2
GNP		GE

COBOL PROGRAMMING EXAMPLE

```
IDENTIFICATION DIVISION.
DATA DIVISION.
WORKING STORAGE SECTION.

01   IN-ROOT.
     02   KEY PICTURE X(6) .
     02   FIELD PICTURE X(84) .
01   IN-DEP1.
     02   KEY1 PICTURE X(4) .
     02   FIELD1 PICTURE X(85) .
01   IN-DEP2.
     02   KEY2 PICTURE X(8) .
     02   FIELD2 PICTURE X(253) .
```

```
01   SSA1.
     02    SEG1-NAME  PICTURE X(8).
     02    PAREN1     PICTURE X VALUE '('.
     02    KEY1-NAME  PICTURE X(8).
     02    OP1        PICTURE XX.
     02    VALUE1     PICTURE 9(6).
     02    PAREN11    PICTURE X VALUE ')'.
01   SSA2.
     02    SEG2-NAME  PICTURE X(8).
     02    PAREN2     PICTURE X VALUE '('.
     02    KEY2-NAME  PICTURE X(8).
     02    OP2        PICTURE XX.
     02    VALUE2     PICTURE 9(4).
     02    PAREN22    PICTURE X VALUE ')'.
01   SSA3.
     02    SEG3-NAME  PICTURE X(8).
     02    PAREN3     PICTURE X VALUE '('.
     02    KEY3-NAME  PICTURE X(8).
     02    OP3        PICTURE XX.
     02    VALUE3     PICTURE 9(8).
     02    PAREN33    PICTURE X VALUE ')'.
01   CALL-FUNC        PICTURE X(4) VALUE 'ISRT'.
```

```
LINKAGE SECTION.
01   PCBNAME.
     02   DBD-NAME            PICTURE X(8).
     02   SEG-LEVEL           PICTURE XX.
     02   STATUS CODE         PICTURE XX.
     02   PROC-OPTIONS        PICTURE XXXX.
     02   RESERVE-DLI         PICTURE S9(5) COMPUTATIONAL.
     02   SEG-NAME-FB         PICTURE X(8).
     02   LENGTH-FE-KEY       PICTURE S9(5) COMPUTATIONAL.
     02   NUMB-SENS-SEGS      PICTURE S9(5) COMPUTATIONAL.
     02   KEY-FB-AREA.
          03 ROOT-SEG-KEY     PICTURE X(6).
          03 SECOND-SEG-KEY   PICTURE X(4).
          03 THIRD-SEG-KEY    PICTURE X(8).
PROCEDURE DIVISION.
BEGIN.
     ENTER LINKAGE .
         ENTRY 'DLITCBL' USING PCBNAME .
     ENTER COBOL.
         MOVE SPACE TO PAREN3.
     ENTER LINKAGE.
         CALL 'CBLTDLI' USING
            CALL-FUNC,
            PCBNAME,
            IN-DEP2,
            SSA-1,
            SSA-2,
            SSA-3.
     ENTER COBOL.
```

```
ROOTbbbb(KEYbbbbbb=678512)
```

```
DEP1bbbb(KEY1bbbbb=3412)
```

```
DEP2bbbb(KEY2bbbbb=56744788)
```

RETURN FROM APPLICATION PROGRAM

COBOL

```
ENTER LINKAGE.
RETURN.
ENTER COBOL.
```

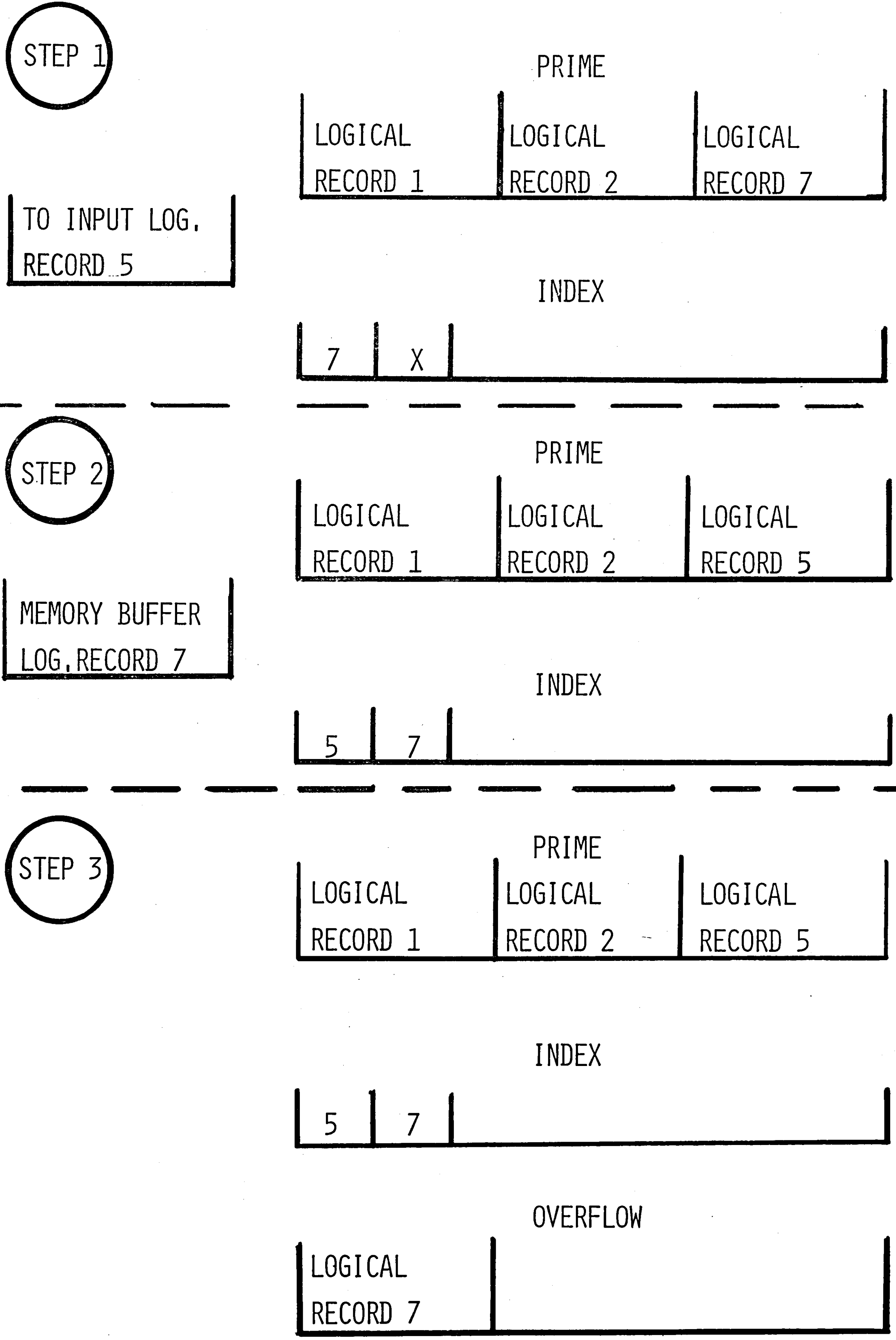
PL/I

```
RETURN:
```

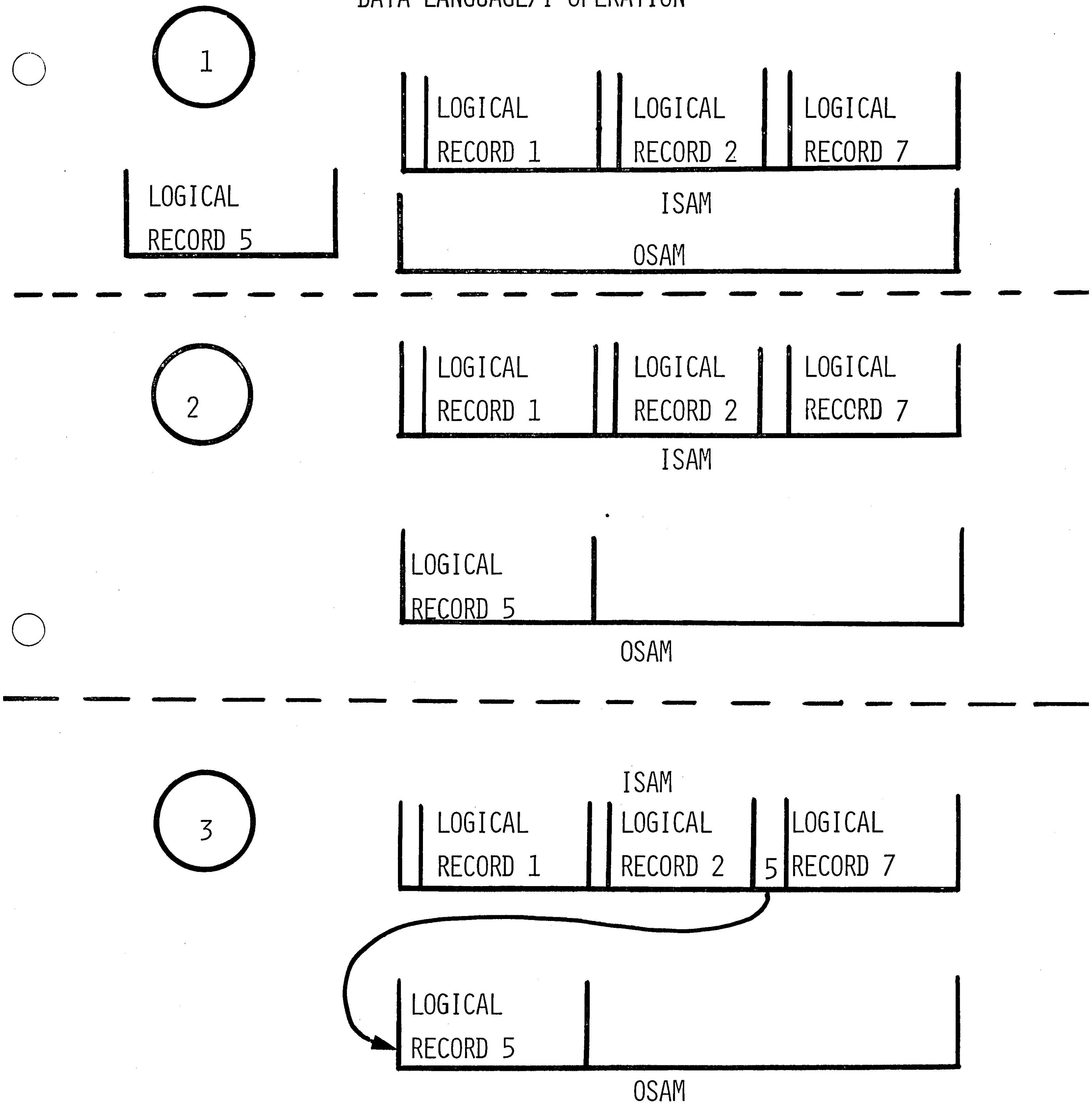
ASSEMBLER

```
RETURN (14,12),RC=0
```

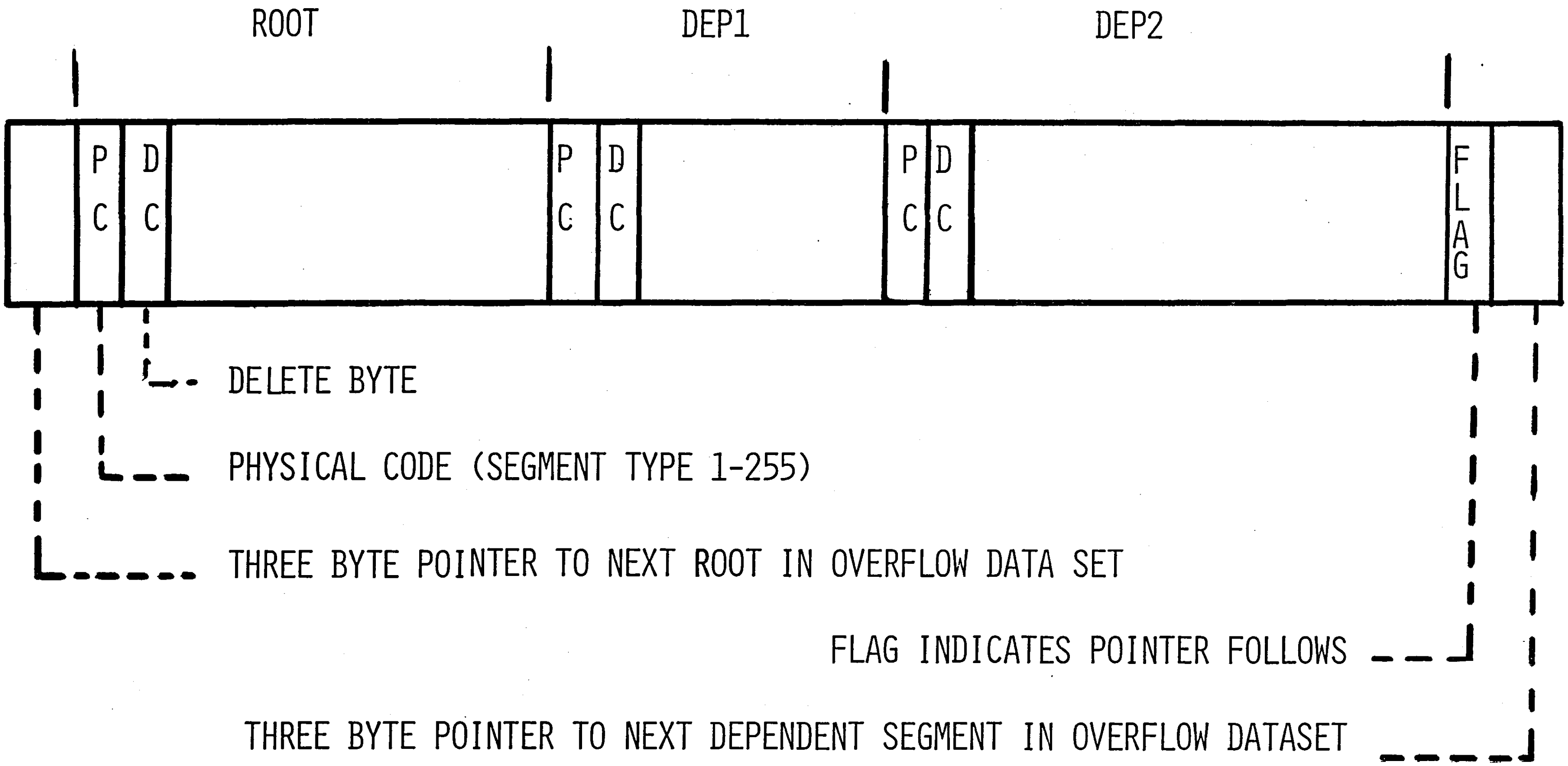
ISAM OPERATION



DATA LANGUAGE/I OPERATION



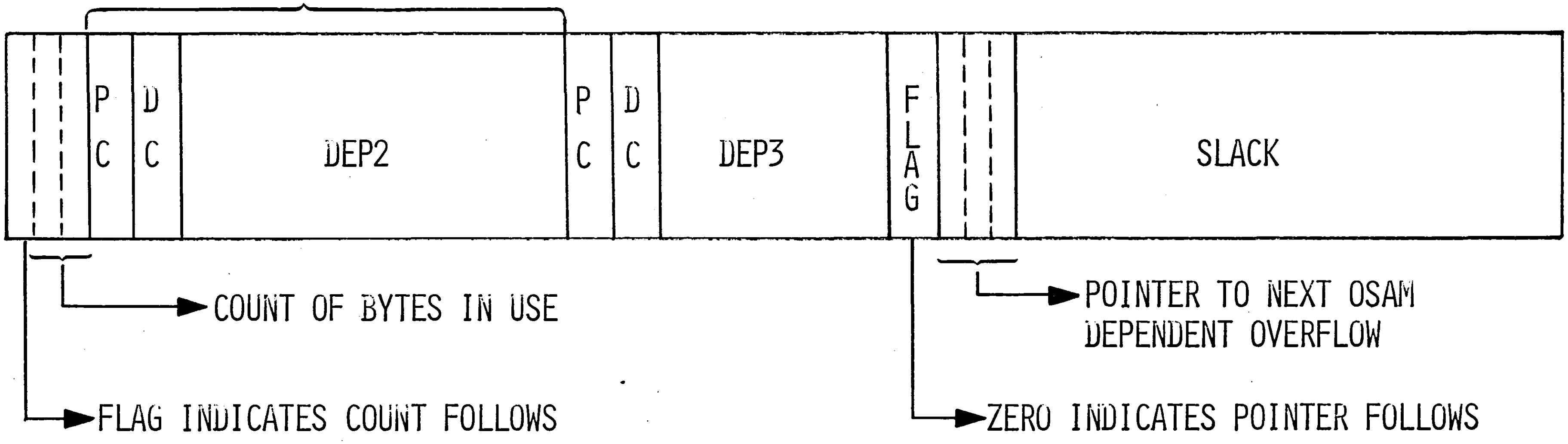
ISAM PHYSICAL STRUCTURE


ROOT
DEP1
DEP2
P C
D C
P C
D C
P C
D C
FLAG
DELETE BYTE
PHYSICAL CODE (SEGMENT TYPE 1-255)
THREE BYTE POINTER TO NEXT ROOT IN OVERFLOW DATA SET
FLAG INDICATES POINTER FOLLOWS
THREE BYTE POINTER TO NEXT DEPENDENT SEGMENT IN OVERFLOW DATASET

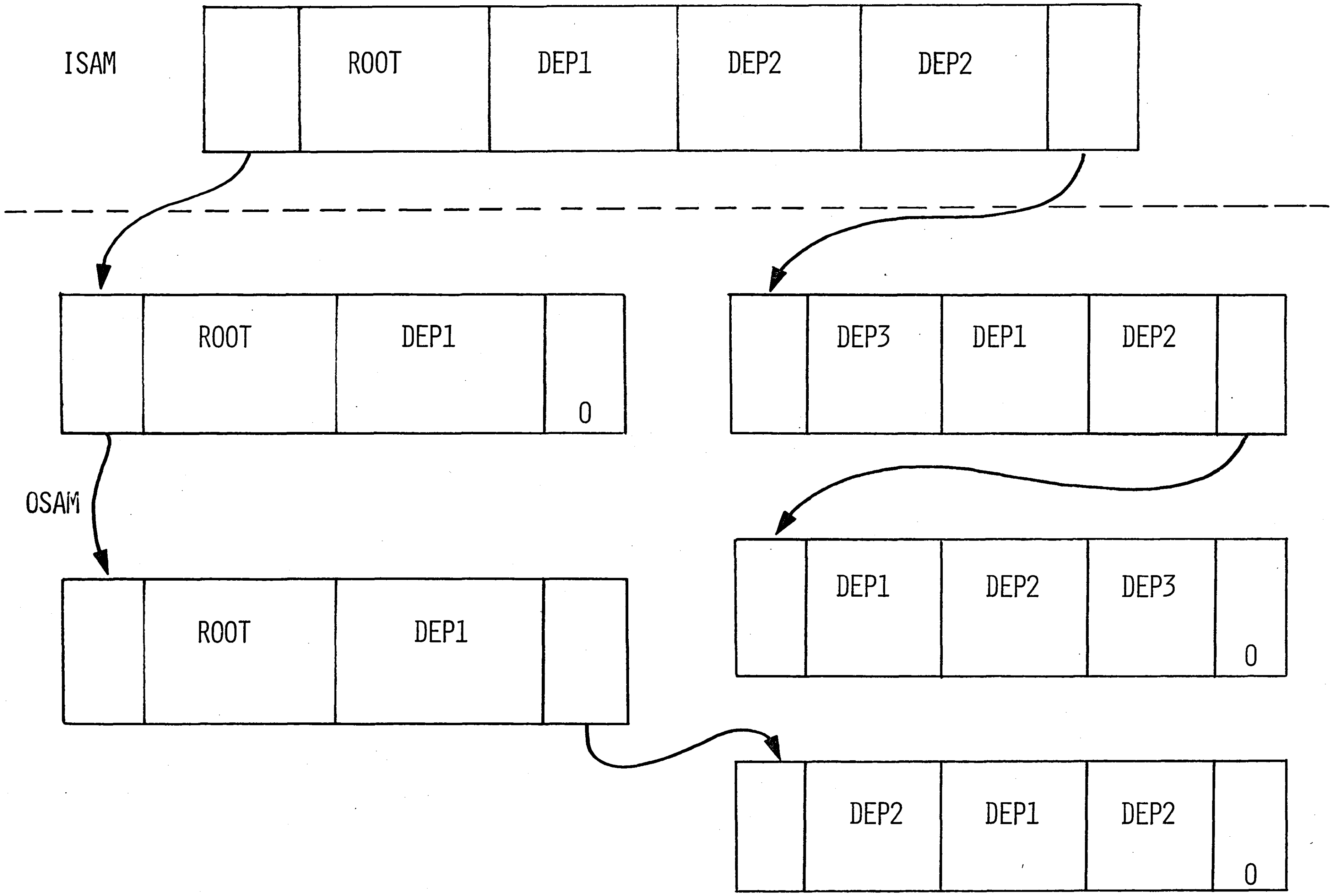
OSAM ROOT OVERFLOW HAS SAME FORMAT

4.3.84

OSAM-DEPENDENT SEGMENT OVERFLOW STRUCTURE

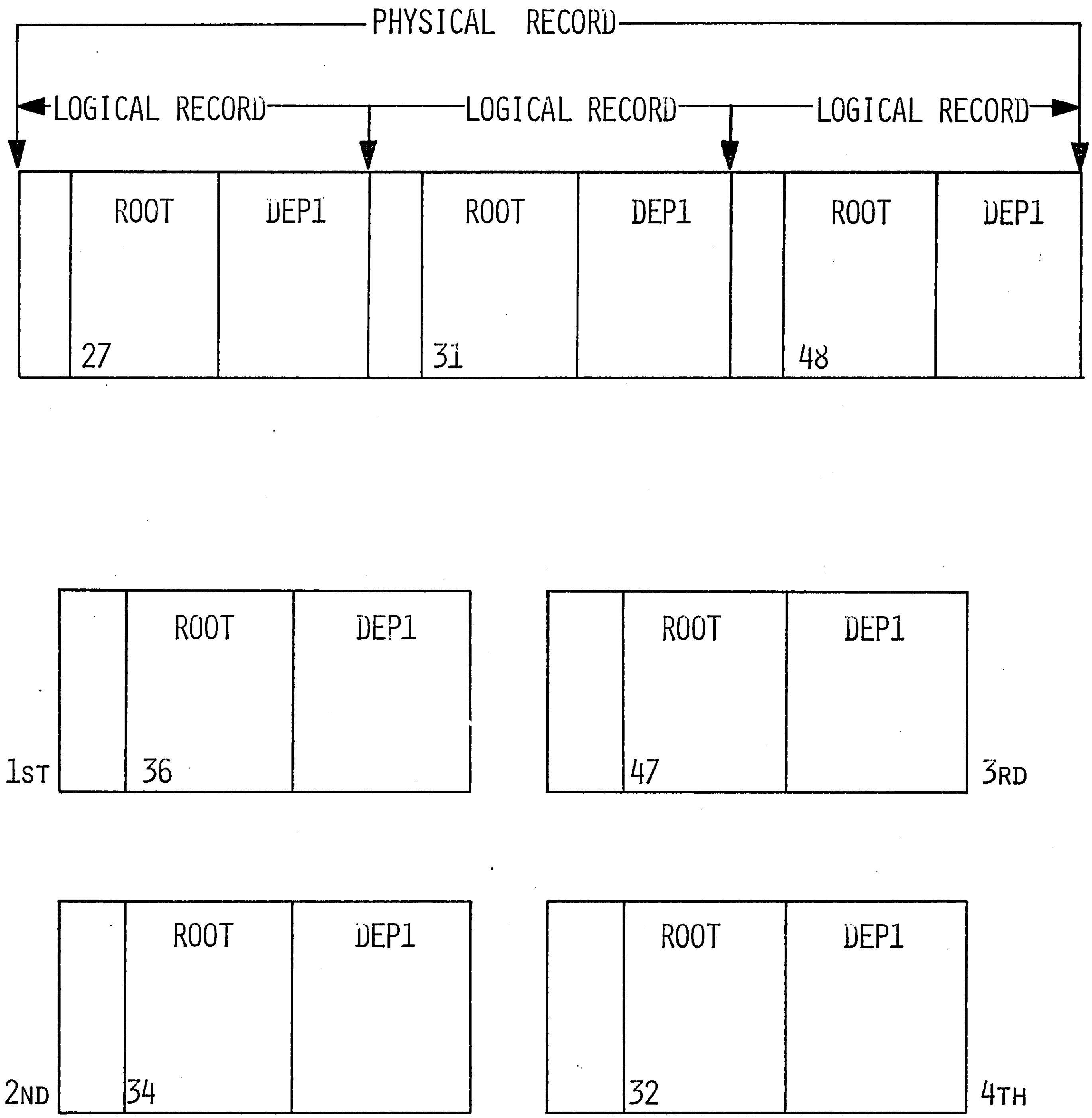

P C
D C
DEP2
P C
D C
DEP3
FLAG
SLACK
COUNT OF BYTES IN USE
FLAG INDICATES COUNT FOLLOWS
POINTER TO NEXT OSAM DEPENDENT OVERFLOW
ZERO INDICATES POINTER FOLLOWS

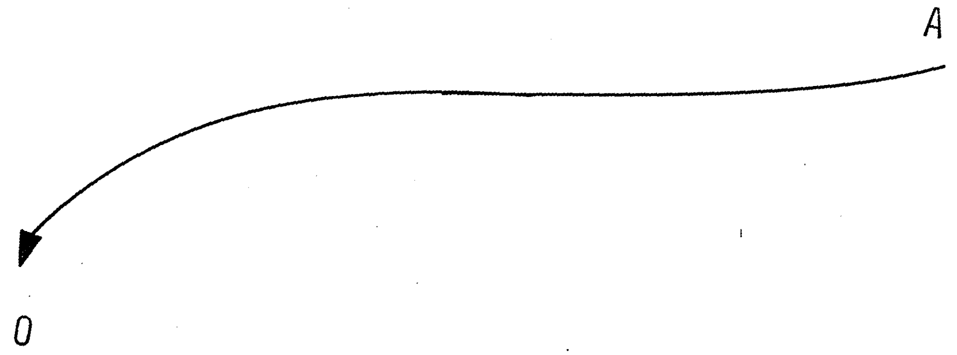
ISAM/OSAM RELATIONSHIP



4.3.86

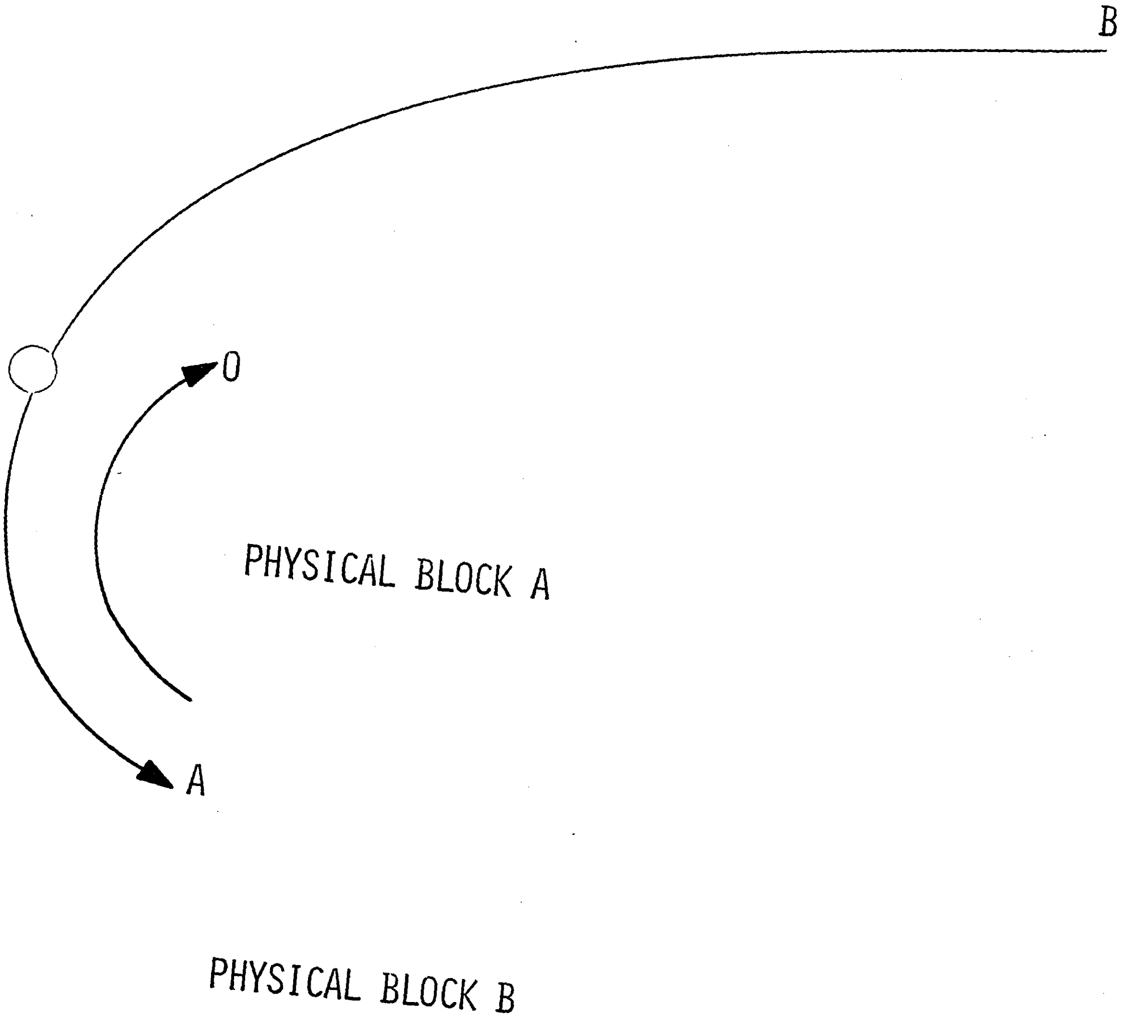
ROOT INSERTION

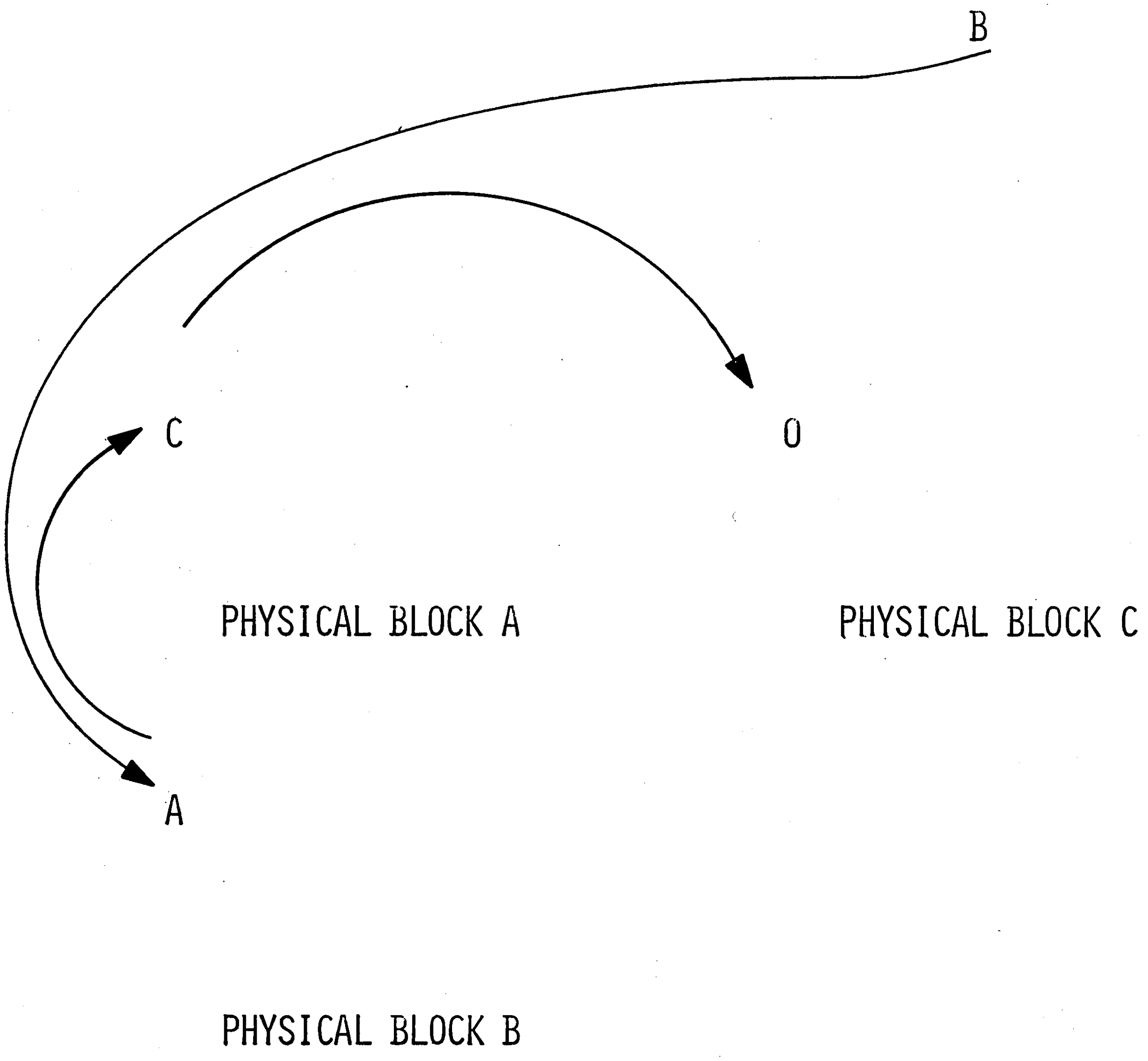


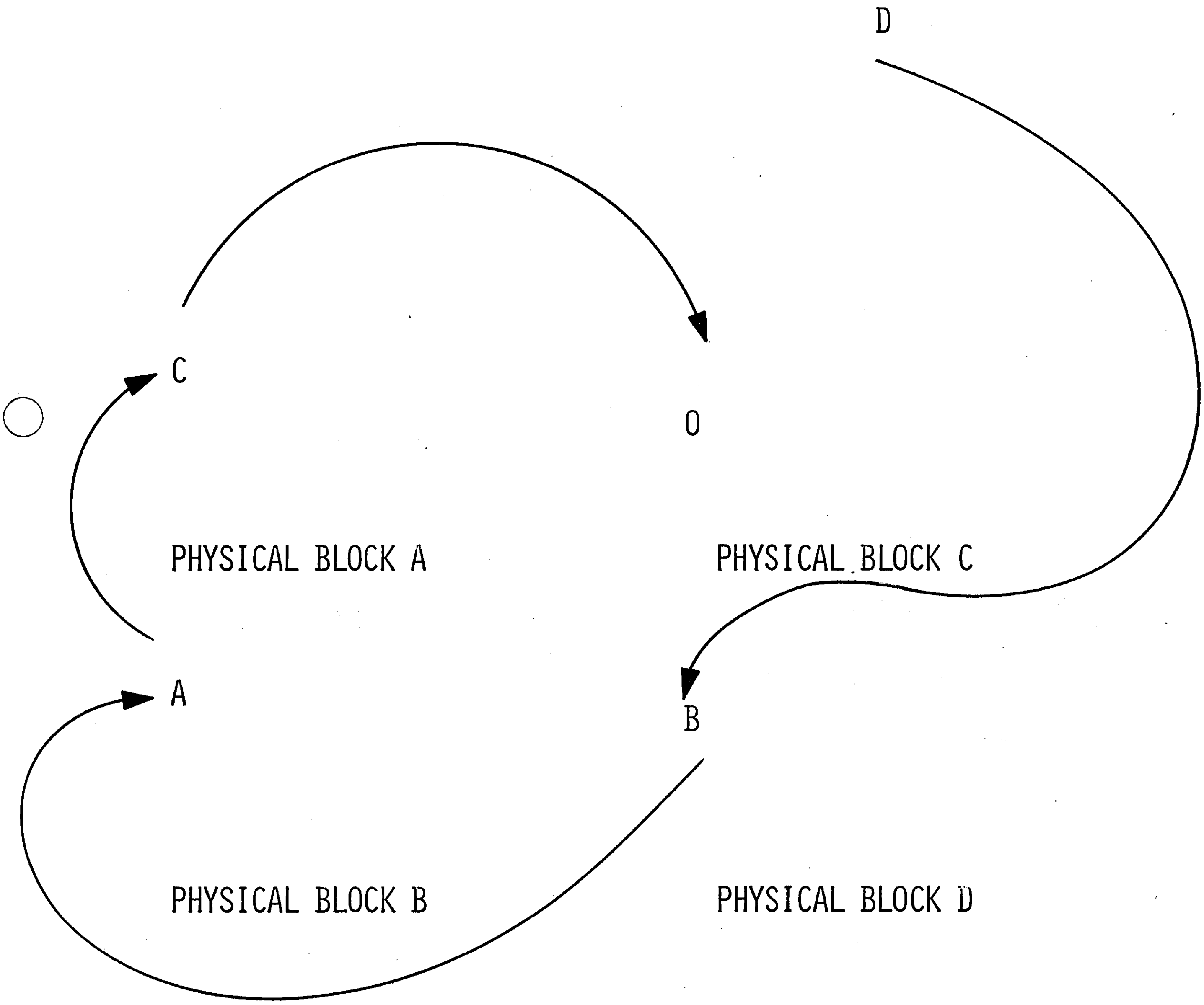


PHYSICAL BLOCK A

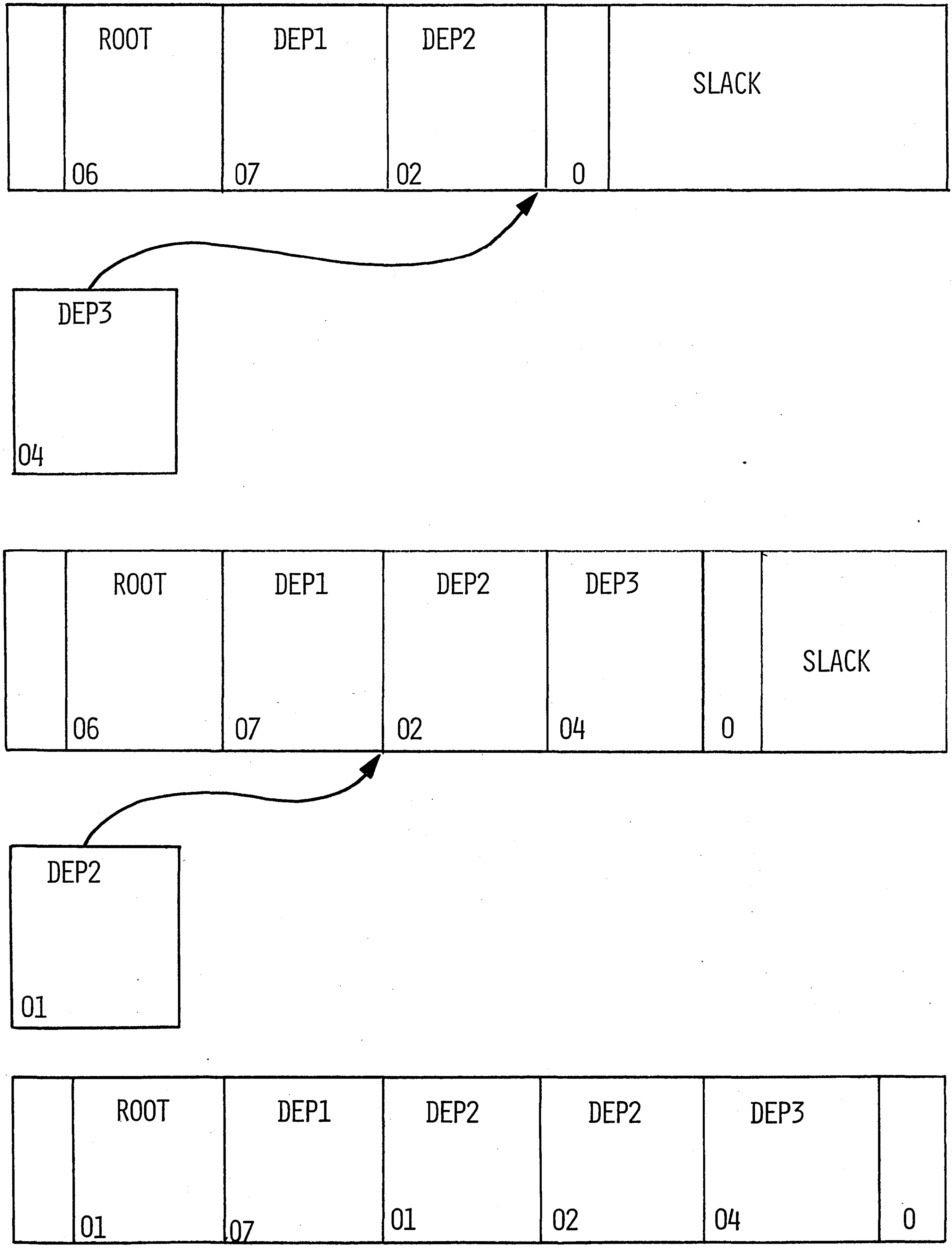
4.3.88


B
0
PHYSICAL BLOCK A
A
PHYSICAL BLOCK B

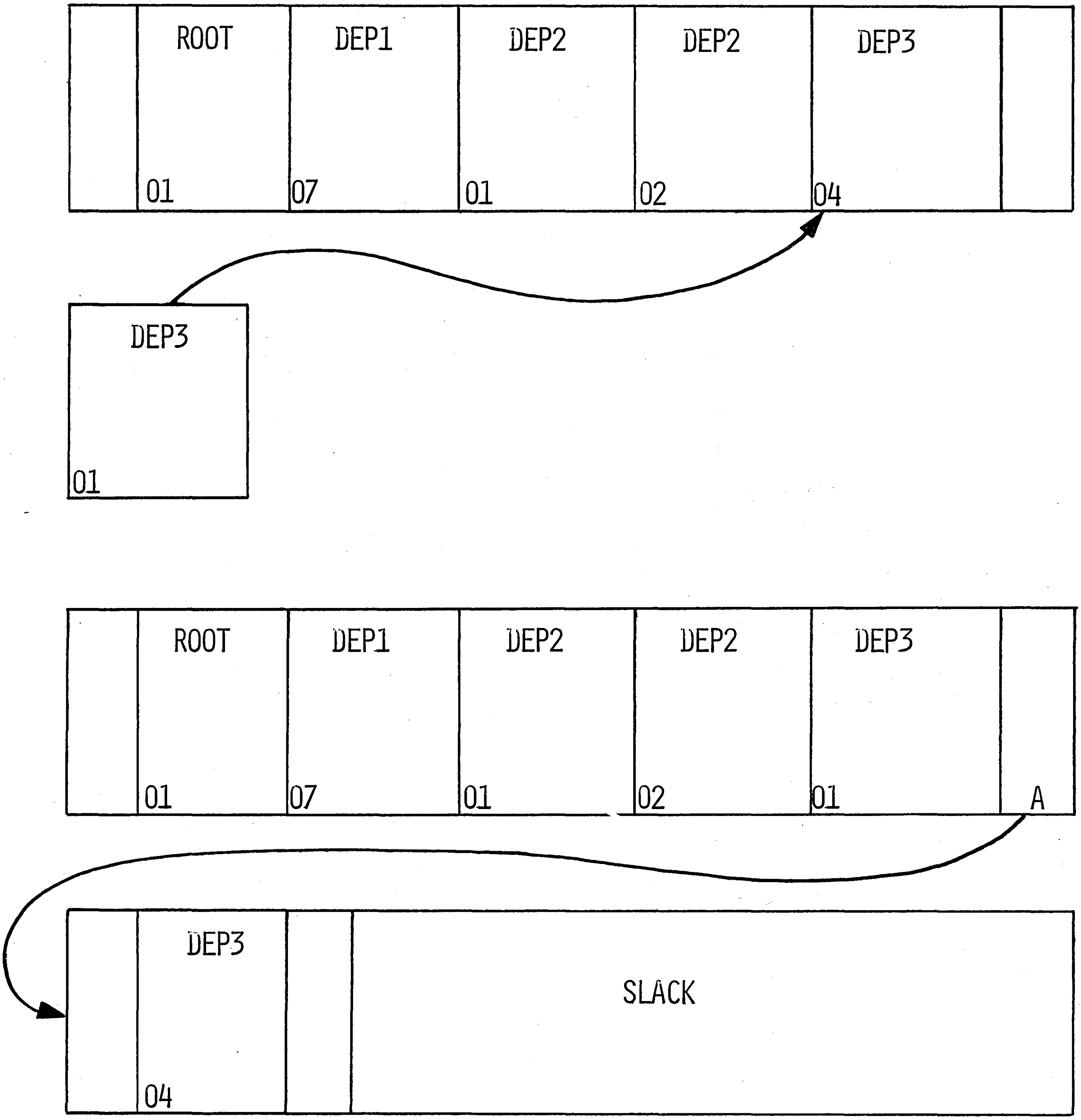

B
C
O
A
PHYSICAL BLOCK A
PHYSICAL BLOCK C
PHYSICAL BLOCK B


D
C
O
PHYSICAL BLOCK A
PHYSICAL BLOCK C
A
B
PHYSICAL BLOCK B
PHYSICAL BLOCK D

DEPENDENT SEGMENT INSERTION



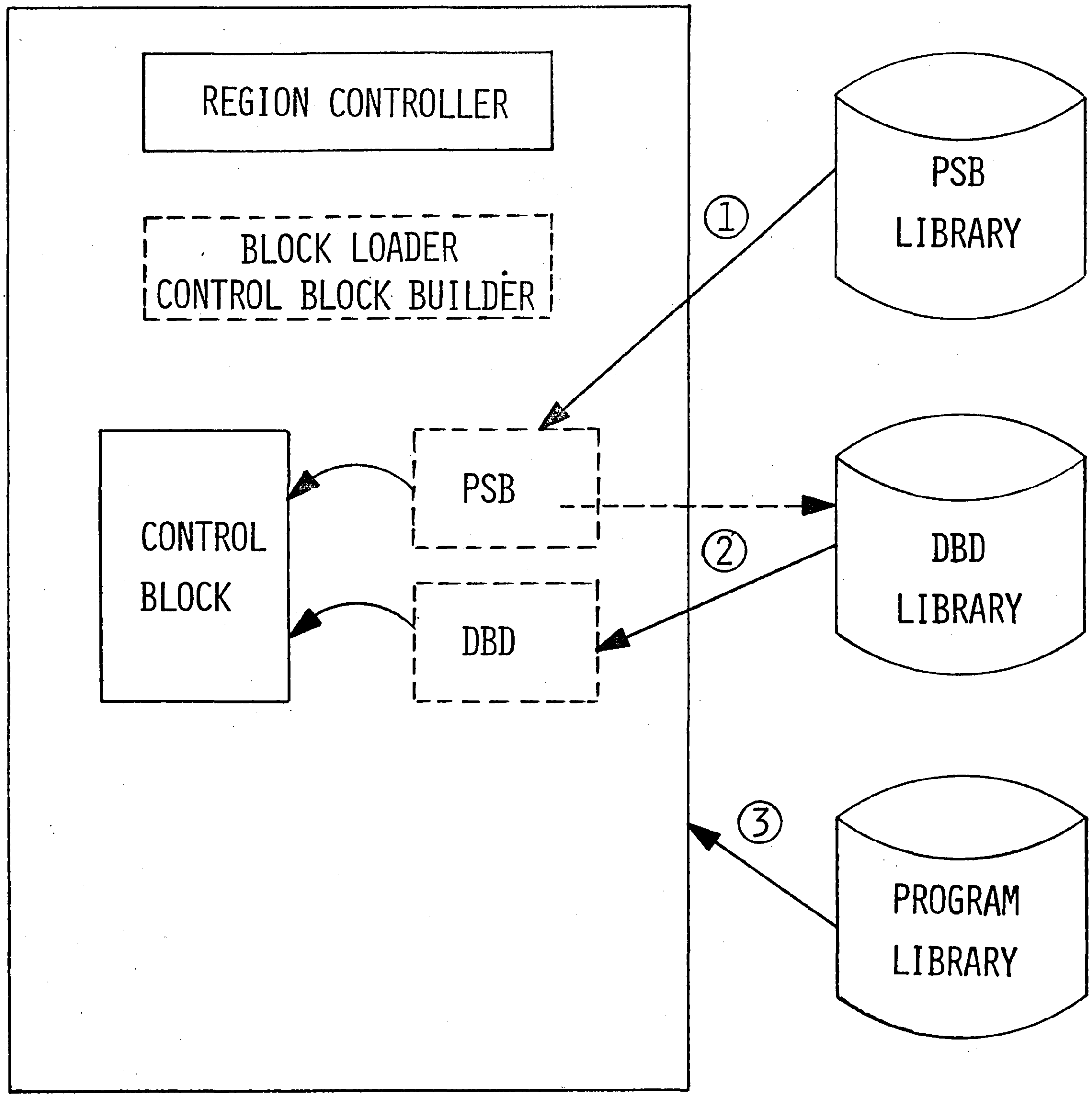
DEPENDENT SEGMENT INSERTION



PHYSICAL BLOCK A

LOADING A BATCH PROGRAM

```
//  EXEC DLIBATCH,PSB=PSBNAME
```



CLASS EXERCISES

INSTRUCTORS' NOTE:

Additional material can be found in Section 5 - Instructor Materials to help in assigning these exercises.

1. Write a data base description using the segment names, lengths, key fields, and key field lengths shown on the following page. The student should provide his own frequencies of occurrences.

   It is suggested that the following page be reproduced and given to the student. The only additional material required by the student is the JCL provided by the instructor.

2. Write a program specification block for the structure on the following page and declare sensitivity to all segments. The PCB Control Card should specify -- TYPE=DCB and PROCOPT=A. The DBDNAME should be provided by the instructor and should be the name of the data base to be used by the class when they write their processing programs.

   The PSBGEN Control Card should specify the language in which the student will later write his processing program to use this PSB. The PSBNAME will be the same as that on the JCL provided by the instructor for running the PSB generation.

   The student should be informed that this PSB will be used by him when his processing program is written later in the course.

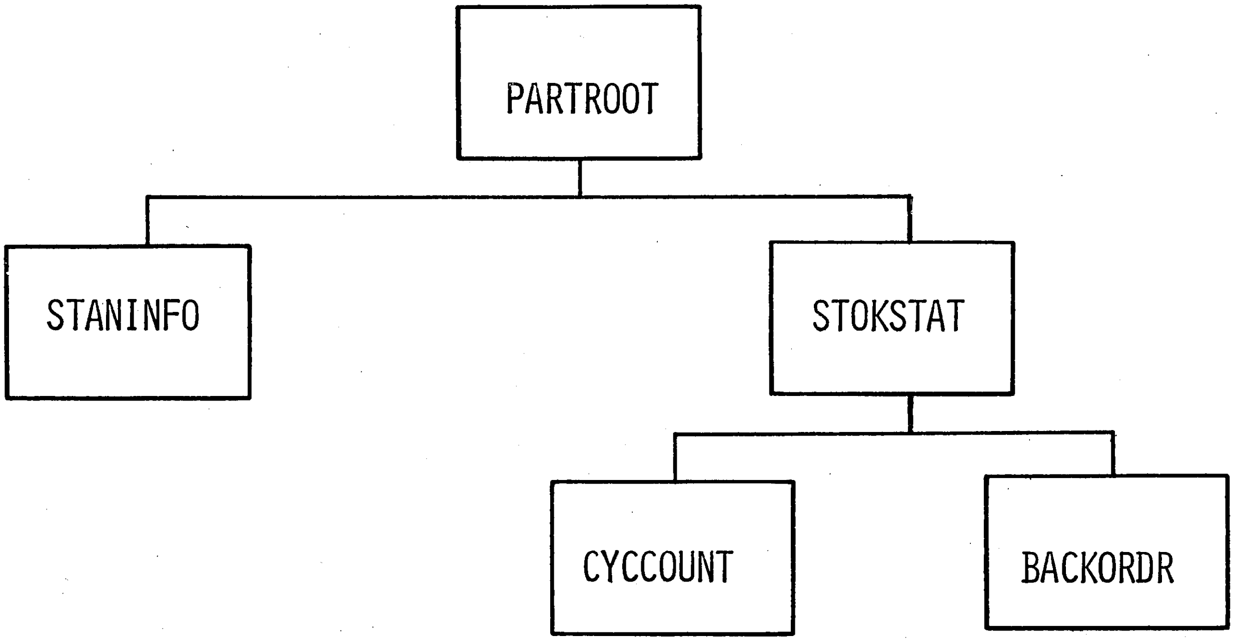
3. Given a program which retrieves and prints the information in the PARTROOT and STANINFO segments shown in the structure on the following page , modify the program to insert new PARTROOT and STANINFO segments, retrieve and print the newly added segments, and then delete them.

   The segment names, sizes, and key fields are those shown on the following page.

   Use the PSB which was generated for exercise 2 above.

   Additional material for assigning this problem can be found in Section 5.

DATA BASE STRUCTURE



SEGMENT NAME	LENGTH	KEY FIELD	LENGTH
PARTROOT	50	PARTKEY	17
STANINFO	85	STANKEY	2
STOKSTAT	160	STOCKEY	16
CYCCOUNT	25	CYCLKEY	2
BACKORDR	75	BACKKEY	10

NOTE: ALL KEY FIELDS ARE AT THE BEGINNING OF EACH SEGMENT.
ALL FIELD TYPES ARE C

ONLINE PROCESSING (MESSAGE PROCESSING)

Outline

Bibliography

Information Management System/360
Application Description Manual H20-0524
Program Description

ONLINE PROCESSING (Message Processing)

Objectives: Upon successful completion of this topic, the student is able to:

1. Convert a batch processing program to a message processing program.

2. Write Data Language/I calls for input messages and output messages to terminals.

3. Describe the necessary program interface to simulate a message processing region in a batch processing region.

4. State the significance of an input message editor.

A. The Online Environment

Teleprocessing with IMS/360 requires either MFT-II or MVT.

(V-1)

1. System structure and facilities

(V-1A)

2. Scheduling a message processing program

B. Preparing the Application Program

1. Required additions to the batch processing program are an input/output PCB and Data Language/I calls for input and output.

2. The input/output PCB is not specified at PSB generation time but is generated internally by IMS/360 when an application program is scheduled.

(V-2)

3. The one-to-one relationship between the PCBs generated at PSB generation time and the PCBs in the linkage section of the COBOL program exists but in an indirect manner.

(V-3)

4. Additional PCBs may be added for outputting to terminals other than the inputting terminal. If output is for other than the input terminal, another terminal PCB must be present. The name of the I/O PCB cannot be altered in the COBOL program.

5. If the output is to be processed by another message processing program, an alternate PCB is used for output. The destination is not a logical terminal name but rather a transaction code.

(V-4)

6. Format of the terminal PCB at PSB generation time.

(V-5)

7. Format of the terminal PCB mask in the COBOL program.

(V-6)

C. Calling for an input message

1. The first line of a message is obtained with a GET UNIQUE call. No SSAs are allowed.

2. Subsequent lines of a message are obtained with a GET NEXT call. No SSAs are allowed.

3. The GET UNIQUE call is used to obtain the first line of the message. A GET NEXT call after the last segment of a message has been obtained will result in the returning of a status code indicating this condition.

4. Structure of the calls for messages.

(V-7)

D. Outputting a response

1. A Data Language/I INSERT call is used to enqueue a line of an output message. No SSAs are allowed.

2. Each line of the output message should be terminated with a carriage return character. A decimal 21 (CR).

3. More than one line of output can be placed in the output area before making the INSERT call.

4. Multiple INSERT calls given in succession will result in one message only.

(V-8)

(V-9)

5. Status codes for Data Language/I message calls.

(V-10)

E. Message Formats

1. The input message has three fields. The first field is a half-word binary field containing the total number of characters in the message line including all three fields. The maximum count is 136.

2. The second field of the input message is a half-word which is reserved by IMS/360.

3. The TEXT portion of the message is the message exactly as it was entered from the terminal. This includes the transaction code, the message text, and the carriage return character.

   If the message consists of multiple lines of text, each subsequent line has the same format.

   The transaction code appears only in the first line.

   If a password is entered with the message, it is edited out before getting to the application program. A blank is placed between the transaction code and the first text character.

   The only two acceptable delimiters for the transaction code are a blank or a left parenthesis.

4. The format of an output message is the same as the input message format but the contents of the text portion are different.

   No logical terminal name is included in the output message. The destination is determined from the PCB.

5. In COBOL, the count field is supplied by the application programmer and is equal to the length of the TEXT portion plus 4.

6. The two byte field following the count is reserved for Data Language/I use and must be binary zeros.

7. Device control characters may be inserted into the message where it is desired to format the message at the terminal output device.

   Idle characters are automatically supplied for tabs and carriage returns.

8. Passing a message from one application program to another has the same format as a message to an output terminal. The destination is obtained through the PCB reference.

   Password security is not available to a program-to-program message.

(V-11)

F. Message Processing Region Simulation

1. Message processing region simulation is not supplied as a part of the distributed IMS/360 program.

2. The checkout of any message processing program in the online terminal environment is often impractical.

3. A technique for simulation is presented here. Minimal change is required when converting the application program to a message processing program. An example can be found in the IMS/360 Program Description Manual.

4. The simulation is accomplished by writing two interfaces. The first interface (A) is used to modify the PCB parameters which are passed to Data Language/I. The second interface (B) is used to simulate message input and output.

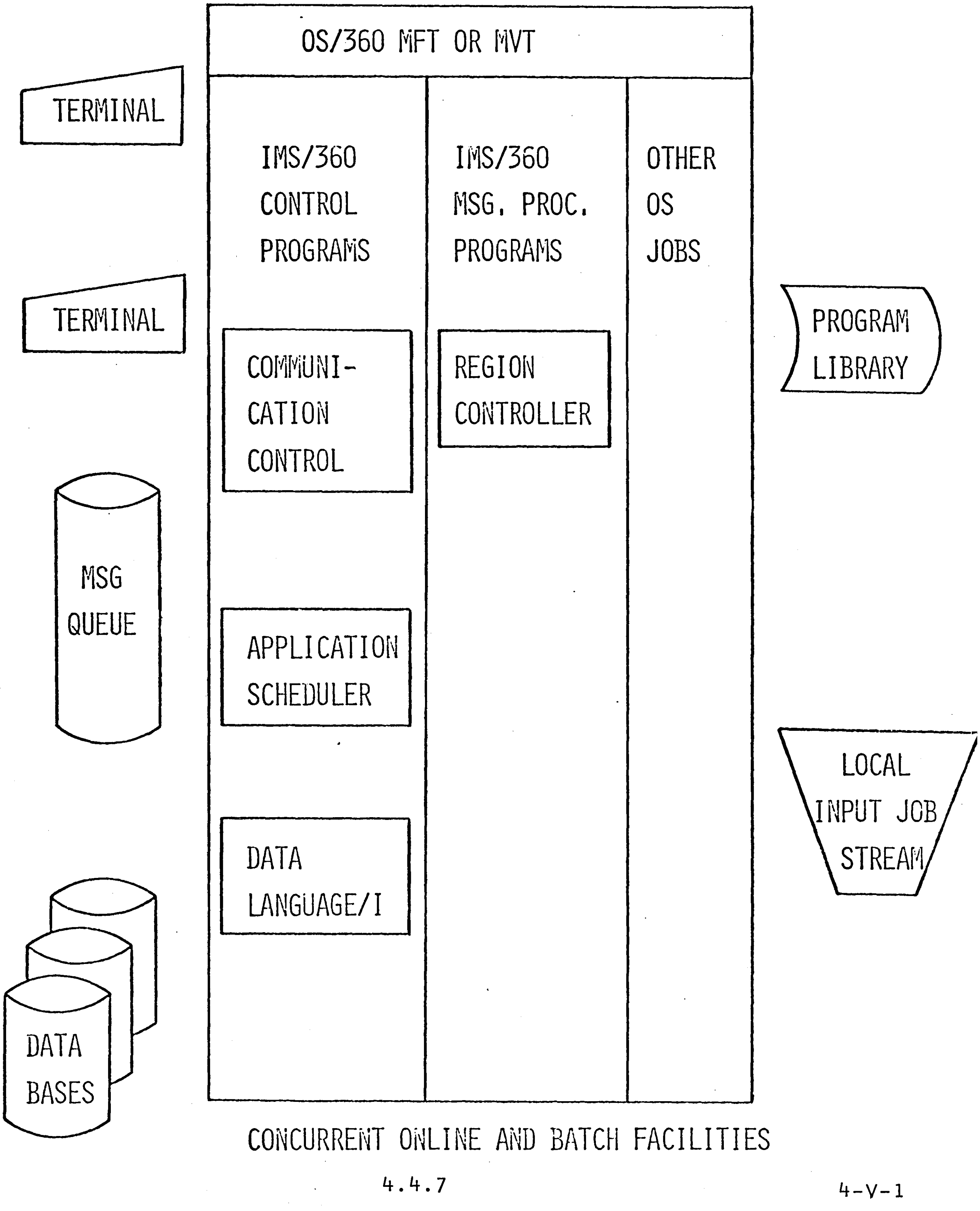
G. An Input Message Editor

1. In order to allow freedom in entering a message at a terminal, it may be desirable to write an input message editor.

2. The input message editor could accept a relatively free form input and convert it to a number of fixed length fields to be operated on by a COBOL or PL/I program.

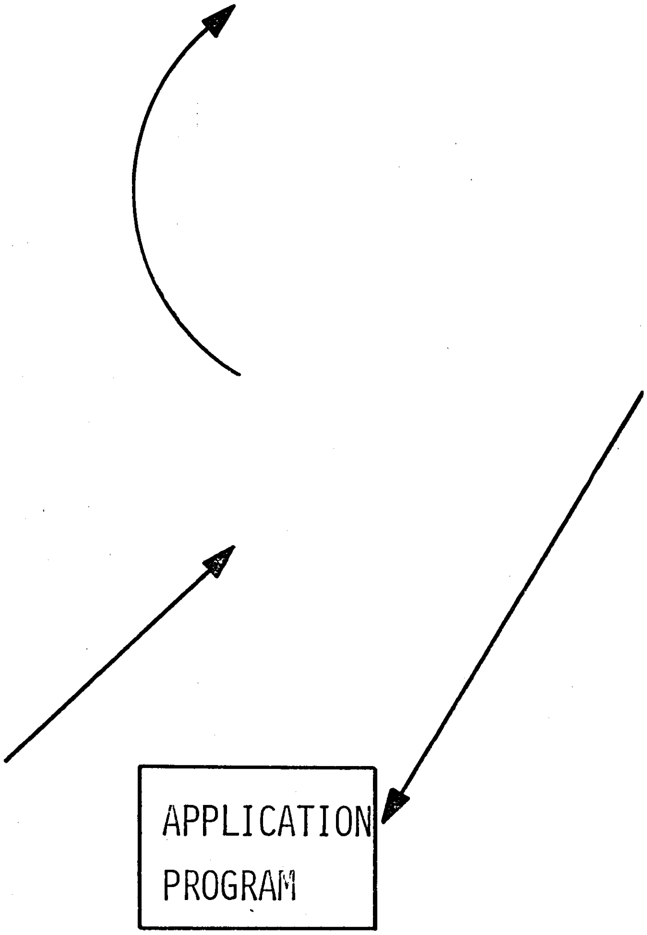
3. A complete guide as to how to write an input message editor can be found in the Program Description Manual.

IMS/360 SYSTEM ORGANIZATION

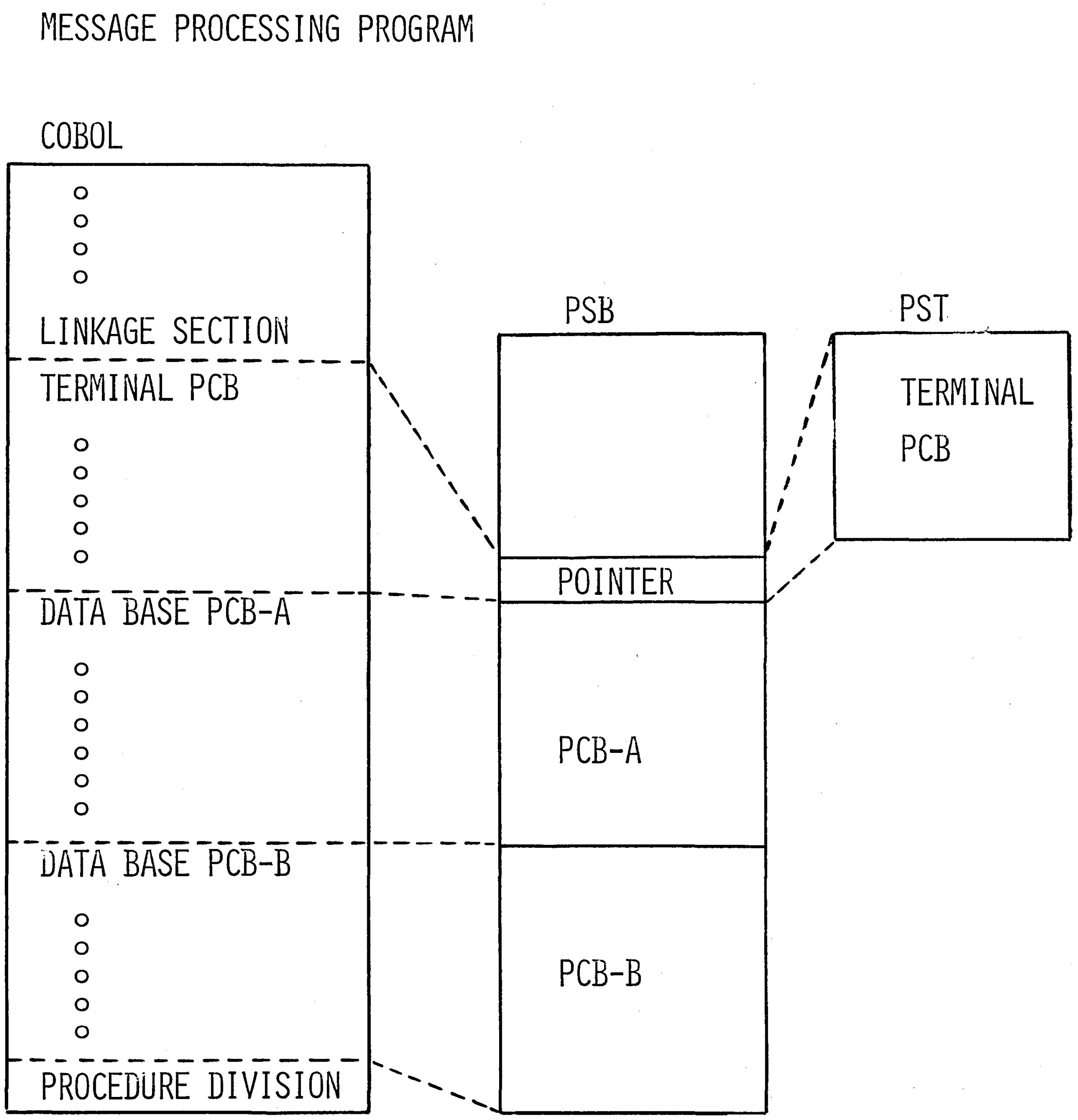


CONCURRENT ONLINE AND BATCH FACILITIES

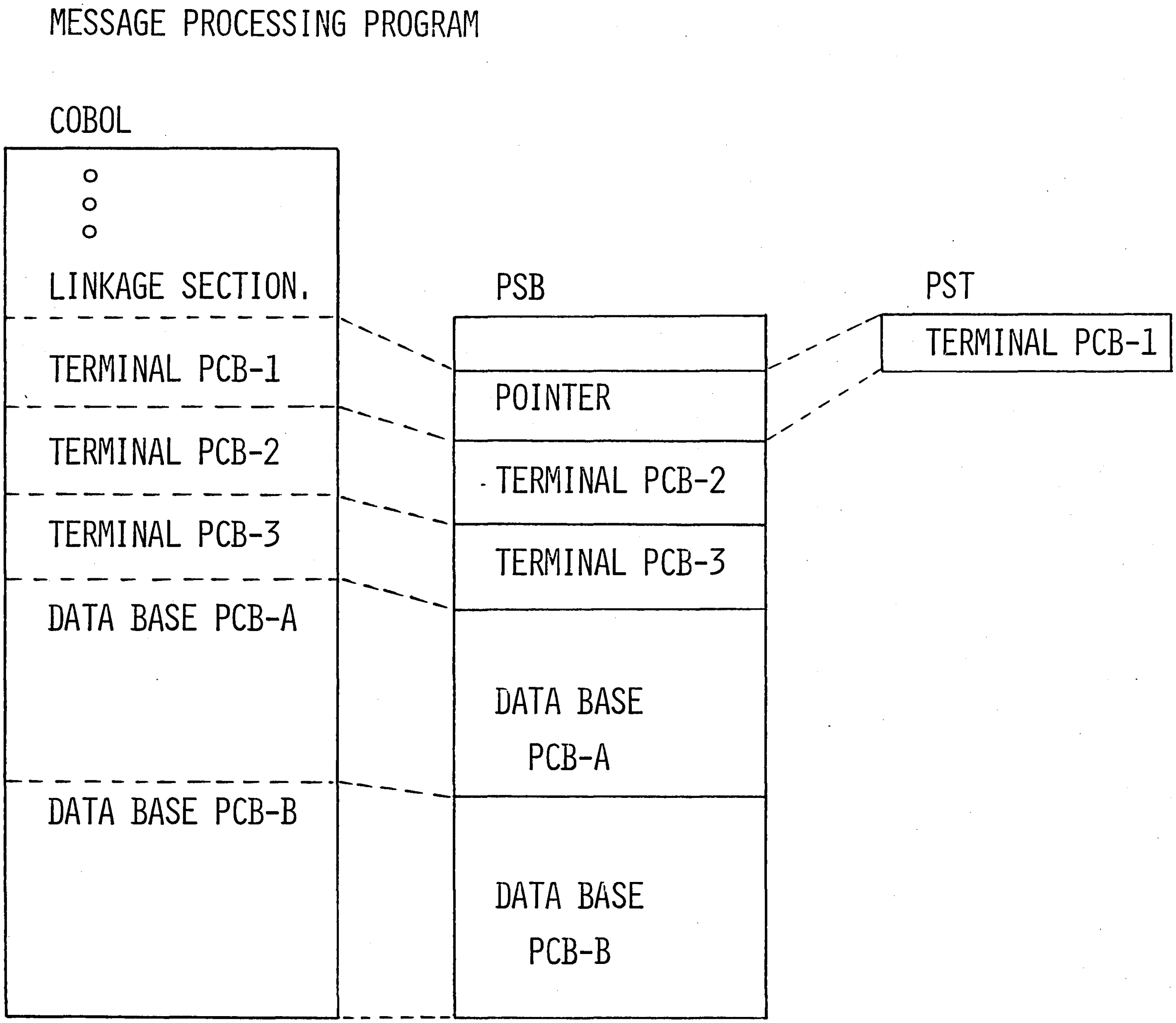
4.4.7


APPLICATION
PROGRAM

I/O PCB


MESSAGE PROCESSING PROGRAM
COBOL
LINKAGE SECTION
TERMINAL PCB
DATA BASE PCB-A
DATA BASE PCB-B
PROCEDURE DIVISION
PSB
POINTER
PCB-A
PCB-B
PST
TERMINAL
PCB

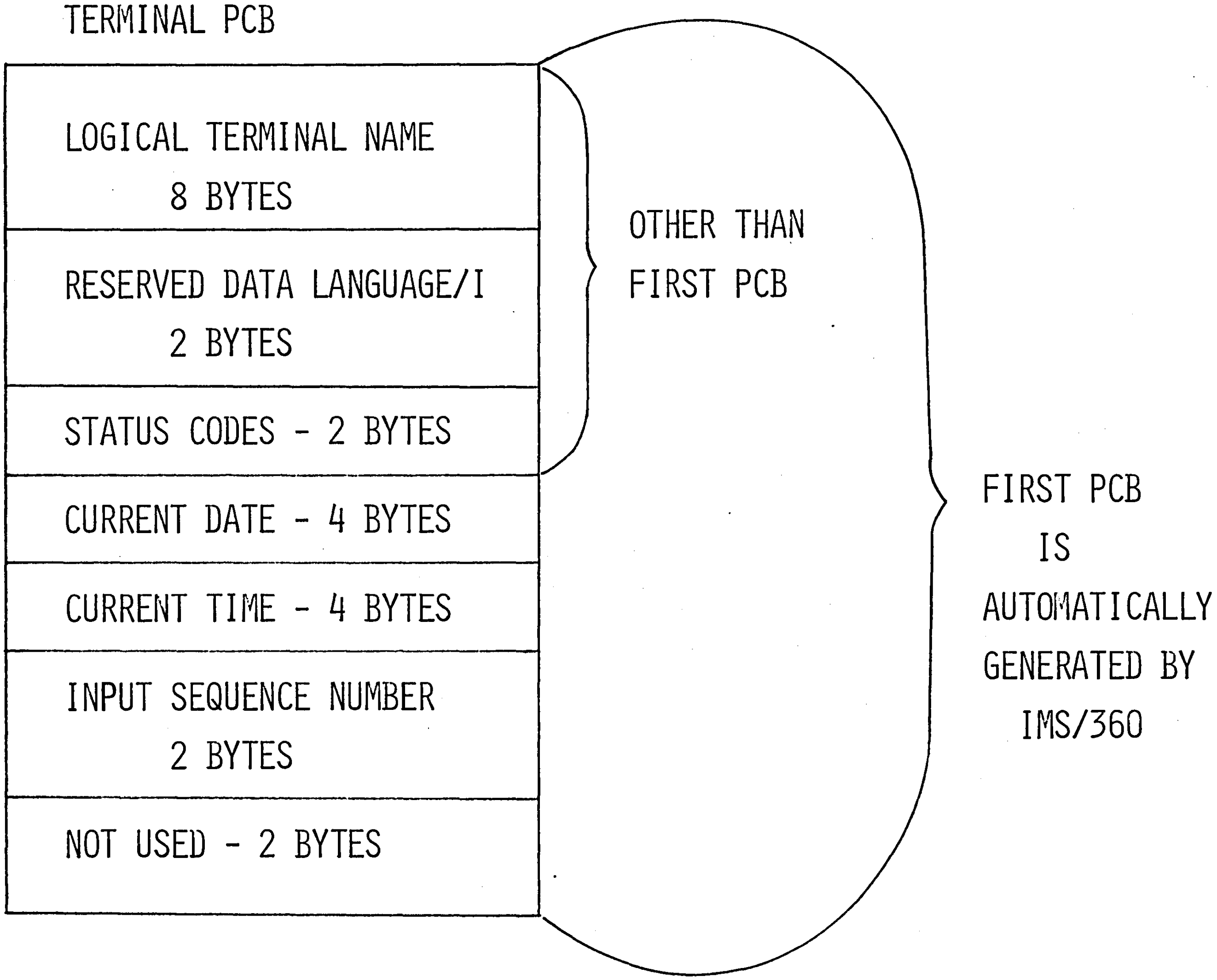
ALTERNATE OUTPUT PCB



PCB FORMAT

PCB TYPE=TP, LTERM= [TRANSACTION CODE / LOGICAL TERMINAL NAME]

TERMINAL PCB



LOGICAL TERMINAL NAME 8 BYTES

RESERVED DATA LANGUAGE/I 2 BYTES

STATUS CODES - 2 BYTES

CURRENT DATE - 4 BYTES

CURRENT TIME - 4 BYTES

INPUT SEQUENCE NUMBER 2 BYTES

NOT USED - 2 BYTES

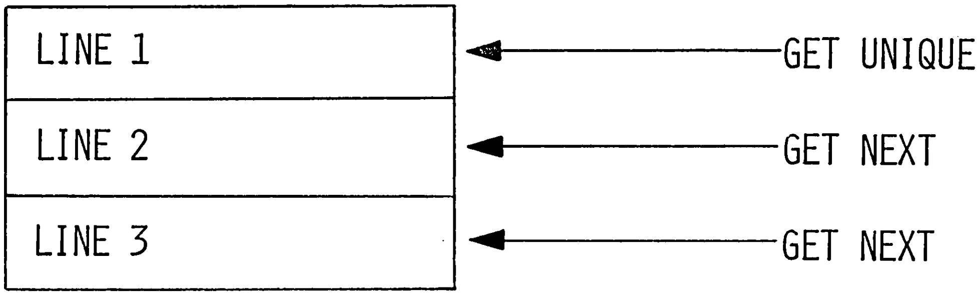
OTHER THAN FIRST PCB

FIRST PCB IS AUTOMATICALLY GENERATED BY IMS/360

```
LINKAGE SECTION.
    01   I-O TERM.
         02   LTERM-NAME       PICTURE X(8).
         02   DLI-RESERVE      PICTURE XX.
         02   STATUS-CODE      PICTURE XX.
         02   DATE-TIME.
              03   JULIAN-DATE PICTURE S9(7)   COMPUTATIONAL-3.
              03   TIME-OF-DAY PICTURE S9(7)   COMPUTATIONAL-3.
              03   MSG-SEQ     PICTURE S9(3)   COMPUTATIONAL.
              03   FILLER      PICTURE XX.
    01   ALT  TERM-A.
         02   LTERM-NAME-A     PICTURE X(8).
         02   DLI-RESERVE-A    PICTURE XX.
         02   STATUS-CODE-A    PICTURE XX.
    01   ALT  TERM-B.
         02   LTERM-NAME-B     PICTURE X(8).
         02   DLI-RESERVE-B    PICTURE XX.
         02   STATUS-CODE-B    PICTURE XX.
    01   DATABASE.
         02   DBASE-NAME       PICTURE X(8).
          o
          o
          o
          o
          o
    PROCEDURE DIVISION.
```

MESSAGE CALLS

MESSAGE A

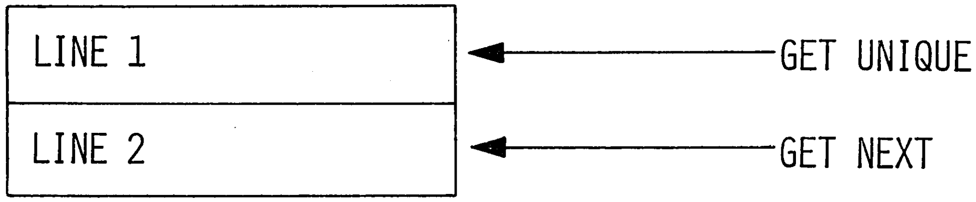


LINE 1 ← GET UNIQUE

LINE 2 ← GET NEXT

LINE 3 ← GET NEXT

MESSAGE B



LINE 1 ← GET UNIQUE

LINE 2 ← GET NEXT

```
ENTER LINKAGE.

     CALL 'CBLTDLI' USING FUNCTION, TERM-PCB-IN,
         MSG-SEG-IO-AREA .
```

INSERTING AN OUTPUT MESSAGE

FIELD	CONTENTS
FIELD-A	NO STOCK ON HAND$_{\underline{\text{CR}}}$
FIELD-B	BACK ORDERS ARE PRESENT$_{\underline{\text{CR}}}$
FIELD-C	THE NEXT SCHEDULED ARRIVAL IS XX-XX-XX $_{\underline{\text{CR}}}$

```
CALL  'CBLTDLI' USING INSERT-FUNC, TERM-PCB, FIELD-A.
CALL  'CBLTDLI' USING INSERT-FUNC, TERM-PCB, FIELD-B.
CALL  'CBLTDLI' USING INSERT-FUNC, TERM-PCB, FIELD-C.
```

THESE THREE CALLS CREATE ONE OUTPUT MESSAGE

FIELD	CONTENTS
FIELD-D	NO STOCK ON HAND$_{\underline{\text{CR}}}$ BACK ORDERS ARE PRESENT$_{\underline{\text{CR}}}$ THE NEXT SCHEDULED ARRIVAL IS XX-XX-XX.$_{\underline{\text{CR}}}$

```
CALL  'CBLTDLI' USING INSERT-FUNC, TERM-PCB, FIELD-D.
```

THIS CALL CREATES ONE MESSAGE OF THREE LINES

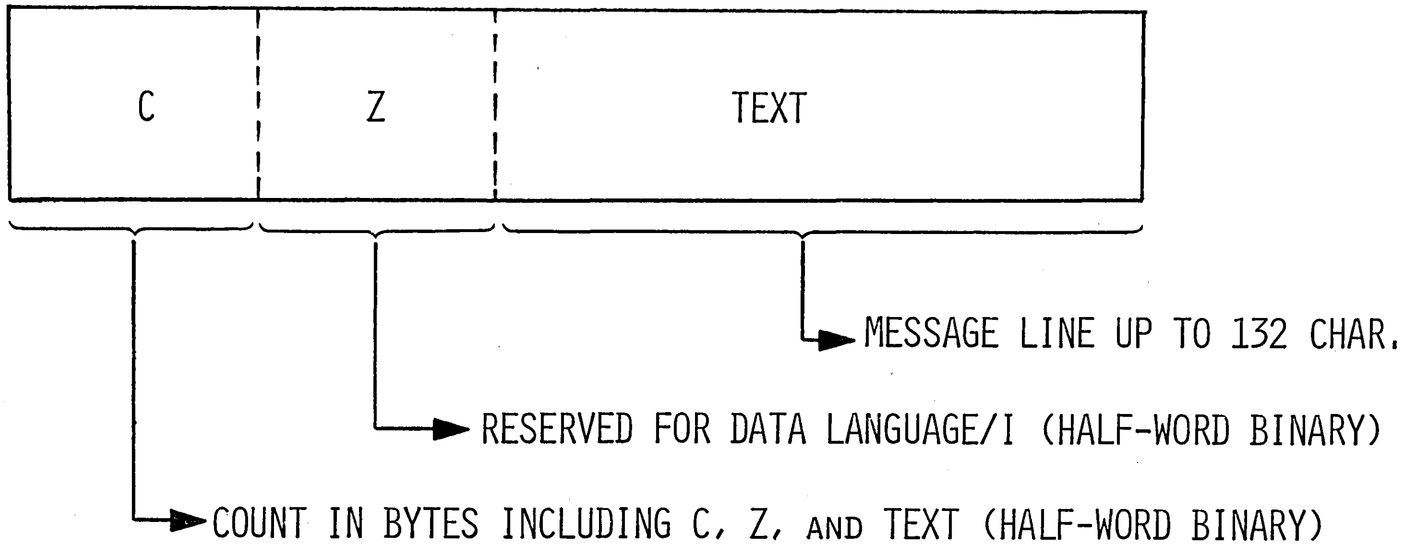
INPUT MESSAGE CALLS

STATUS CODE	MEANING
AB	NO SEGMENT I/O AREA IN CALL
AQ	READ I/O ERROR. MESSAGE CHAIN CANNOT BE FOLLOWED. MINIMUM OF ONE LOST MESSAGE
AR	READ I/O ERROR. MESSAGE SEGMENT HAS BEEN LOST. MESSAGE CHAIN IS STILL INTACT.
AS	QUEUES NOT AVAILABLE
QC	NO MORE INPUT MESSAGES
QD	NO MORE SEGMENTS FOR THIS MESSAGE
QE	GET NEXT REQUEST BEFORE GET UNIQUE
QG	QUEUE MANAGER ERROR
QI	GET NEXT AFTER END OF MESSAGE
QJ	UNKNOWN SYSTEM ERROR

INSERT MESSAGE CALLS

STATUS CODE	MEANING
AB	NO SEGMENT I/O AREA IN CALL
AP	I/O ERROR IN OSAM
QF	SEGMENT LESS THAN FIVE CHARACTERS (SEG LENGTH IS MSG TEXT PLUS FOUR CONTROL CHARACTERS)
QG	QUEUE MANAGER ERROR
QH	TERMINAL SYMBOLIC ERROR - OUTPUT DESIGNATION UNKNOWN TO IMS/360 (LOGICAL TERMINALS OR TRANSACTION CODE)

FORMAT OF A MESSAGE


C
Z
TEXT
MESSAGE LINE UP TO 132 CHAR.
RESERVED FOR DATA LANGUAGE/I (HALF-WORD BINARY)
COUNT IN BYTES INCLUDING C, Z, AND TEXT (HALF-WORD BINARY)

TRANSACT (PASSWORD) THIS IS THE MESSAGE TEXT

TEXT

TRANSACT THIS IS THE MESSAGE TEXT

TRANS THIS IS THE MESSAGE TEXT

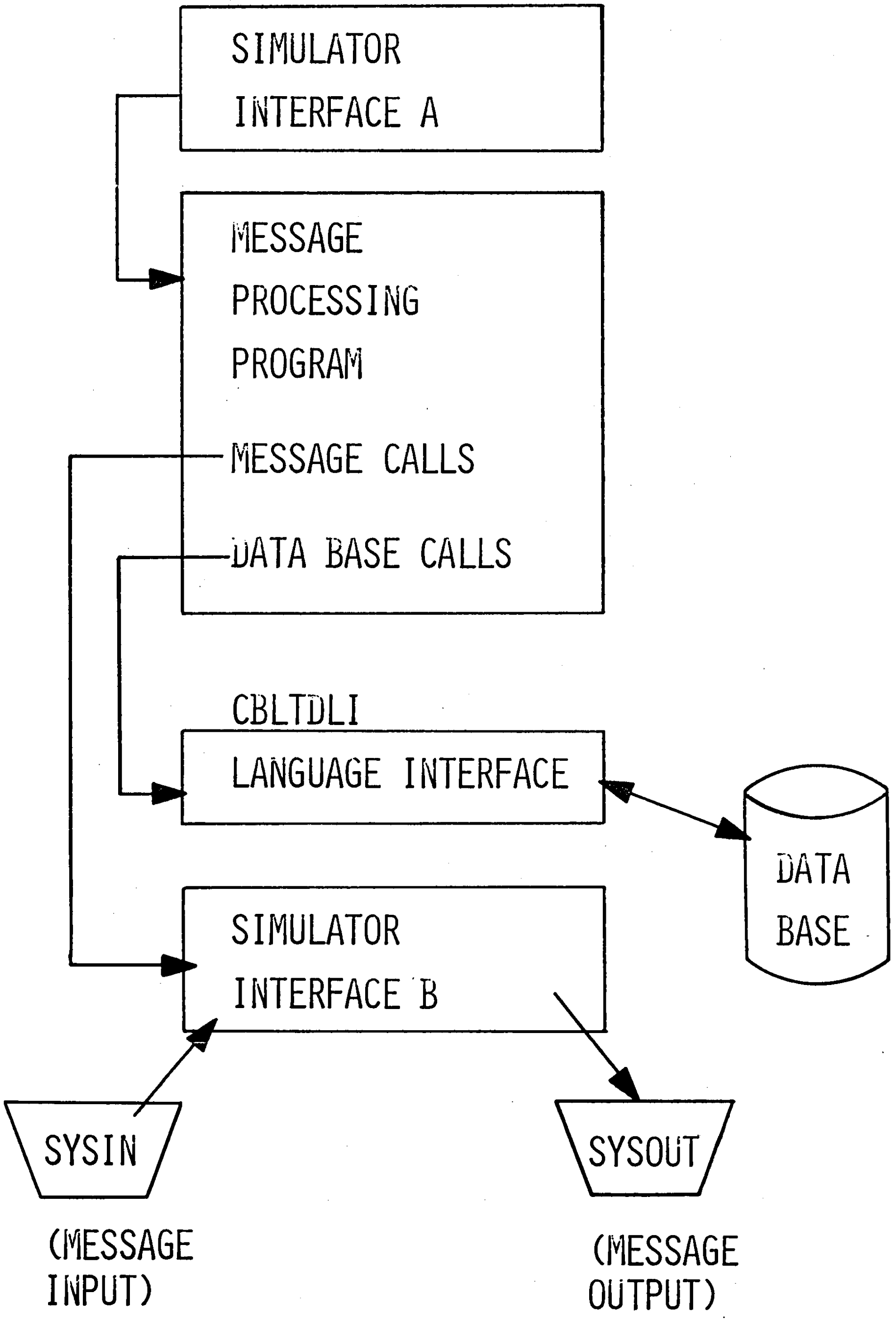
TEXT

TRANS THIS IS THE MESSAGE TEXT

MESSAGE PROCESSING REGION SIMULATION

ENTRY:

DLITCBL



CLASS EXERCISES

INSTRUCTORS' NOTE:

See Section 5 - Instructor Materials for additional information on this exercise.

Convert the batch program written in the last module to a message processing program. Do not change any existing logic or data base calls. Do not include an alternate PCB.

FUNCTIONAL DESCRIPTION

Outline

Bibliography

Information Management System/360
Application Description Manual H20-0524
System Operations Manual

IMS/360 FUNCTIONAL DESCRIPTION OF FACILITIES

Objectives: Upon completion of this topic, the student is able to:

1. Descrbie the process of IMS/360 system definition.

2. Describe the logging process of IMS/360 and the reports produced by the statistical utilities.

3. Describe the checkpoint, restart, and data base dump and recovery facilities of IMS/360.

4. Describe the security maintenance facilities of IMS/360.

(V-1)

A. System Definition

The IMS/360 system definition is similar to OS/360 generation.

1. Macro-instruction control cards describing the user's IMS/360 system are input to Stage 1.

2. Output from Stage 1 is a set of punched control cards describing a series of jobs which are input to Stage 2.

3. Output from Stage 2 is the IMS/360 system.

B. IMS/360 System Log

The system recorder is designed to facilitate the placing of data on the system log. The information is used primarily for checkpoint/restart and data base recovery functions.

(V-2)

1. Some information is written for restart

   a. Message queue control blocks

   b. Checkpoint data, consisting of dynamic fields in IMS control blocks which vary as a result of normal processing or master terminal commands.

c. OS/360 data set open or close

d. Changes to a data base on an optional basis (adds, deletes, and updates)

2. For restart and statistics

a. All messages received from terminals

b. All messages sent to terminals or programs

3. For statistics only

a. Error segments from or to terminals

b. At completion of sending a record to a terminal

c. Application accounting record

d. IMS/360 Accounting Record

4. Records are written to the log using QSAM.

(V-3)

5. Statistical utilities are provided to process log tape.

(V-4)

6. Types of statistical reports

(V-5 thru V-10)

7. Examples of statistical reports

(V-11)

C. Checkpoint, Restart, Data Base Dump, and Recovery

There are four checkpoint commands and two data base dump commands.

1. The simple checkpoint command causes the IMS/360 control blocks and tables which control the system to be written to the log tape.

2. The CHECKPOINT FREEZE causes orderly shutdown of communications, stops program scheduling, dumps control blocks, and terminates IMS/360.

3. The CHECKPOINT DUMPQ causes the same action as the CHECKPOINT FREEZE, and, in addition, dumps the input and output message queues to the log tape.

4. The CHECKPOINT PURGE requires the longest to shut down as all input messages in the system at the time of the request are processed and all output messages are sent to their destinations if possible. The control blocks are then written to the log tape.

5. The DBDUMP command causes the data base to be dumped to tape after transactions which update it are stopped.

6. The DBDUMP STOP is a command which prepares the system for data base reconstruction.

7. There are two types of restart and a data base recovery command.

   a. NRESTART is used to initially start the system and, after a normal CHECKPOINT command has been used, to shut the system down.

   b. ERESTART, or emergency restart, is used after a system failure such as the loss of core or the loss of message queues.

   c. DBRECOVERY is used to reprocess transactions against a damaged data base which has been rebuilt from a previously dumped copy.

(V-12)

D. Security Maintenance

The IMS/360 system definition supplies no password security capabilities.

1. The terminal and password security is assigned and changed through a utility program.

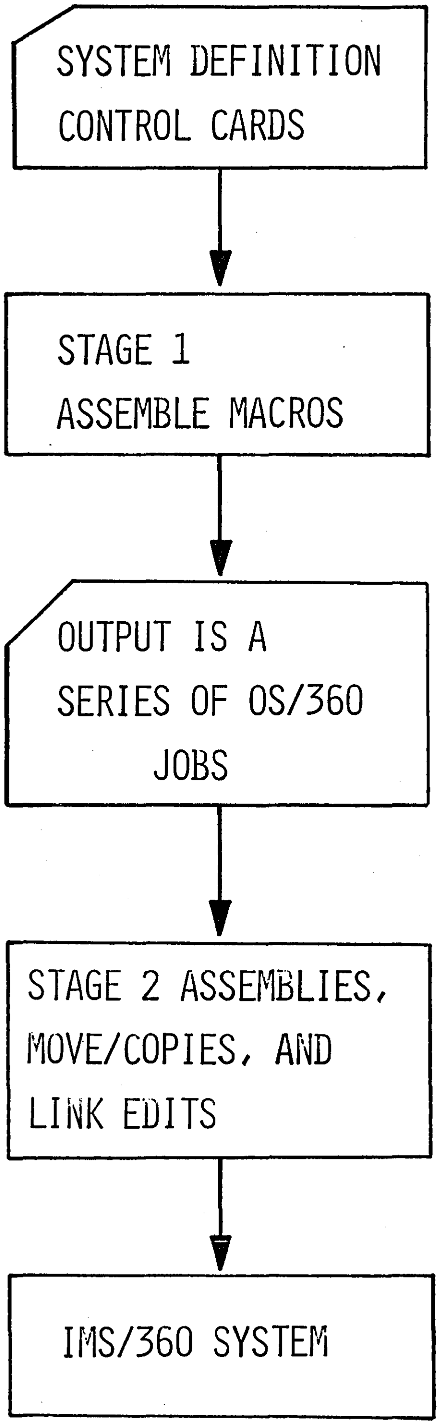
2. This structure allows the security information to be changed without a new IMS/360 generation.

(V-13)

3. Through use of the security maintenance program (SMP), passwords can be changed or assigned for transaction codes, terminal command verbs, program status changes,

data base status changes, and logical and physical terminal status changes.

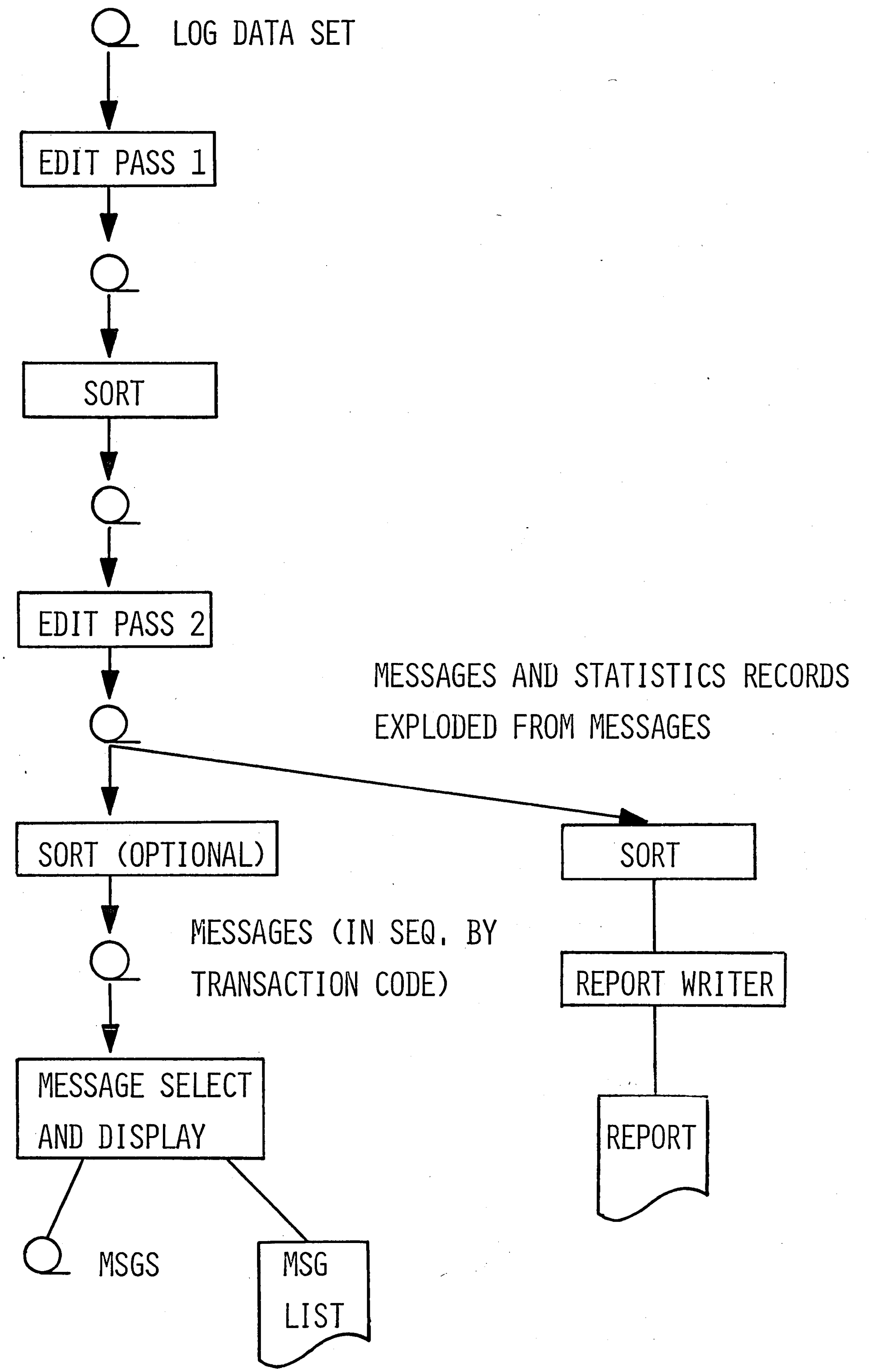
SYSTEM DEFINITION FLOW



IMS/360 SYSTEM LOG ENTRIES

FOR RESTART	WHEN WRITTEN
● MESSAGE QUEUE CONTROL BLOCKS	WHEN CHANGED
● CHECKPOINT DATA	AT TIME OF CHECKPOINT
● OS/360 DATA SET OPEN OR CLOSE	WHEN DATA SET OPENED OR CLOSED
● CHANGES TO A DATA BASE	INSERT, DELETE, OR REPLACE AGAINST A DATA BASE
FOR RESTART AND STATISTICS	
● MESSAGE FROM TERMINAL	WHEN MESSAGE COMPLETE
● MESSAGE TO TERMINAL OR PROGRAM	WHEN MESSAGE COMPLETE
FOR STATISTICS ONLY	
● ERROR SEGMENTS	WHEN HARDWARE ERROR
● COMPLETION OF SEND RECORD	COMPLETION OF SENDING
● APPLICATION ACCOUNTING RECORD	TERMINATION OF APPL. PROGRAM
● IMS/360 ACCOUNTING RECORD	IMS/360 IS STARTED OR STOPPED

PROCESSING THE LOG



TYPES OF STATISTICAL REPORTS

- MESSAGES QUEUED BUT NOT SENT -- BY TERMINAL
- LINE AND TERMINAL LOADING BY TIME OF DAY
- ERROR REPORTS ON BAD TRANSMISSION
- TRANSACTION REPORT
- TRANSACTION RESPONSE REPORT
- APPLICATION ACCOUNTING REPORT
- IMS/360 ACCOUNTING REPORT

```
                  IMS ACCOUNTING REPORT             DATE 1-22-68



IMS  CPU TIME FOR DAY 1/20/68 IS 07H 47M 07.9S OR 28,027.9S

IMS  CPU TIME FOR DAY 1/21/68 IS 06H 30M 29.5S OR 23,429.5S

IMS  CPU TIME FOR DAY 1/22/68 IS 07H 40M 39.5S OR 27,639.5S

IMS  CPU TOTAL TIME           IS 21H 58M 16.9S OR 79,096.9S
```

```
                  APPLICATION ACCOUNTING REPORT          DATE 01/02/68

                                              DATA BASE COUNTS
PROGRAM  TRANSACTION MESSAGE - - - - COUNTS                MOVE   BAD      TOT MESS   AVR
NAME      NAME      PRI  QTY   GU   GN  ISRT   GU   GN     CALL   CC       CPU TIME   TIME

PSB00001  TRANS001  01    71  142   14   71    81   42       1     1       10.65S     0.15S
                    02    81  162   16   81    91   31       1     0       12.15S     0.15S
                    **   152  304   30  152   172   73       2     1       22.80S     0.15S
     SYSTEM TOTAL        152  304   30  152   172   73       2     1
```

4.5.11

```
                    TRANSACTION RESPONSE REPORT             DATE 08-31-67


   TYPE       TOTAL      LONGEST   95%       75%       50%       25%       SHORTEST
TRANSACTION  RESPONSES  RESPONSE  RESPONSE  RESPONSE  RESPONSE  RESPONSE  RESPONSE


T231T05M        25     05M 30.0S 05M 00.0S 03M 00.0S 02M 20.0S 01M 00.0S    40.0S
T2359ALL         5         20.0S     15.0S      8.0S      6.0S      4.0S     3.0S
```

TRANSACTION REPORT

TRANSACTION		TOTAL	TOTAL	AVG	HOURLY	DISTRIBUTION	
CODE	R/S	MESSAGES	CHARACTERS	SIZE	00-07	07-08	08-09
T21CAS1R	R	5	250	50	0	1	1
T21CAS2S	S	15	1250	83	5	2	3
SYSTEM	S	15	1250	83	5	2	3
TOTALS	R	5	250	50	0	1	1

LINE AND TERMINAL REPORT DATE 2/11/69

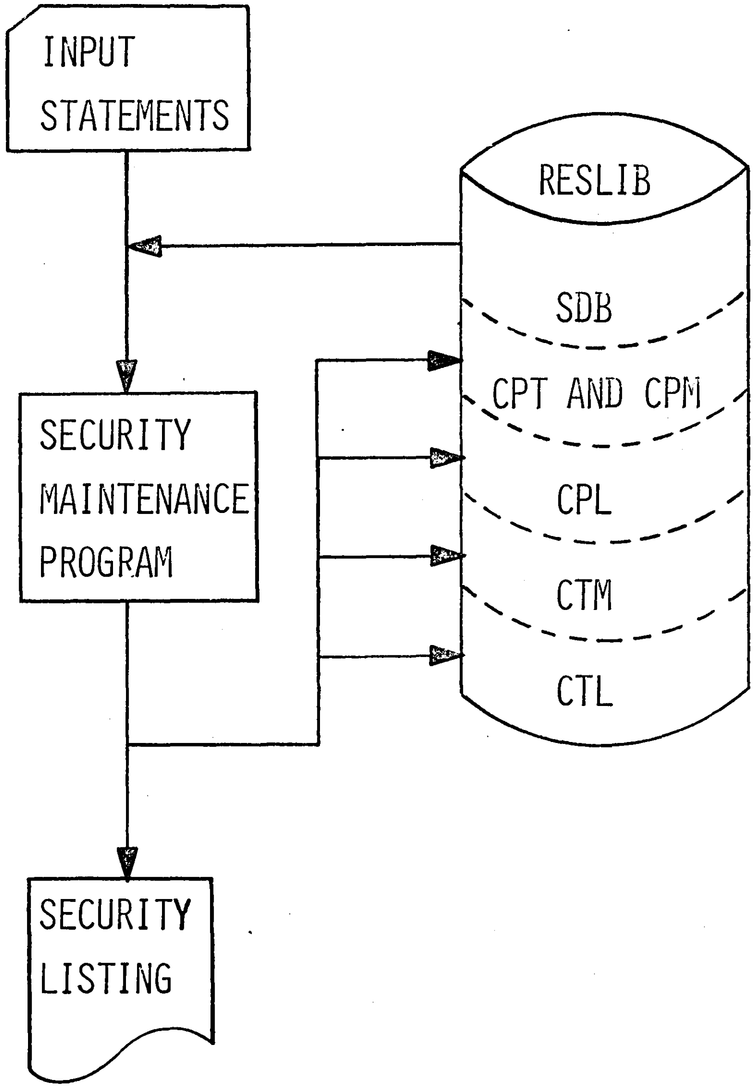
			TOTAL	TOTAL	AVG.	HOURLY	DISTRIBUTION	
LINE	TERM	R/S	MESSAGES	CHARACTERS	SIZE	00-07	07-08	08-09
001	A0							
T15CASIA		S	12	600	50	1	3	2
		R	14	840	60	1	2	2
002	A0							
T15CAS2A		S	4	320	80	1	0	0
		R	4	80	20	0	0	0
T15CAS2B		S	6	180	30	1	0	1
		R	5	200	40	1	2	1
TRM		S	10	500	50	2	0	2
TOTALS		R	9	280	31	1	2	2
SYSTEM		S	22	1100	50	3	3	4
TOTALS		R	23	1120	48	2	4	4

```
     MESSAGES -- QUEUED BUT NOT SENT              DATE 05/31/67

TRANSACTION        TOTAL
   CODE            MESSAGES

T19QCA2A                  9
T19QCA2B                 19
```

TERMINAL COMMANDS

```
/CHECKPOINT

/CHECKPOINT FREEZE

/CHECKPOINT DUMPQ

/CHECKPOINT PURGE

/DBDUMP

/DBDUMP STOP

/NRESTART

/ERESTART

/DBRECOVERY
```

SECURITY MAINTENANCE FLOW



SECURITY MAINTENANCE

ADD, DELETE, OR CHANGE PASSWORDS FOR THE FOLLOWING RESOURCES:

- TRANSACTION CODES
- TERMINAL COMMAND VERBS
- PROGRAMS
- DATA BASES
- LOGICAL TERMINALS
- PHYSICAL TERMINALS

SPECIFY TERMINAL SECURITY FOR:

- TRANSACTION CODES
- COMMAND VERBS

IMPLEMENTATION AND INTERNALS

Outline

Bibliography

Information Management System/360
  Application Description Manual H20-0524
  System Operations Manual
  Program Description Manual
  Machine Operations Manual

IMPLEMENTATION AND INTERNALS

Objectives: Upon successful completion of this topic, the student is able to:

1. Describe the initialization process for IMS/360

2. Describe the flow of control between the batch and message processing regions and the IMS/360 control modules

3. Write the necessary control cards and generate an IMS/360 system

4. Write the necessary input cards to the security maintenance program to establish security requirements for the IMS/360 system.

5. Describe the terminal commands for IMS/360.

6. Explain the use of the checkpoint and restart facilities during normal and abnormal conditions.

7. List the various log entries and statistical and accounting reports available from the system.

8. Explain the relationship of the various communication control blocks in a switched and non-switched environment.

9. Estimate storage requirements for an IMS/360 system

INSTRUCTORS' NOTE:

This section describes how the system is initialized and shows the flow of control once the system is loaded.

A. IMS/360 Initialization and System Flow

1. Assume that OS/360 is to be loaded. An IMS/360 cold start is to take place.

   a. The operator depresses the IPL key.

(V-1)

b. The OS/360 nucleus is loaded, system queue space is allocated, and messages are sent to the OS/360 console asking for system parameters, time, automatic initialization of readers, writers, and initiators. The operator responds in the normal manner when starting OS/360.

(V-2)

c. The operator keys in start WRITER and READER commands.

d. JCL to start IMS/360 is placed in the card reader.

e. The job is read in and placed in the OS/360 System Job Queue.

(V-3)

f. The operator starts an INITIATOR.

2. The INITIATOR reads the job from the job queue, loads IMS/360, and gives it control. Initiator will usually be overlayed by IMS/360.

3. IMS calls its initialization routines and gives them control.

(V-4)

a. The initialization routine loads the Data Language/I ISAM, OSAM, BTAM, and QSAM into the IMS region if they are not to reside in Link Pack. The stand-alone modules and access methods are loaded.

b. Buffer pools are created for message queues, data bases, PSB and DBD control blocks

c. The master terminal line group is opened.

(V-5)

d. A message is sent to the master terminal: "IMS READY 068092/115930" (Julian date and time).

e. The master terminal is polled as IMS/360 is expecting a restart command.

4. The master terminal operator enters an /NRESTART command with the appropriate parameters. Since this is a cold start, CHKPT=0 would be one parameter.

   a. "NRESTART IN PROGRESS" will be typed on the master terminal by IMS/360.

   b. The message queue data sets and the log tape are opened.

   c. The security tables are loaded and initialized as part of the /NRESTART command if the appropriate parameters are entered from the master terminal.

   d. IMS sends a message to the master terminal: "*COLD START COMPLETED, ENTER START COMMANDS".

5. The operator will start other lines on the system through the /START command and IMS responds with *START COMMAND COMPLETED". A message is provided on each terminal that is started: "TERMINAL STARTED". The other lines are then polled and input will be accepted or output transmitted.

6. In order for message processing programs to be executed, it is necessary to start a message region.

   a. A message region may be started by entering a deck of JCL cards through the card reader or by a /START MSGREG command from the master console.

   b. The /START MSGREG command causes the IMS control program to start one message region. If the command is entered a second time, another region will be started.

   c. The /START MSGREG command causes a START READER command to be simulated and the JCL for a job is read from disk and placed in the system job queue. Part of the /START MSGREG command's responsibility is to start an initiator which will later be used to initiate the message region as an OS/360 job.

(V-6)

7. When a message region is started, a region controller is loaded into the region and given control (or a copy of the reentrant region controller already in link pack is given control). The region controller is approximately 500 bytes.

a. The region controller is resident for the life of the message region.

b. The region controller, upon gaining initial control in the message region, executes one of the interregion SVCs and informs IMS that the region is available for message processing.

(V-7)

8. Immediately after IMS/360 gains control and the message region's region controller is placed in the wait state, the IMS dispatcher gives control to the IMS scheduler if there is a message to be processed.

a. The scheduler determines the program to be loaded and executes another interregion SVC to post the region controller of the message region control.

b. One of the parameters which is passed to the region controller is the name of the program to be loaded.

(V-8)

9. Prior to loading the application program, the region controller loads a module called the Data Language/I block loader (DFSIDLLO) which determines if sufficient control blocks are in core in the IMS region to execute the program. These are blocks such as the PSBs and DBDs.

a. If the necessary blocks are not available in core, the block loader reads in the necessary information and builds the control blocks.

b. After the control blocks are built, the block loader requests, through the interregion SVC, that the blocks be moved into the IMS0 region.

c. IMS (Data Language/I block move module) moves the control blocks and issues another interregion SVC to return control to the block loader. The block loader then returns to the region controller.

d. The region controller ATTACHes the appropriate program and it is loaded into core and given control.

e. The application program calls on Data Language/I with a GET UNIQUE for the first segment of the input message.

10. Once IMS/360 is loaded and initialized, it is ready to start accepting messages from the master terminal.

    a. The IMS/360 dispatcher of the Type 0 region gives control to the telecommunications facility for communications with the master terminal.

    b. The master terminal operator restarts the system.

    c. The restart facility restarts the system from a log tape if it is a "Warm Start". No input log tape is required if this is a "Cold Start".

    d. The restart facility returns control to the telecommunications facility, which allows the master terminal operator to start the other communication lines.

    e. The telecommunications facility returns control to the dispatcher.

    f. Telecommunications receives the incoming message and invokes the common service facility.

    g. The common service facility logs the message, queues it on disk, and returns control to the dispatcher through the telecommunications facility.

    h. When a complete message is received, telecommunications notifies the message scheduling facility of input available for processing.

    i. The region controller of the message region is given control by an SVC.

    j. The application program is loaded by the region controller and given control.

    k. The message processing program accesses messages and data bases through the Data Language/I facility. (Control comes through the IMS/360 dispatcher.)

l. When the message or data base segment is given to the application program, another SVC is issued to give control back to the application program.

m. The same sequence of events (as obtaining a message) is repeated when sending output.

n. When the application program terminates, the region controller gets control and issues an SVC to give the IMS/360 dispatcher control.

o. The dispatcher passes control to the message scheduling facility, which notifies the telecommunications facility of pending output.

p. Subsequently, the telecommunications facility gets control and invokes the common service facility to obtain a message from the queue. It is then transmitted by the telecommunications facility.

q. Based upon either message volume or notification from the master terminal, a checkpoint of the system occurs. This is performed by the checkpoint facility through the common service facility.

11. Once IMS/360 is initiated, a Type 2 (batch) processing program can be initiated.

    a. The Type 2 (batch) processing program is controlled through a region controller in the same manner as a Type 1 region.

    b. The batch program has access to the input message queues, data bases, and output message queues.

    c. The transaction type to be processed from the input message queue by the batch processing program is specified in the JCL for the batch program (on the EXEC card).

12. It is recommended that a Type 2 batch processing program not update a data base used for online processing. Data base backout does not handle data base backout for Type 2 batch regions.

INSTRUCTORS' NOTE:

The terms "tables" and "blocks" are sometimes used interchangebly in this section. The specific structure of various blocks and tables can be found in the IMS/360 System Manual.

B. Communications Control

Communications Control provides the user an interface between his remote terminal and IMS/360. Within the provisions of security control, transactions may be entered from a remote terminal, resulting in the scheduling of message programs that may update or inquire into one or more bata bases.

(V-10)

1. Inputting a message from a terminal

   a. The user keys in the message.

   b. The input message is received by communications control and is translated from terminal code to EBCDIC.

   c. Backspace and control character elimination is provided.

   d. A check is made to determine if the segment ends with the characters **. If so, the segment is cancelled and ignored.

   e. The destination of the message is determined from the transaction code.

   f. Security requirements are checked

   g. Message is written on the log and queued on a random access device.

   h. The application scheduler is notified that a complete message has been received.

(V-11)

2. Processing the message

a. When the user program which processes the entered transaction is available and scheduled, it requests a message from the message queue.

b. The user program obtains the message a segment at a time and may generate one or more messages as a result.

(V-12)

3. Outputting the response

a. Output messages are written on the log and queued on a random access device by logical terminal destination or transaction code. Output from one program may be input for another program.

b. If the output message has been queued upon a logical terminal, the message is dequeued when the output physical terminal and its associated line are available.

c. The message is translated to terminal code and edited relative to the insertion of control characters.

d. The message is sent to the specified terminal.

(V-13)

4. Communication Blocks

a. Communication Line Block (CLB) - this is the basic control block for communications.

1) Each physical line is represented by a CLB and it is used for control of that line.

2) The basic pointer in this block is to a Communication Terminal Block (CTB).

b. Communication Terminal Block (CTB) - each physical terminal is represented by a CTB. CTBs are grouped according to lines and are generated in line number order.

1) The number of the associated line is present in the CTB. A pointer to the associated CLB can be calculated, as the CLBs are in line number order.

2) An index to a communications terminal table (CTT) exists in the CTB, providing the ability to calculate the associated CTT address.

c. Communications Terminal Table - there is an entry in the CTT for each type of terminal. The CTT provides information about the hardware features of a terminal. A separate CTT exists for each type of terminal and for each set of unique hardware features within the terminal type.

d. Communications Name Table (CNT) - each logical terminal is represented by a communications name table.

1) The CNT functions as an output queue block and points to the physical terminal (CTB) to which the output will be directed. Although any number of logical terminals may direct their output to one physical terminal, one logical terminal can direct its output to only one physical terminal at a given time. The association of a logical terminal with a given physical terminal can be changed by means of the command language (/ASSIGN).

2) Many logical terminals may point to one physical terminal.

3) A logical terminal may point to only one physical terminal.

4) The destination of messages can be changed by changing the relationship of physical and logical terminals (/ASSIGN).

(V-14)

e. Multiple CTBs may be related to a CLB.

1) A pointer always exists to the first CTB related to the line.

2) Upon the receipt of the first segment of a message from a terminal on the line, a pointer is dynamically established from the CLB to the inputting CTB.

3) Multiple logical terminals (CNTs) may be related to a CTB.

4) Multiple CTBs on the same line point to a single entry in the CTT.

(V-15)

f. A single CNT (logical terminal) may be associated with one physical terminal (CTB) for input and another for output.

This can be accomplished through use of the /ASSIGN command.

INSTRUCTORS' NOTE:

This section logically breaks away from the communications blocks. The blocks described here are used for controlling and routing the message once it is in the system.

(V-16)

g. Scheduler Message Block (SMB) - each transaction code known to the system is represented by an SMB.

1) The SMB contains a relative offset pointer to an entry in the Program Specification Block Directory, thus tieing this transaction code to a specific message program for processing.

2) There are three priorities - normal, limit, and current, contained in the SMB, as well as the count of unprocessed messages of this type.

h. Scheduler Priority Table (SPT) - this table (actually a number of separate blocks) is used for queuing SMBs by priority.

1) Priorities range from 0 to 15. Priority 15 is not available as it is reserved for the system. Priority 0 is a null priority and will not cause a message processing program to be scheduled.

2) A batch program can request the 0 priority messages.

i. Message Request Queue (MRQ) - the MRQ is interrogated by the dispatcher to determine if a message is ready for processing. If the Partition

Request Queue (PRQ) is also posted, then a partition is available and the application scheduler gains control.

1) The MRQ indicates the presence of messages within the system through an event control block.

2) When a message is enqueued on an SMB, SMB is enqueued on the SPT representing the appropriate priority. The specific SPT is then enqueued on the MRQ in priority sequence, and the MRQ is posted.

Schedulable priority levels are then obtained in order by dequeuing them off the MRQ.

(V-17)

j. The MRQ contains pointers to the highest and lowest SPTs with SMBs ready for scheduling.

SMBs are chained off the SPT entries by priority and order of arrival within priority. SMBs are FIFO enqueued at the appropriate SPT priority level.

(V-18)

k. The SMB has messages queued on it. The queue consists of Queue Control Records (QCRs). A QCR may contain a pointer to an incore buffer, a pointer to a relative disk block containing the message, or it may contain the message.

Messages of one segment are contained in the QCR. If it is necessary to queue on disk, a relative block pointer to the message is established.

INSTRUCTORS' NOTE:

The fields of all communication control blocks and their function can be found in the Blocks and Tables chapter of the System Manual.

C. Switched Communications Network

1. Before discussing the switched network it is important to have a good understanding of the nonswitched environment.

(V-19)

   a. The relationship between the physical terminal and the logical terminal is a fairly stable one and is defined at system definition time. This can be varied with the /ASSIGN command.

   b. Typically, there is a one-to-one relationship between the physical terminal and the logical terminal. There may, however, be a number of logical terminals associated with a physical terminal.

(V-20)

2. In the switched network environment, the relationship between a logical terminal and a physical terminal is not established until the remote user dials the computer and issues the /IAM command.

   The relationship between a terminal user, a physical terminal, a commucication network, and IMS/360 logical terminals at system definition time appears as shown. A physical terminal may have the possibility of relating to a number of logical terminals.

   Once the user dials the System/360 computer and issues the /IAM command to sign himself on to IMS/360, a stable relationship between the physical terminal and one or more logical terminals is established.

(V-21)

3. In the switched communications network environment, the IMS/360 user employs system definition to define one or more communications lines.

   a. Associated with each line there must be one logical terminal designated as the inquiry logical terminal for the dialable communication line.

   b. In addition to an inquiry logical terminal for each dialable communication line, pools of logical terminals may be defined at system definition time.

   c. One or more logical terminals from the pools of logical terminals are associated with a particular line when a remote terminal user dials the IMS/360 system.

(V-22)

d. Within any logical terminal pool for a switched communications network, the IMS/360 user can define logical terminal subpools.

   1) A logical terminal subpool is composed of one or more logical terminals within a given logical terminal pool.

   2) A particular logical terminal may exist in only one pool and subpool.

   3) A remote user may call in and sign on for a single logical terminal or all logical terminals within a logical terminal subpool.

(V-23)

e. After dialing the computer, the relationship is established by the /IAM command.

   1) The LTERM parameter may specify the inquiry logical terminal.

   2) The LTERM may specify a logical terminal from the pools of logical terminals.

   3) If LTERM and PTERM parameters are specified, all logical terminals within a subpool are associated with the physical terminal.

(V-24)

f. The logical terminal subpool concept allows for efficient use of communication facilities. If the PTERM parameter was specified, all of the output queued on each logical terminal in the subpool for which the /IAM command was issued is sent to the physical terminal.

g. A subpool might be defined to contain the logical terminals for all of the users of a single physical terminal.

   While a user is signed onto a logical terminal within the subpool, the subpool is unavailable to users signing on from other physical terminals.

h. The relationship of the control block in a switched line group can be seen here.

4. A /START LINE command, when issued against a switched network answering line, results in the specified line adapter being enabled.

(V-25)

a. A physical connection will be established when a remote terminal operator dials the telephone number of the answering line adapter.

b. After a phyiscal connection is established, the communications controller monitors all terminal input to assure that a logical connection is completed before any transactions or commands other than /IAM are accepted.

c. If a logical connection is not established within five input messages, the physical connection is terminated by disabling the answering line adapter, and reenabling it to answer the next call.

D. System Generation

1. IMS/360 is distributed on a nonlabeled, nine or seven track 800 BPI tape. An optional tape is available containing the source modules.

(V-26)

2. The first step is to move the data sets to disk. This is accomplished with IEHMOVE. The data sets are unloaded copies of partitioned data sets.

INSTRUCTORS' NOTE:

The JCL for accomplishing move is in Chapter 2, Volume 1, of the Systems Operation Manual.

(V-27)

3. System generation is accomplished by using the IMS/360 macro-instruction contained within IMS.GENLIB.

The IMS/360 generation is similar to an OS/360 generation. It is accomplished in two stages.

a. Stage 1 causes a series of jobs to be produced

b. Stage 2 consists of processing the individual jobs produced from Stage 1.

4. A system may be defined for online and batch processing or batch-only processing.

(V-28)

5. System generation macros are used in defining the IMS/360 system. A complete list of the macros is shown here. Each will be treated in turn.

Note the requirements for batch-only system.

a. Three groups of macro-instructions are required for the description of user resources.

1) Group 1 describes the OS/360-IMS/360 operating environment and resources such as SVCs, buffers, and data sets.

2) Group 2 describes application programs and their related resources.

3) Group 3 describes communication line groups, communication lines, and associated physical and logical terminals.

(V-29)

b. IMSCTRL is used to describe the basic IMS/360 control program options and the Operating System/360, with which IMS/360 will operate.

1) SYSTEM - specifies which system environment is to be used - MVT-MFT-PCP.

a) ALL - Generate batch and teleprocessing system.

b) BATCH - Generate batch-only system.

2) MAXIO - Specifies the maximum number of terminal I/O requests, message queue requests, and Data Language/I data base requests which may be in progress in the IMS/360 control program region at any one time.

3) MAXREGN - The maximum number of regions or partitions to be supported at any one time

4) COMMSVC - specifies the numbers of the Type 1 SVC which IMS/360 uses for interreqion communication

5) OCENDA - specifies the load module member name to be given the OSAM channel end appendaqe.

6) OSAMSVC - specifies the user SVC number to be given the OSAM Type 2 SVC.

7) MSGBUFF - specifies number of incore message buffers for multiple line messages.

8) CKPT - a checkpoint will be written each time the specified entries are made to the log.

(V-30)

c. MSGQUEUE - defines the input and output single line message and multiple line message data sets desired by the user.

1) QCRIN - defines the single line input message data set.

2) QCROUT - defines the single line output message data set.

3) MSGIN - defines input multiple line message data set.

4) MSGOUT - defines the output multiple line data set.

(V-31)

d. A series of definitions for the various libraries must be made as to unit and volume. A default option for the name is provided for each. These macros cause the various data sets to be moved to the appropriate volume. The XXXLIB macros are all of the same format with the exception of MACLIB which has a COPY. The COPY allows the entire macro library to be copied or only the macros required for PSB or DBD to be copied.

e. IMSGEN - Specifies the data sets, volumes, and I/O devices required for the definition process.

1) UT1SDS - names a utility to be used during Stage 2 by the assembler and linkage editor.

2) ASMPRT - specifies whether assembly listings will be produced for the module assembled during system definition.

3) LEPRT - the linkage editor print options LIST, MAP, and XREF. XREF includes MAP.

(V-32)

(V-33)

f. APPLCTN - Describes the program resource requirements for application programs which will run under control of the IMS/360 Type 0 region.

1) PSB - specifies the logical name of the program specification block as generated using the IMS/360 PSB generation utility.

2) PGMTYPE - specifies message processing region or batch.

g. DATABASE - Defines all data bases to be used by the preceding APPLCTN macro.

1) DBD - logical name of the data base as generated by the DBD generation utility.

2) INTENT - specifies whether the program is read-only, update, or sole use to the exclusion of all other applications which may use the same data base.

3) LOG - the logging of all segments, added, deleted, or replaced in the data base will allow data base "backout".

h. TRANSACT - Specifies the transaction codes which will cause the program named in the APPLCTN macro to be scheduled.

1) CODE - specifies the transaction code.

2) PRTY - specifies the priorities at which this transaction code contends for IMS/360 resources with other transaction codes.

a) Normal and limit priorities may range from 0 through 14.

b) The limit count may range from 1 through 65535.

c) Default values for normal, limit, and limit count are 1,1,65535.

3) MSGTYPE - Specifies when a message is considered complete. That is, whether it is single line or multiple line.

a) SNGLSEG - Single line input

b) MULTSEG - Multiple line input

c) NONRESPONSE - Accept further input without waiting to respond to the message previously entered.

d) RESPONSE - upon completion of an input message, no further input from the line and terminal is accepted until the response to the input is sent. (One in - one out.)

4) PROCLIM - Specifies the maximum number of seconds allowed for processing each message and the maximum number of messages to be processed upon each loading of the program. Default values are 65535, and 65535.

5) INQUIRY - If the operand is NO, data base recovery reprocesses all messages entered against this transaction code and no activity is allowed against this transaction during DBDump.

If the operand is YES, data base recovery does not reprocess messages against the transaction code and input is allowed against this transaction code.

(V-34)

(V-35)

i. LINEGRP - Defines the beginning of a set of physical terminals, communication lines, and logical terminal definitions.

1) DDNAME - specifies the ddname which is used to allocate the communication line devices described in the following LINE and TERMINAL macros.

2) FEAT - specifies features concerning the line group.

3) UNITYPE - specifies the unit number.

(V-36)

j. LINE - Defines the beginning of a set of TERMINAL and NAME macro-instructions which describe the physical and logical terminals on a single communications line.

1) FEAT - Describes the features on the terminals which are attached to this line.

2) ADDR - Address of the communication line

3) ZONE - specifies the WATS area zone to be associated with this line.

k. TERMINAL - Describes a physical terminal which must be an input device. It may in addition be an output device.

ADDR - specifies the physical terminal address

l. NAME - defines a logical terminal name to be associated with the physical terminal described by a preceding TERMINAL or SUBPOOL macro. At least one must follow TERMINAL or SUBPOOL.

m. MASTTERM - Identifies the master terminal. Only one of these is allowed for each IMS/360 generation.

(V-37)

n. POOL - Describes a pool of logical terminals which are to be associated with a set of switched communication lines.

1) ZONE - specifies the WATS area zone associated with these logical terminals.

2) FEAT - specifies the POOL of logical terminals to be associated with those physical lines

defined by the LINE macro with the equivalent FEAT operands.

o. SUBPOOL - delimits a set of logical terminals to be associated with a given physical terminal. At least one for each POOL macro.

1) TELNO - specifies the telephone number for AUTOCALL operations.

2) lterm name - a logical terminal name of one to eight alphameric characters must be unique.

(V-38)

(V-39)

6. The output from Stage 1 consists of a series of job steps. These job steps can be logically divided into six groups.

E. Security Maintenance

(V-40)

The security maintenance program is a utility program which is run after IMS/360 generation if security is desired in the system.

1. The security maintenance program accepts control statements and data statements and produces two matrices which are used for terminal and password security.

(V-41)

a. Password security may be specified for transaction codes, terminal command verbs, programs, data bases, logical terminals, and physical terminals.

b. Terminal security may be specified for transaction codes and commands.

2. The security can become effective on the next restart. The master terminal operator may specify this with the restart command (TERMINAL and/or PASSWORD).

3. The security maintenance program is not executable unless an IMS/360 system definition has been performed. The Security Maintenance Program (SMP) requires the IMS/360 System Definition Block (SDB) as input.

(V-42)

4. Input cards to the SMP are control statements or data statements.

   a. Control statements contain a right parenthesis,), and left parenthesis (, in positions 1 and 2; position 3 is blank followed by the control word.

   b. Data statements contain a blank in the first position.

   c. The valid combinations of control and data statements are shown here.

      1) A password may begin with any alphanumeric character.

      2) Passwords are one to eight characters in length.

   d. Only the first three characters of the operation code of control or data statements are necessary to identify the statements.

   e. Physical terminal numbers may be found in the terminal map printed at the end of Stage 1 of IMS/360 system definition.

5. Security maintenance is discussed in two parts. The first covers password security and the second covers terminal security.

(V-43)

   a. Password security may be expressed as a password profile or a resource profile. The results are the same.

      1) The first part of this example shows a password profile. The password SAMSMITH gives access to the resources which follow.

      2) The second part shows a resource profile. These resources require these passwords.

      3) The results of these two series of statements would be identical. The two methods simply offer two ways of looking at the problem.

(V-44)

b. This shows a different way of looking at the profiles. In the vertical here, we see a password profile.

If turned horizontially (turn), we see a resource profile.

(V-45)

c. The actual implementation can be seen here. Each resource is assigned a row of matrix and each column is assigned a password.

Here we see that transaction code PAYREC requires the password SAMSMITH.

6. Terminal security is provided for the command language and transaction codes. Terminals may be limited as to which commands and transaction codes they may enter.

(V-46)

a. Terminal security may be expressed as a transaction and command profile or as a terminal profile.

1) The first part of the example shows a transaction and command profile. The second part shows a terminal profile.

2) Results of the two are identical.

(V-47)

b. The vertical here shows a terminal profile. Terminal DEPT 40 can enter transaction codes PAYROLL and PERS.

c. The horizontal (turn), shows a transaction and command profile (no commands are present). INVENTORY may be entered from logical terminals S274001, S274002, and S274003.

(V-48)

7. This shows the implementation of the terminal matrix. Each transaction code and command which is secured is assigned a row in the matrix. Each logical terminal included in security is assigned a column.

In the example shown, TRAN2 could be entered from logical terminal CNTA and CNTD.

(V-49)

8. The security maintenance run is a three-step job.

   a. Step 1 edits the control and data cards for the security maintenance program. The cards are checked for validity against the system being maintained.

   (V-50)

   b. Step 2 is an assembly.

   c. Step 3 is a linkage edit which places four sequential members in IMS.RESLIB.

      1) Communication Password Table and Matrix

      2) Communication Terminal Matrix

      3) Communication Password List

      4) Communication Terminal List

   d. Only those members affected are changed when the security maintenance program is run. For example, the communication terminal and the communication terminal list can be generated or altered without affecting the communication password table and matrix and the communication password list.

   e. A listing is provided of the created maintenance tables.

9. JCL required to run the security maintenance program can be found in the IMS/360 Systems Operation Manual.

(V-51)

F. Cc.ımand Language Facilities

The command language is divided into two groups for the purpose of presentation -- master terminal commands and remote terminal commands. A different division may be selected by each installation at IMS/360 system definition time.

1. Any command message entered results in a completion message going to the originating terminal and affected terminals; error messages go only to the originating terminal.

2. Certain command type messages which interrogate, alter, and control the overall system should be restricted to entryzfrom the master terminal.

3. Passwords may be required for commands. This is specified when the security maintenance program is run.

(V-52)

4. The format of the command is simple and relatively free form.

   a. The first character is a slash (/).

   b. The slash is followed by the command verb.

   c. If required, the password follows the verb and is enclosed in parentheses. The command and the password left parenthesis may be separated by one or more blanks.

   d. Following the password or command may be a list of key words and parameters.

   e. Although much freedom is allowed when inputting the command, the following rules apply:

      1) The password is enclosed in parentheses (bypass and restore characters may be substituted on the 1050).

      2) Delimiters are space, dash, or equal sign.

      3) A series of parameters is separated with commas.

      4) A period is not required, but if present, it designates the end of the command.

      5) Any characters following the period are treated as a comment.

(V-53)

      6) Key words are used with commands. A partial list is shown here. A complete list is

available in the IMS/360 Machine Operations Manual.

a) LINE - a communication line; one to three character numeric line numbers.

b) PTERM - A physical terminal; one or two character physical terminal address

c) LTERM - A logical terminal; one to eight character alphameric logical terminal address

d) TRAN or TRANS - a transaction code; one to eight alphameric characters.

e) PROG - PROGRAM or PGM - a program; one to eight alphameric character program name

f) DATABASE - a data base; one to eight alphameric character data base names.

g) PASSWORD or PSWD - a password; passwords as designated at system definition time.

h) ALL - may be used with many key words. Acceptable uses are explained with the individual commands

i) CPRI - current priority; one or two character numeric priorities between 0 and 14, inclusive

j) LPRI - limit priority; one or two character numeric priorities from 0 to 14, inclusive.

k) LMCT - limit count of a transaction code; values range from 0 to 32,000.

l) TERMINAL or TERM or TER - a terminal when deleting terminal security.

m) REGION or REG or MSGREG - refers to a message region.

n) Certain key words are limited to use with the checkpoint/restart commands and the display command. They are discussed with these specific commands.

o) Certain words are considered null if used in a command. These are FOR, SECURITY, TO, ON, MODE, and AFTER.

5. Master Terminal Commands

Certain commands are restricted to the master terminal by IMSGEN. Others may be restricted through the security maintenance program.

(V-54)

a. /CHANGE - changes one password to another. In the example shown, ABCD must be present in the password table and the addition of EFGH must not create a duplicate entry in the password table (discussed under security).

b. /ASSIGN - can be used for the reassignment of a logical terminal relative to input and/or output.

A logical terminal that is to be assigned must be stopped unless the logical terminal is the master logical terminal.

This command is also used to change current priority, normal priority, limit priority, or limit count for an SMB.

(V-55)

c. /DELETE - used to eliminate password security for a transaction code, physical terminal, logical terminal, program, or data base. It can also be used to delete terminal security for one or more SMBs.

(V-56)

d. /DISPLAY - displays the status of the system relative to requested information.

1) STATUS - causes the displaying of transaction codes and their status relative to inputting, scheduling, locked or unlocked, or locked specifically for DBDUMP.

The status of data bases, programs, lines, physical terminals, and logical terminals is also displayed.

2) ACTIVE - displays the active program in Type 1 and Type 2 regions and the transaction code for which they were loaded.

3) QUEUE - displays the message queue according to priority, transaction type, and message count. If used with PRIORITY transaction code names for that priority will be displayed.

4) TRAN - displays data for the transaction code(s) specified.

5) PGM - displays data for the program(s) specified.

6) DATABASE - displays data for the data base(s) specified.

7) LINE - displays data for the specified line. For a further breakdown PTERM may be used.

8) LTERM - displays data concerning the logical terminal name(s).

9) ASSIGNMENT LTERM - displays which input and output communication line and physical terminal address are assigned to each LTERM. LINE and PTERM may be used instead of LTERM.

10) MASTER - displays the logical terminal name, physical terminal address, and line number assigned as the master terminal.

(V-57)

e. /PSTOP, /PURGE, and /STOP are used to temporarily stop various system resources such as lines, terminals, transaction codes, programs, and data bases.

(V-58)

The action taken with each command and its key word are shown in chart form. The numbers under each key word refer to the action taken.

f. /START is used to start various system resources initially or after a /PSTOP, /PURGE, or /STOP.

6. Remote Terminal Commands - These are used by the remote terminal operator to control his own resources. These

include those allowed by system definition and not restricted by running the security maintenance proqram.

(V-59)

a. /BROADCAST - used to transmit a message to one or more terminals.

   1) This is an exception to the rule that all command messaqes are one line in lenqth.

   2) The command is entered as one line and the message is entered on a separate line.

(V-60)

b. /TEST - Places the user's terminal in a test mode such that any input messaqes that are entered into the user's terminal will be turned around and transmitted back to the user's terminal. Messaoes from outside sources will not be transmitted to a terminal that is in test mode.

c. /EXCLUSIVE - This command is used to place the user's own terminal into exclusive or inquiry mode.

   In this mode, no output will be sent to the terminal which is not in response to an inquiry from that terminal.

d. /END - this command is used for endinq the /EXCLUSIVE or /TEST mode.

e. /LOG - causes the entire line to be written on the log tape.

f. /CANCEL -zcauses the cancellation of an entire input message regardless of the number of lines.

   This is the only command that can be entered as other than the first line of a message.

(V-61)

g. /LOCK and /UNLOCK - used to control various facilities of IMS. /LOCK is functionally similar to /PSTOP. /LOCK is explained here.

   1) PTERM - applies to the physical terminal into which the command is entered. This causes the

physical terminal to be locked. No command except /UNLOCK will be accepted.

2) LTERM - a logical terminal or a series of logical terminals which are associated with the physical terminal from which the command is entered. Parameters can be one or more names or ALL; parameters apply to user's own, only; cannot use ALL with /UNLOCK.

3) TRAN - do not schedule this transaction code.

4) PROGRAM - do not schedule this program.

5) DATABASE - do not schedule any program that uses this data base.

(V-62)

h. /IAM - this is the first acceptable command from a dial-up terminal.

1) Logical terminal P1 is associated with the physical terminal.

2) PTERM - LTERM - causes the attachment of all logical terminal in which P1 exists.

i. /RDISPLAY - displays the logical terminal name, the physical terminal address, and the line number assigned as the master terminal. This is useful in determining the name of the master terminal to be used when broadcasting a message.

j. /SET - sets the destination of all messages entered into this terminal to another terminal or to an SMB.
This mode may be reset by the /RESET, /START LINE for the terminal or by the /IAM command.

k. /RESET - eliminates the existing destination mode.

G. Checkpoint, Restart, Data Base Dump, and Recovery

1. The checkpoint facilities of the IMS/360 control program provide the means for periodically recording control information and status to enable IMS/360 restart after failure.

2. The checkpoint facility provides for orderly shutdown.

(V-63)

3. There are four checkpoint commands, two data base dump commands, two restart commands, and a data base recovery command.

(V-64)

   a. The checkpoint command, /CHECKPOINT with no operands, causes a simple checkpoint.

      1) It may be invoked from the master terminal.

      2) It may be taken automatically based on the number of log entries.

(V-65)

   b. The simple checkpoint logs the status of all dynamically changing IMS/360 control program blocks.

   c. Scheduling of programs into message regions is stopped while the checkpoint is taken. Other functions are not interrupted.

   d. The three remaining checkpoint commands are each used to orderly terminate the IMS/360 system.

      Each is invoked from the master terminal.

(V-64)

   e. The /CHECKPOINT FREEZE is the fastest means of orderly termination.

      1) Input communications lines are terminated as soon as any messages being entered are completely received.

      2) Output communications lines are terminated as soon as any messages being sent are completely transmitted.

      3) Message regions are terminated as soon as the current messages being processed have been completed.

4) All remaining input messages to be processed and all output messages remaining to be transmitted are retained in the message queue data sets.

5) Control blocks are logged.

6) The /NRESTART command is used to restart the system.

7) If ABDUMP is included a SYSDUMP of the IMS/360 Type 0 region will be provided.

f. The /CHECKPOINT DUMPQ operates exactly as the /CHECKPOINT FREEZE but, in addition, all input and output messages in the queues are dumped to the log tape.

The /NRESTART with message queue reconstruction is used to restart IMS/360 after a /CHECKPOINT DUMPQ.

g. The /CHECKPOINT PURGE command is the most orderly and time-consuming.

1) Input communication lines are terminated as soon as messages being entered are completely received.

2) All messages in the input queue are processed.

3) All output messages are transmitted to their specified destinations.

4) The message regions are terminated.

5) Input and output messages which could not be processed are dumped to the log tape.

6) IMS/360 control program job is terminated.

7) The /NRESTART with message queue reconstruction is used to restart.

(V-66)

8) The order of action during shutdown checkpoint is shown here.

4. The data base dump capabilities of checkpoint include the functions of creating a dumped tape image of a complete data base.

It also performs preparatory functions for the reconstruction of a data base.

(V-64)

a. The /DBDUMP command is entered from the master terminal.

1) All transaction input of an update nature against the data base is stopped.

2) Transactions already in the queue against the data base are processed.

3) A special checkpoint request is issued. This takes a checkpoint and then forces an end-of-volume on the system log.

4) A special utility is scheduled for execution in a message processing region.

5) The utility program issues GETs against the data base and ISRTs to an HSAM tape.

6) At the completion of the data base dump, the tape is unloaded.

7) Update transactions are allowed to come in from terminals.

b. The /DBDUMP STOP is used in preparatory procedures prior to data base reconstruction.

1) Transactions against the data base are not scheduled for processing.

2) The data base is reconstructed with a batch program executed from an IMS/360 Type 3 region.

3) After reconstruction, all transactions on all log tapes since the last data base dump must be reprocessed.

5. The restart facilities provide for recovery of IMS/360, its message queues, and the data bases.

a. Normal restart has two basic versions.

1) Cold start

2) Restart after normal shutdown

Checkpoint number, julian date, and time of dayz

3) FORMAT ALL

4) BLDQ

b. Emergency restart is used to restart after a system failure.

1) In order to recover after a loss of core, the /ERESTART command is issued with the checkpoint number of the last checkpoint prior to failure and the tape serial number.

a) IMS/360 control blocks are restored from the checkpoint data.

b) The log tape is processed forward from the checkpoint to the point of failure.

This process allows checkpoint/restart to determine the message processing programs which were active at the time of failure so that they may be rescheduled.

2) If message queues as well as core are lost, it is necessary to rebuild the message queues.

a) The FORMAT ALL parameter on the /ERESTART command causes the message queues to be reformatted.

b) The BLDQ causes the message queues to be rebuilt.

c) In order to restart with BLDQ or FORMAT ALL, it is necessary to back up to the last cold start or the last /CHECKPOINT with PURGE or DUMPQ.

The message queues are rebuilt by starting with the queues as they were dumped and then processing forward on the log tape.

Messages are queued and dequeued as a result of the log entries up to the point of failure. The message queues are totally rebuilt at that point.

d) Data base backout may be required as a result of a failure.

If data base logging has been requested against a data base, all changes to the data base are written on the log tape.

When recovering, if a program was executing against a data base and changes were being logged, then all changes which resulted after that program was loaded will be backed out.

e) Data base recovery can be performed if a data base has been previously dumped and update activity against the data base has been logged.

The first action is to stop all activity against the data base with /DBDUMP command with the STOP operand.

The next action is to restore the data base from the last dumped copy. This is done with a batch program in a Type 3 region.

The /DBRECOVERY command with the data base names and the serial numbers of the log tapes is used to initiate processing of the log tapes against the data base.

All transactions of an update nature are reprocessed. Messages which result from the processing may be resent to the originating terminal if this option is specified with the /DBRECOVERY command.

H. System Log

INSTRUCTORS' NOTE:

The JCL necessary to make the statistical run can be found in the Systems Operation Manual - Volume 1, Chapter 4. Examples of statistical reports can be found in Chapter 5.

(V-67)

The system recorder is a service routine designed to write data on the system log.

1. Information is written on the log tape for restart.

   a. Message enqueue blocks -- when they change

   b. Checkpoint data -- when a checkpoint is taken

   c. Record indicating data management open or close -- when an IMS/360 data set is opened or closed

   d. Changes to a data base.

2. Information can be written for restart and statistics

   a. Message received from terminal -- when a message is completely received or when the disk block is full.

   b. Message sent to a terminal or another program -- when the message is complete or when the disk block is full.

3. Information may be written for statistics only

   a. Error segments -- when a hardware error is detected while receiving or sending to a terminal.

   b. At the completion of a send record -- at completion of sending a message to a terminal.

   c. Application accounting record -- when application program terminates.

   d. IMS/360 accounting record -- when the system is started or stopped.

(V-68)

4. Records are written on the log tape using QSAM variable length blocked records.

   a. LL represents the total length.

   b. BB is a half word used by OS/360.

   c. FLAG is a one byte field identifying the log entry type.

d. RECORD is the variable length portion.

e. The logical record consists of FLAG and RECORD.

(V-69)

f. There are seven different types of records which are used for statistics and accounting.

1) The basic information in the records is similar.

2) The format for a log entry for an input message, output message, statistics, and errors is shown here.

The variable information contains such things as the transaction code and text of a message or response.

(V-70)

3) The application accounting log record contains information about an application program and is written each time a program terminates.

g. No processing is performed by the logging routine.

5. The IMS0 procedure includes DD cards for old and new log data set allocations.

a. The old log DD card name is LOGDCBR.

b. The new log DD card name is LOGDCB.

(V-71)

6. System flow of statistics utilities

a. Edit pass 1 edits the prefix of each log entry to ensure that when the log tape is sorted related input and output messages are contiguous.

b. Edit pass 2 explodes the system log entries and produces records to be used to produce statistical reports.

(V-72)

7. Statistical reports provide a means of evaluating line and terminal loading, traffic volumes, response times, and accounting (billing) information.

(V-73)

a. Messages queued but not sent -- by transaction code.

b. Messages queued but not sent by terminal. Format is the same as by transaction code.

(V-74)

c. Line and terminal report -- shows loading by time of day

d. Error reports -- for terminals. Format is the same as the line and terminal report.

(V-75)

e. Transaction report -- loading by transaction code by time of day.

(V-76)

f. Transaction response report -- measures time from complete receipt of input message until response starts back to terminal.

(V-77)

g. Application accounting report -- provides sufficient data to allow machine charges to be distributed to terminal users

   1) Counts of all requests to Data Language/I

   2) Amount of CPU task time

   3) Number of bad completion codes by program

   4) Average CPU time processing

(V-78)

h. IMS/360 accounting report -- shows amount of CPU time used by IMS/360 region. (Task time does not include wait time.)

(V-79)

8. The message select and display program selects messages based on control cards.

   a. Control cards begin in column 1 with an identifying key word.

   b. Following the key word is a series of parameters enclosed within parentheses and separated by commas.

   c. Control cards cannot be continued beyond column 71.

   d. Multiple control cards with the same key word are permitted.

   e. Within parentheses, all parameters are positional. Missing parameters must be indicated by commas.

   f. A group of names may be selected by terminating the parameter with an *. Example: INV* selects INV, INVENTORY, INVA, and INVB.

   g. The name parameter ALL may be used to select all names rather than a specific name.

9. There are five types of control cards.

   a. Transaction code

   b. Symbolic terminal name

   c. Hardware terminal address

   d. Time control card

   e. Nonprintable character control card

(V-80)

10. The sort of the log tape may be by date. If sorted by date, reports are provided by date. If sorted by period, reports are produced by period.

    SORT FIELD = (5,1,CH,A,9,4,PD,A,13,36,CH,A)

    or without date

    SORT FIELD = (5,1,CH,A,13,36,CH,A)

11. A line count parameter LINECOUNT=XX may be included in the execute card. This controls the number of lines per page, and, if not included, default is 36.

I. Message Queue Space Allocation

1. All messages (transactions) coming into the system are queued on direct access storage devices.

   a. Messages may be received from communication terminals or application programs.

   b. Messages may be destined for communication terminals or application programs.

(V-81)

   c. All transactions of the same type are queued in a serial chain based upon time of receipt.

   d. Messages destined for a particular communications logical terminal are queued serially by message class. There are four logical classes:

      1) Response Messages (Reply from Response SMB)

      2) Replies for an Exclusive terminal

      3) System Messages

      4) Other traffic (Message switches etc)

      5) Although the classes are logically separate, the messages are queued physically (on disk) in two separate strings by manipulating the queue pointers as necessary to maintain their relative priorities.

2. Messages may be single line or multiple line.

   a. Single line messages are normally maintained in a QCR (Queue Control Record).

   b. Multiple line messages are normally maintained in one or more message buffer records which have been queued to the QCR controlling the message.

   c. Small multiple line messages will be wholly contained within QCR if the text length is less than the available text capacity of the QCR.

3. From two to four data sets may be used to store messages. At least one QCR data set must be available for message queuing, and one message buffer data set must be available to allow for messages which may exceed the capacity of a QCR.

(V-82)

   a. If two data sets are used, the QCR data set is used for both input and output single line messages and for the control of both input and output multiple line messages. The Message Buffer data set is used for both input and output text which cannot be contained in a QCR.

(V-83)

   b. Three data sets may be used. A common Message Buffer data set is shown with separate QCR data sets. Possible use is when very little multiple line input is expected with a large percentage of multiple line output.

(V-84)

   c. Four data sets are normally used because this allocation provides the least contention between input and output messages.

   Message queues are OSAM data sets.

4. Message queue data sets must be preformatted before initial usage.

   a. Use of preformatted queues provides increased performance and reliability.

      1) Performance is increased by preassigning direct access storage records for any chain of messages. This reduces the input/output operations required for queue management.

      2) Reliability is increased as record X is not relied upon to write record X + 1.

   b. Messages are written sequentially from the beginning of the data set to the end.

      1) Space is reused after the entire data set is written.

2) Restart after a PURGE or DUMP queue causes the allocation of queue space to be reinitialized to the beginning of the queue data sets.

(V-85)

5. Actual space allocation is best explained by example.

a. Space required depends upon:

1) How many data sets are used (2 or 4).

2) How many messages are received from and sent to terminals.

3) The length of the messages received from or sent to terminals.

b. The calculation for the example shown assumes four data sets of a 2314.

c. The computation shown is an outside limit, as no reuse of space has been considered.

(V-86)

d. Reuse of space can significantly decrease the space allocation required for message queues.

1) Message turnover rate and number of transactions and logical terminals must be considered.

2) Message buffer records can only be reached through QCR records; therefore, allocation should be such that QCR data sets are reused before message buffer data sets.

(V-87)

3) Reuse of space adds a significant amount of disk I/O time to writing message queues (112.7%).

a) To search for each QCR string requires time.

b) QCR must be read to follow backward chain when reassigning the QCR.

c) Message buffer strings must be obtained from QCRs and linked together.

d) Message buffer record must be read to follow backward chain when reassigning the message buffer.

J. Estimating DASD Space for Data Bases

1. IMS/360 uses the OS/360 convention in space allocation for direct access storage devices.

   a. The amount of space can be specified in terms of blocks, tracks, or cylinders.

   b. If device independence is desired, space should be specified in blocks.

   c. ISAM data sets must be allocated by cylinder.

   d. Allocating space for an IMS/360 Data Base which uses ISAM and OSAM data set.

      1) OS/360 ISAM has three areas - index, prime, and overflow.

      2) IMS/360's HISAM Data Base has three areas - index, prime, and OSAM overflow.

2. In generating a data base description, the logical record size and blocking factor may be computed by the generation utility or it may be overridden on the DMAN card with LRECL and BLKFACT specifications.

   a. LRECL (logical record size) and BLKFACT (blocking factor) may be specified on the DMAN card.

   b. When LRECL is computed by the DBD generation utility, it considers the device and rounds to the next higher 1/4 track, 1/3 track, or 1/2 track.

(V-88)

3. Data Base allocation is best explained by an example.

   a. Assume a data base in which 50% of the logical records are 300 bytes or less; 70% (includes the 50%) are 400 bytes or less; 90% (includes the 70%) are 900 bytes or less in length; 100% (includes the 90%) are 1200 bytes or less in length.

b. With fixed length ISAM it is necessary to establish a fixed value for the logical record length (LRECL).

c. In the example given, an LRECL of 1200 bytes would accommodate the data base record but 90% of the records would have at least 300 bytes of slack whereas 70% of the records would have at least 800 bytes of slack.

d. If an LRECL of less than 1200 bytes is selected, it will not accommodate all of the data base records. Some dependent segments will be placed in OSAM.

e. In determining the best balance between ISAM and OSAM, a number of points must be considered.

1) Access to records completely contained in the prime area is faster than accessing data in two areas.

2) OSAM space can be used to hold overflow segments from any logical record. Slack in the prime is tied to a specific root.

3) Another consideration is that records in the prime area are blocked while records in OSAM are unblocked. With small data base records, this can make a significant difference.

4) Are the longer data base records accessed more or less often than the shorter data base records? If longer records are accessed more often, it may be best to have them in the prime area rather than in OSAM.

(V-89)

4. A data base record size is computed by the DBD generation utility using the segment sizes and frequency.

a. The calculation for a data base record length is shown by example.

b. The calculation is shown with a letter above the calculation identifying specific segments.

c. If one desired, he could specify the size of the LRECL on the DMAN card.

5. Once the LRECL is established, the blocking factor must be computed.

   a. The LRECL is always 1/2 track or less. If the computed LRECL is greater than 1/2 track, then 1/2 track LRECL will be used.

   b. The blocking will be 1/2, 1/4, or 1/3 track - whichever accommodates the most logical records. This is computed by the DBD generation utility.

(V-90)

6. If, in the example, we assume 1/2 track blocks on a 2314, six LRECLs would be placed on each track.

   a. The table shows that our LRECL of 909 falls in the 870-1158 range, which gives us three records per block.

   b. A 2314 track is shown beneath the chart. We have two blocks of three records each or six records per track.

7. Our next step is to estimate the number of roots in the data base. Let us assume 20,000 parts or 20,000 logical records.

   a. We are now ready to calculate the prime area required for the data base.

   b. Our records are blocked 6 per track and there are 19 tracks per cylinder, excluding track indexes. Thus we get 114 logical data base records per cylinder. Our requirements are approximately 176 cylinders of a 2314 pack.

8. In order to estimate the OSAM space required, one must allow space for initial loading plus space for adding new segments of information.

   a. In order to determine what is required in initial loading in our example, we must know something of the unaccounted for 10% in GENINFO, STOKSTAT, and BACKORDR segments (only 90% were less than the frequency used - the other 10% had a greater occurrence).

   b. Our calculated LRECL of 909 would have accommodated 90% of the records in our data base but we were given an LRECL of 1158.

c. With an LRECL of 1158 we would have very little loaded into OSAM, since none of our records are over 1200 bytes long.

9. This example has been given to show the approach in analyzing a data base's space requirements and should not be looked at as an absolute approach to the problem.

10. Space for the ISAM index is as required by OS/360 Data Management.

INSTRUCTORS' NOTE:

No simple answer or explanation can be given as to core requirements for IMS/360. The requirement can vary over a wide range and result from the equipment to be supported and the IMS/360 option specified. The information presented here should be used for guide lines and not as absolute requirements.

K. Estimating Core Storage Requirements for IMS/360

1. The IMS/360 control program region can vary in size as a result of the number of lines, terminals, programs, transaction types, and data bases, and the placing of certain program modules in OS/360 link pack instead of the IMS/360 control program region.

(V-91)

2. The basic requirements are presented in chart form. All modules shown other than the IMS resident nucleus may be placed in link pack.

(V-92)

In addition to these basic requirements, space must be provided for control blocks, buffers, and BTAM device modules.

3. The BTAM device modules vary in size depending on the terminal and features to be supported.

(V-93)

4. Control blocks are required for communication lines, terminals, transaction types, message processing programs, and open data bases.

5. Buffers are required for communication lines and data bases.

(V-94)

6. To estimate the total requirements for an IMS/360 system, we must first define the environment in which the system is to run. Assumptions are shown in chart form.

(V-95)

   a. The basic storage requirements consist of 60,000 bytes for the IMS/360 nucleus and 30,100 bytes of action modules and access modules. The 30,100 bytes may be in OS/360 link pack.

   b. There are two line groups - one for 2740s and one for 1050s. The requirements for the BTAM device support are shown.

   c. IMS/360 control block requirement are a function of the number of lines, terminals, transaction types, data bases, and programs. An estimate of 18,000 is used here. Some blocks included are 500 bytes for each line, 75 bytes for each terminal, and 60 bytes for each transaction type.

   d. One queue control record is required for each line (16 X 176).

   e. This example assumes that about one-third of all messages coming in are multiple line, thus space is required to store the multiple segments (5 X 880).

   f. It is assumed that output will be going to 5 lines concurrently (5 X 500).

   g. It is assumed that there are five open data bases each requiring buffer space of 3000 bytes. Since two regions may be concurrently resident, this requirement must be doubled (2 X 5 X 3000).

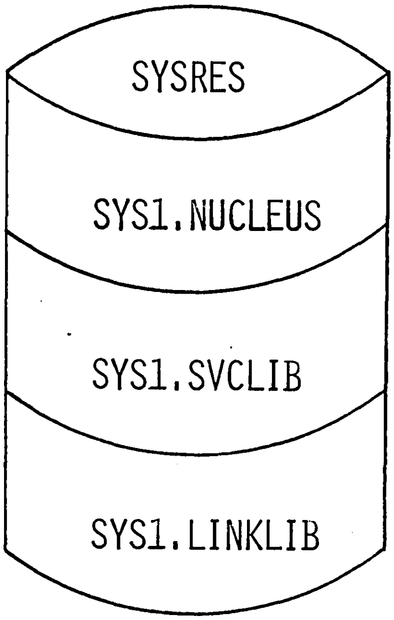
   h. There are 10 program specification blocks concurrently resident. The typical size of a PSB is from 500 to 1,000 bytes. This requirement can vary significantly depending on the complexity of the data base and the specified sensitivity of the PSB.

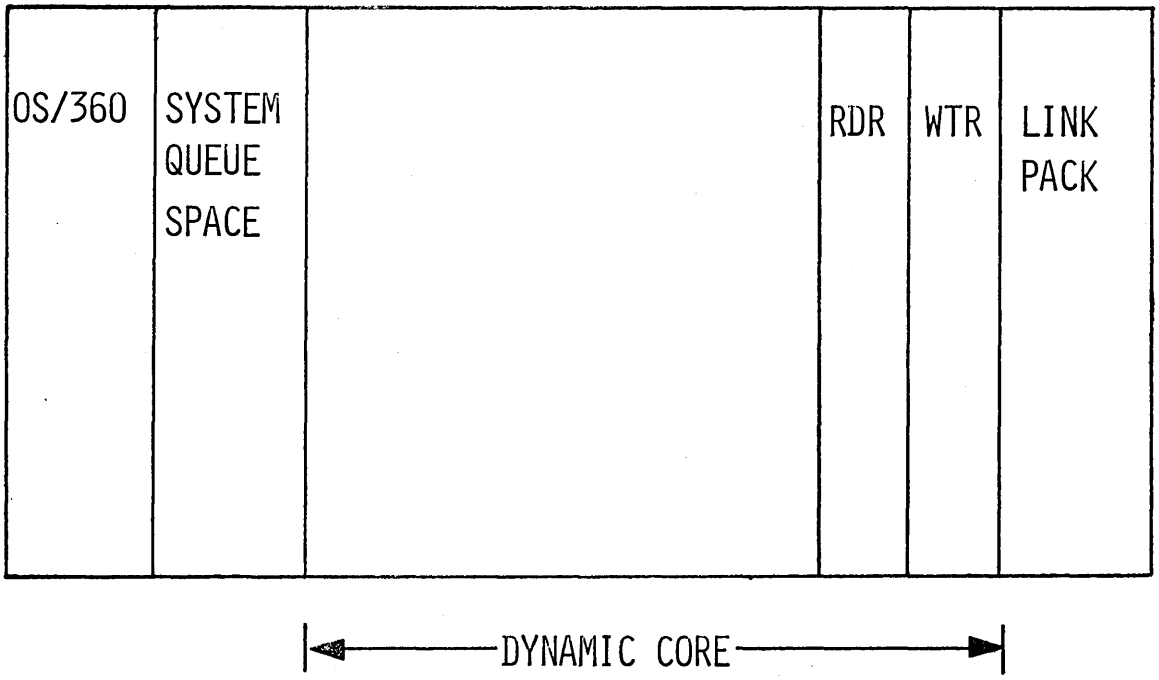
i. The data management block required for each data base varies with the degree of complexity of the data base. The assumption here is that two require 1,000 bytes while three require 500 bytes.

j. The total estimate shown is based on the assumption of specific requirements. No generalization should be based on this figure.

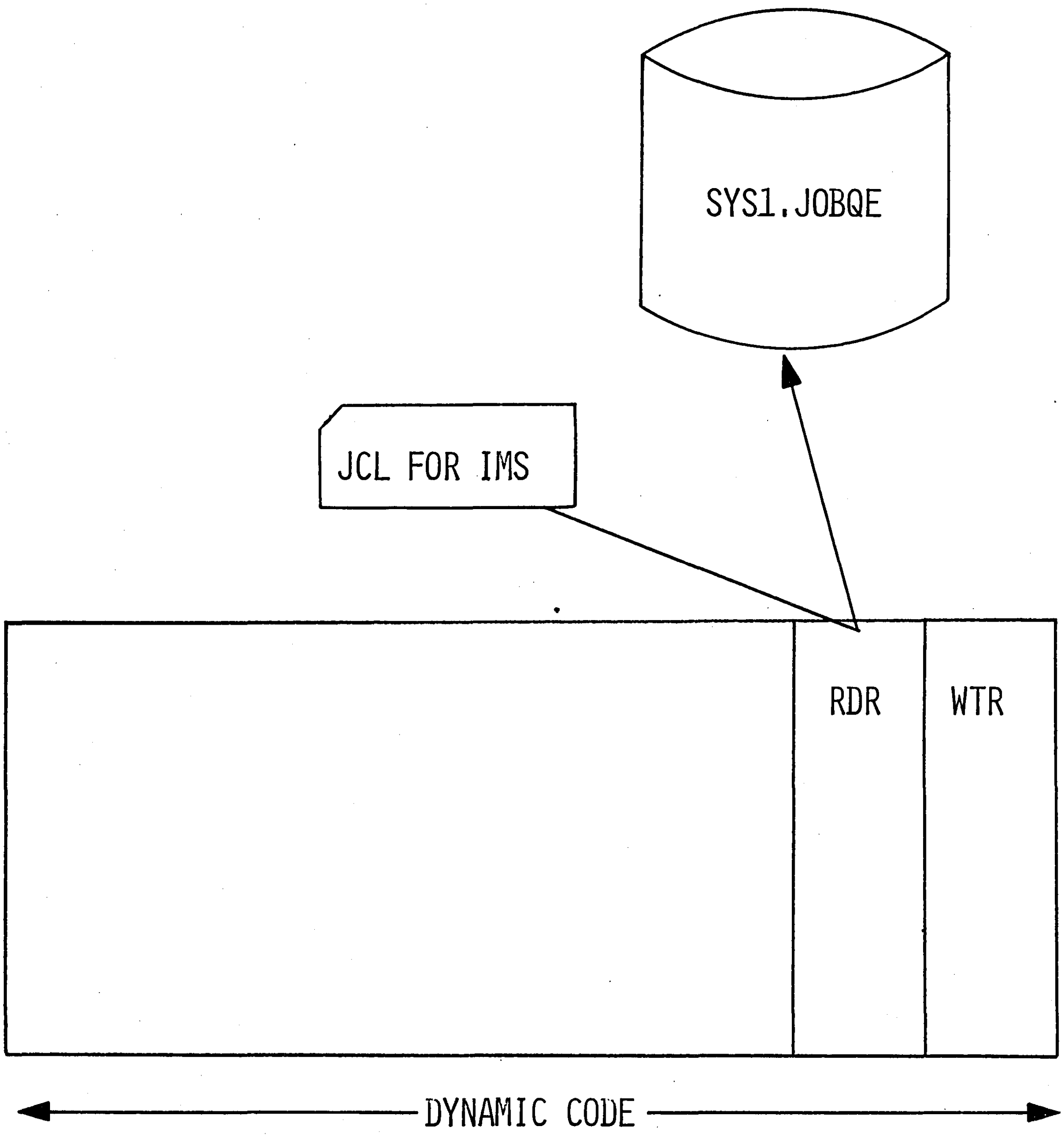
7. The MVT requirements exceed the MFT-II requirements as a result of the system queue space required and the additional requirements for system fetch work area and ABEND work area. This will vary from approximately 3,000 bytes to 4,000 bytes.

IPL

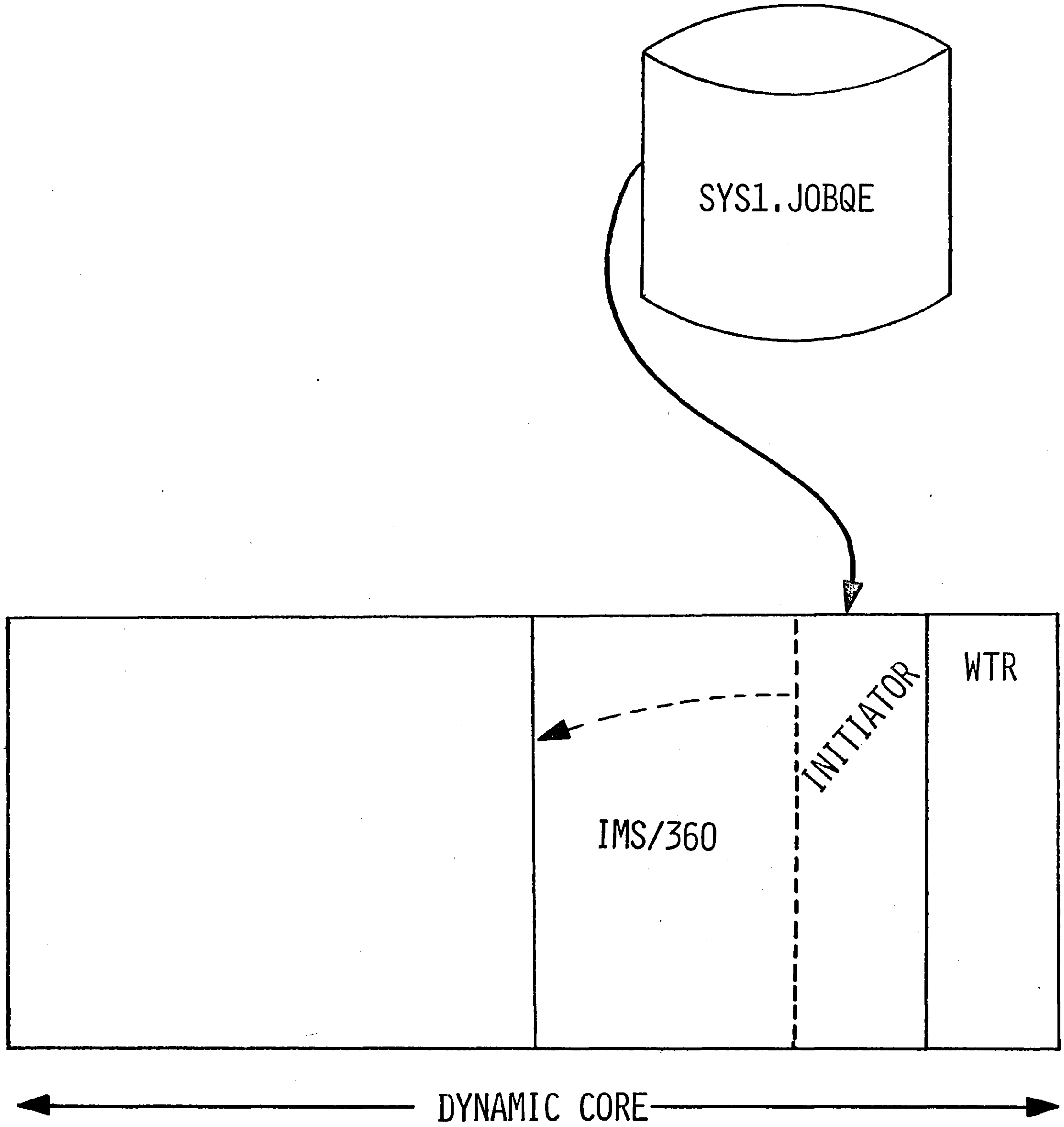

SYSRES
SYS1.NUCLEUS
SYS1.SVCLIB
SYS1.LINKLIB


OS/360
SYSTEM QUEUE SPACE
RDR
WTR
LINK PACK
DYNAMIC CORE

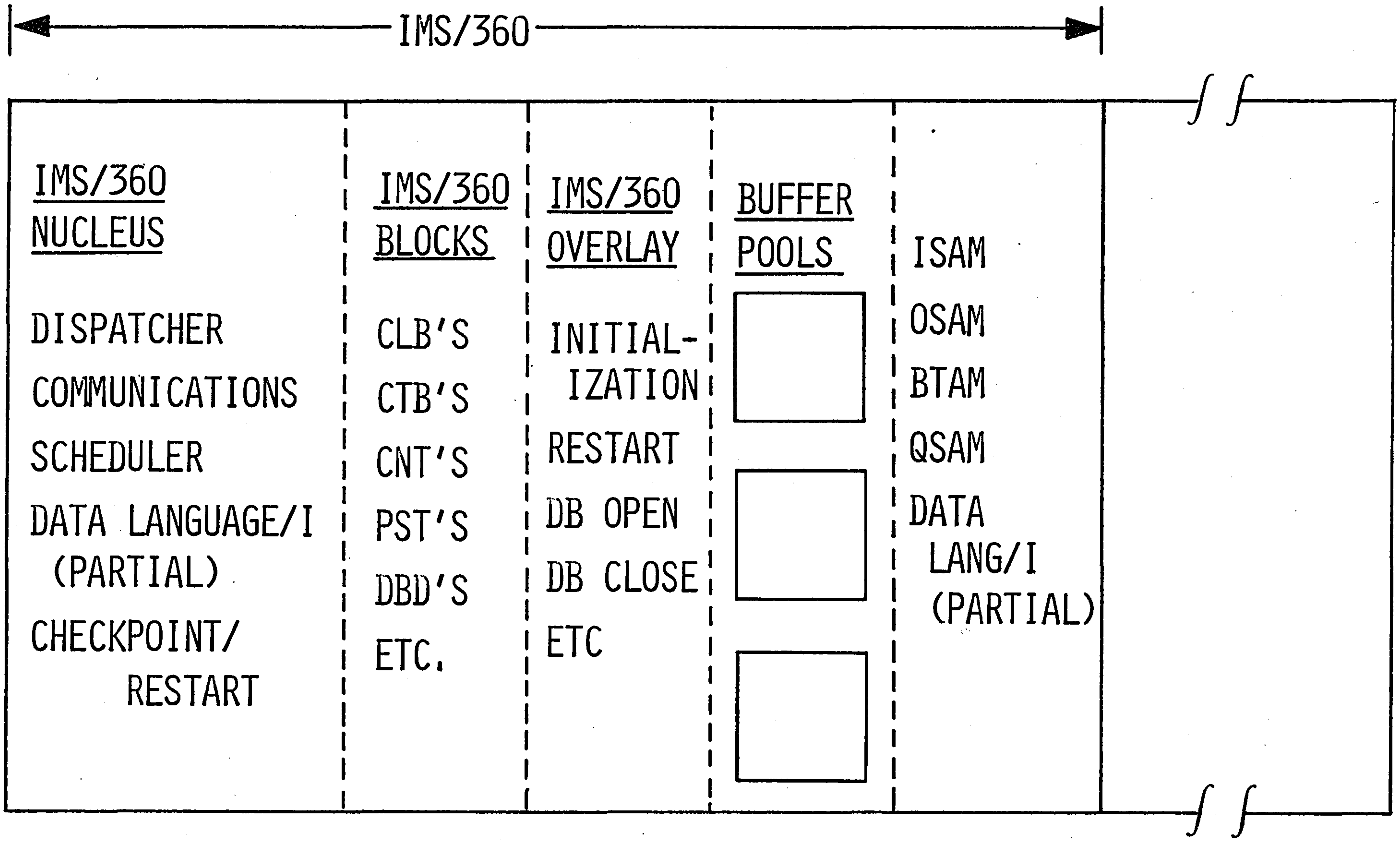
JCL FROM CARD READER



IMS/360 IS INITIATED



PORTION OF DYNAMIC CORE

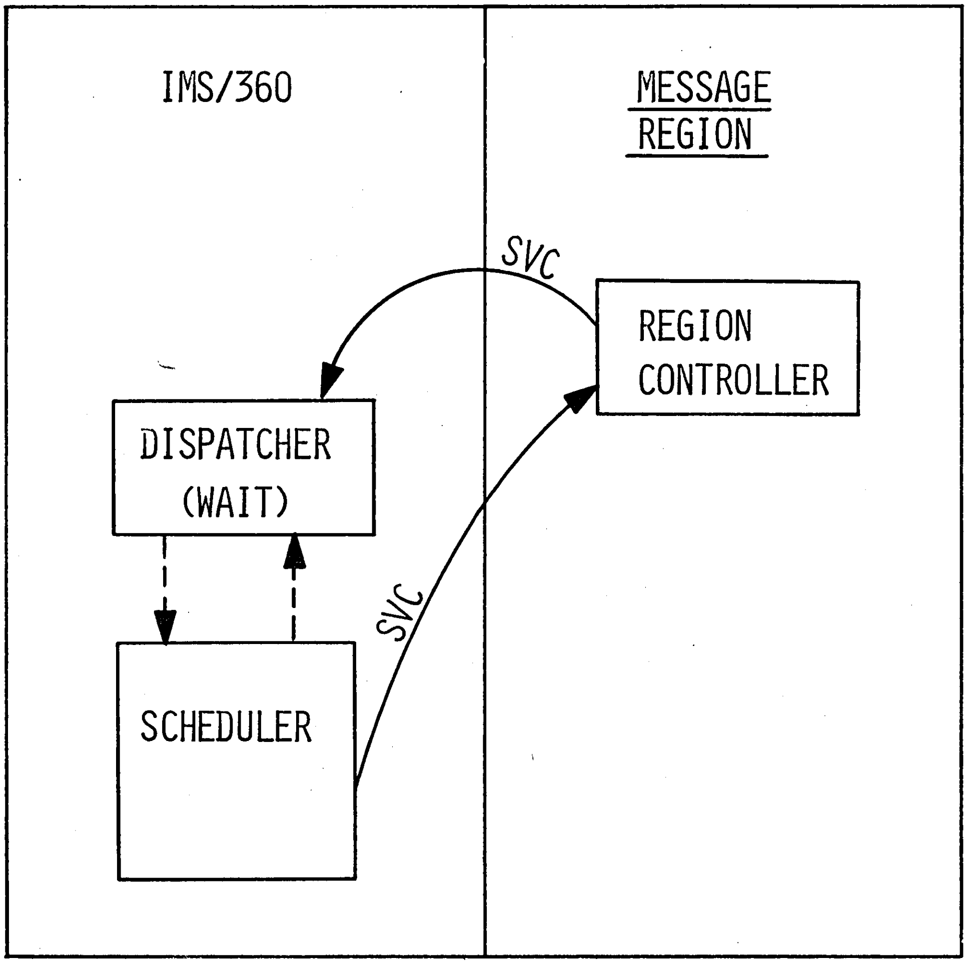


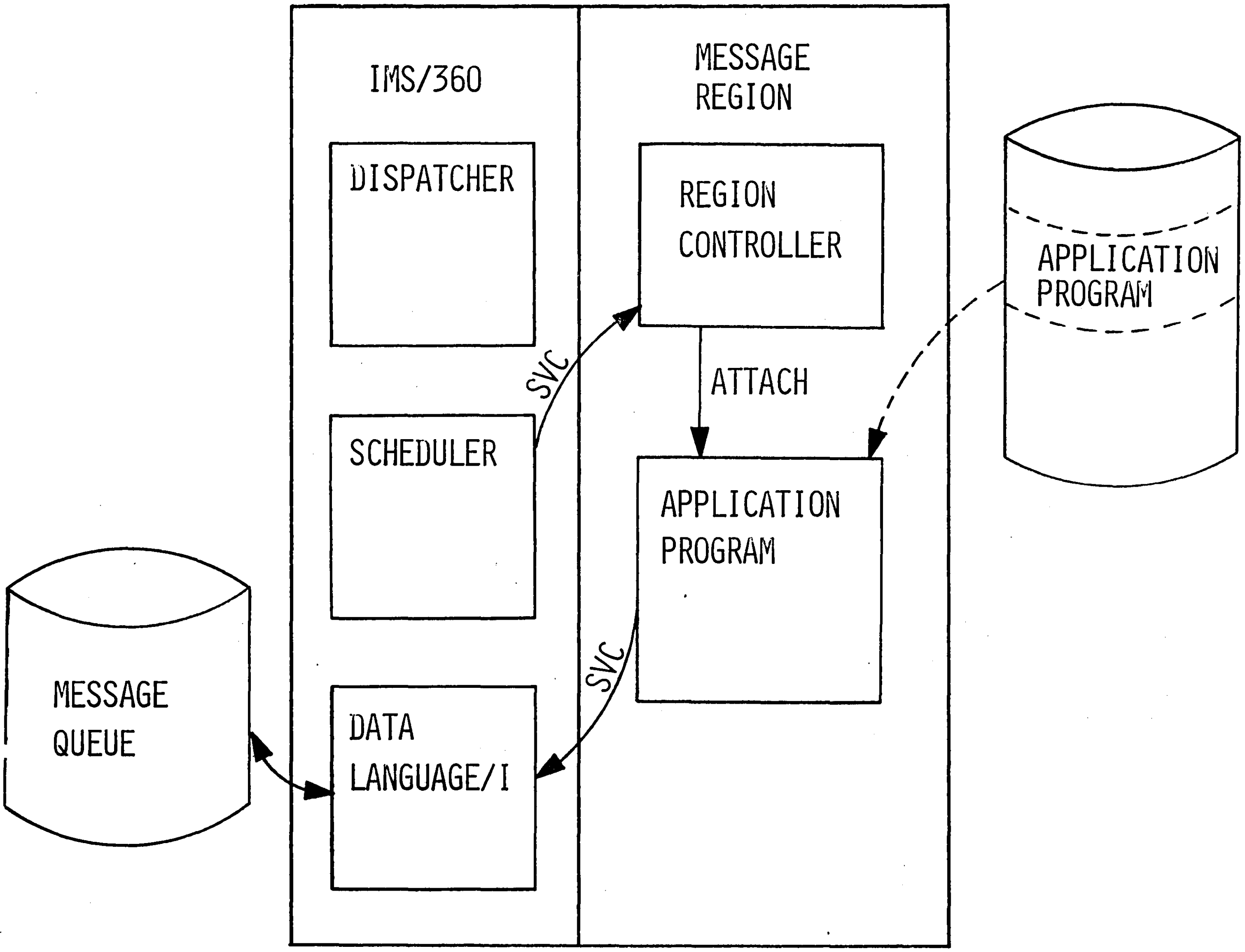
(CAN BE IN
LINK PACK)

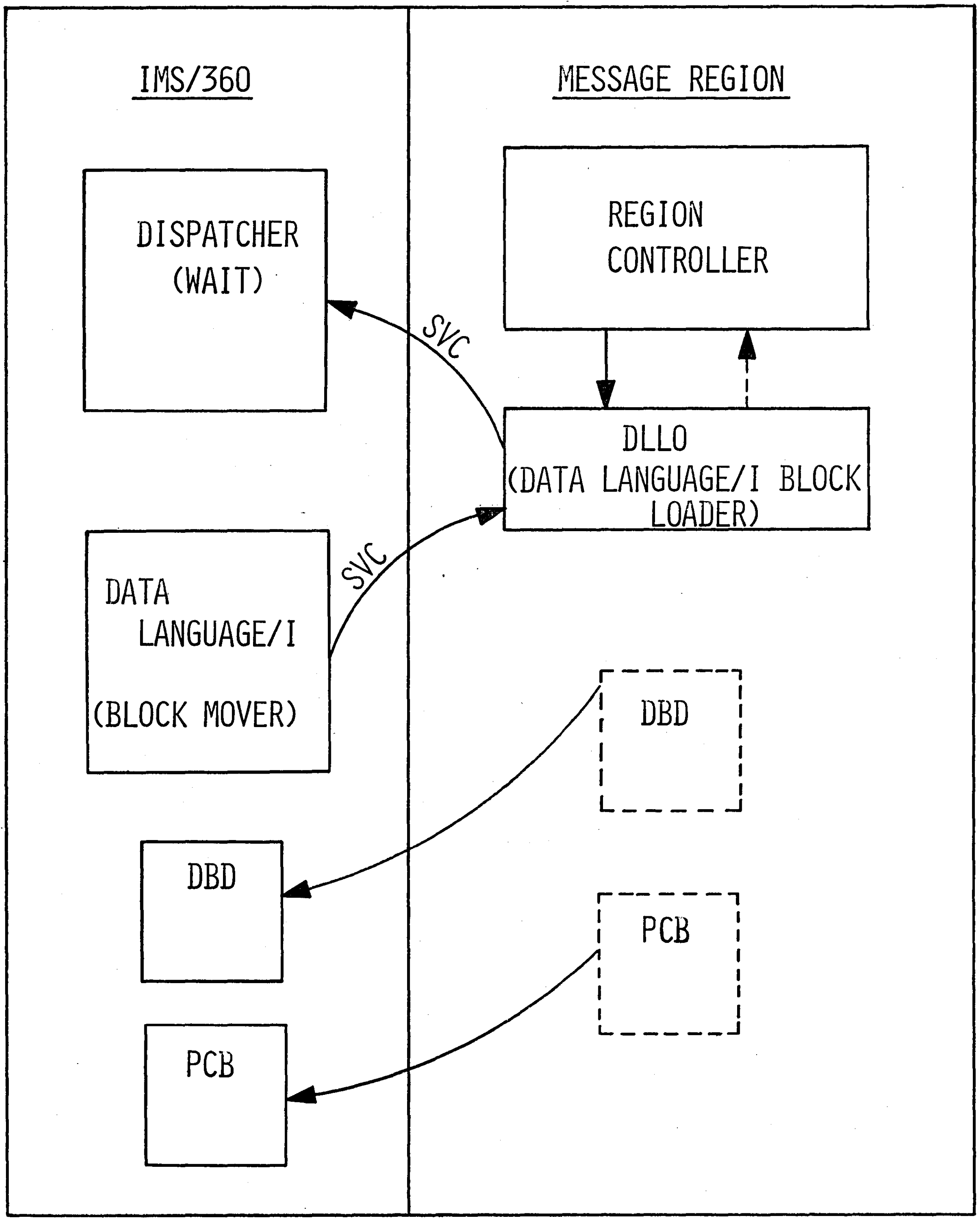
MASTER TERMINAL MESSAGES AND COMMANDS

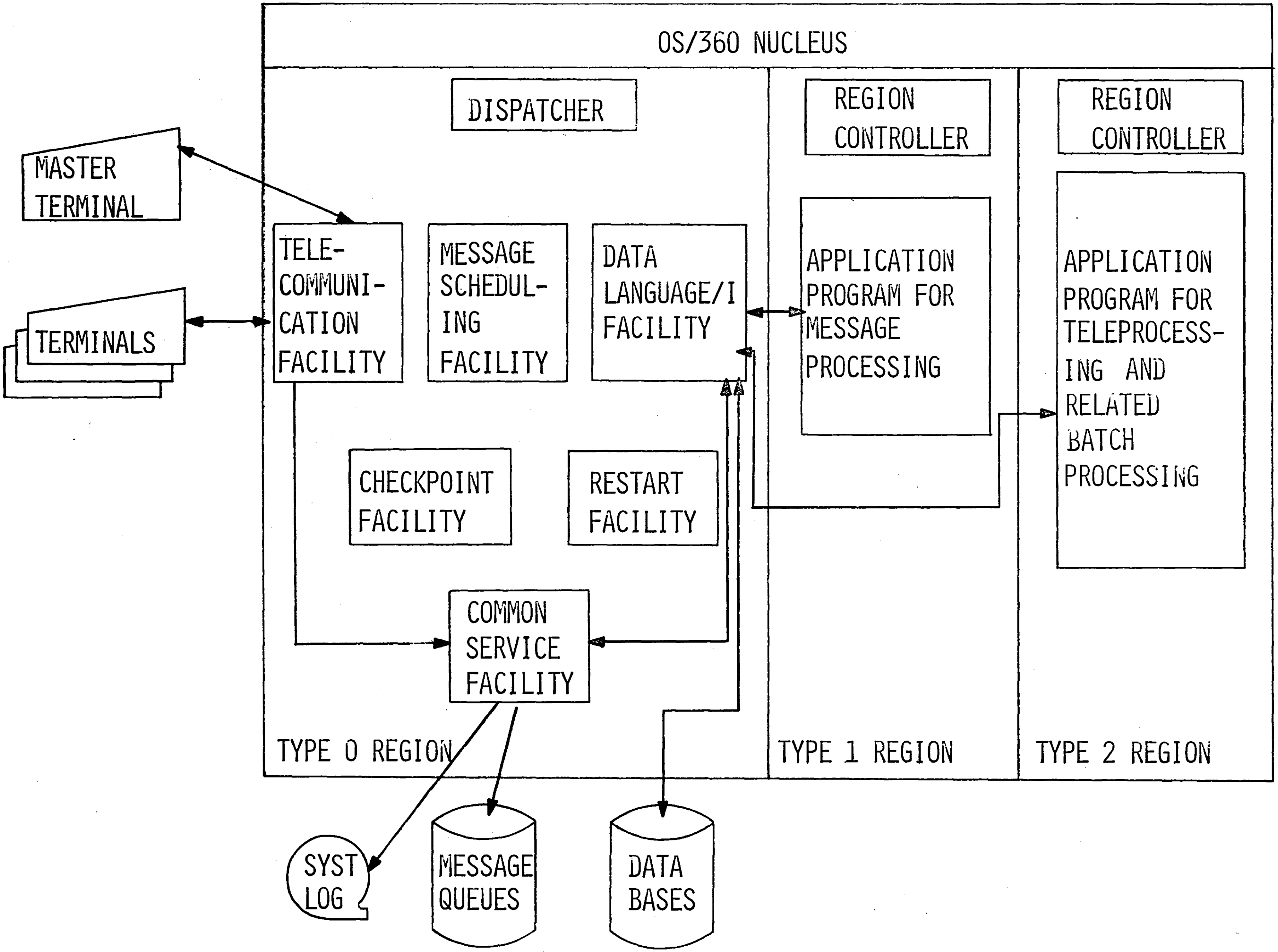
```
IMS READY    068092/1159308

/NRESTART CHKPT 0 FORMAT ALL  [TERMINAL] [PASSWORD]

*NRESTART COMMAND IN PROGRESS

*COLD START COMPLETED, ENTER START COMMANDS

/START LINE 2, 4, 7, 9

*START COMMAND COMPLETED

/START MSGREG

/START MSGREG IN PROGRESS

IMS116I MESSAGE PROCESSING REGION STARTED
```

INTER-REGION COMMUNICATION



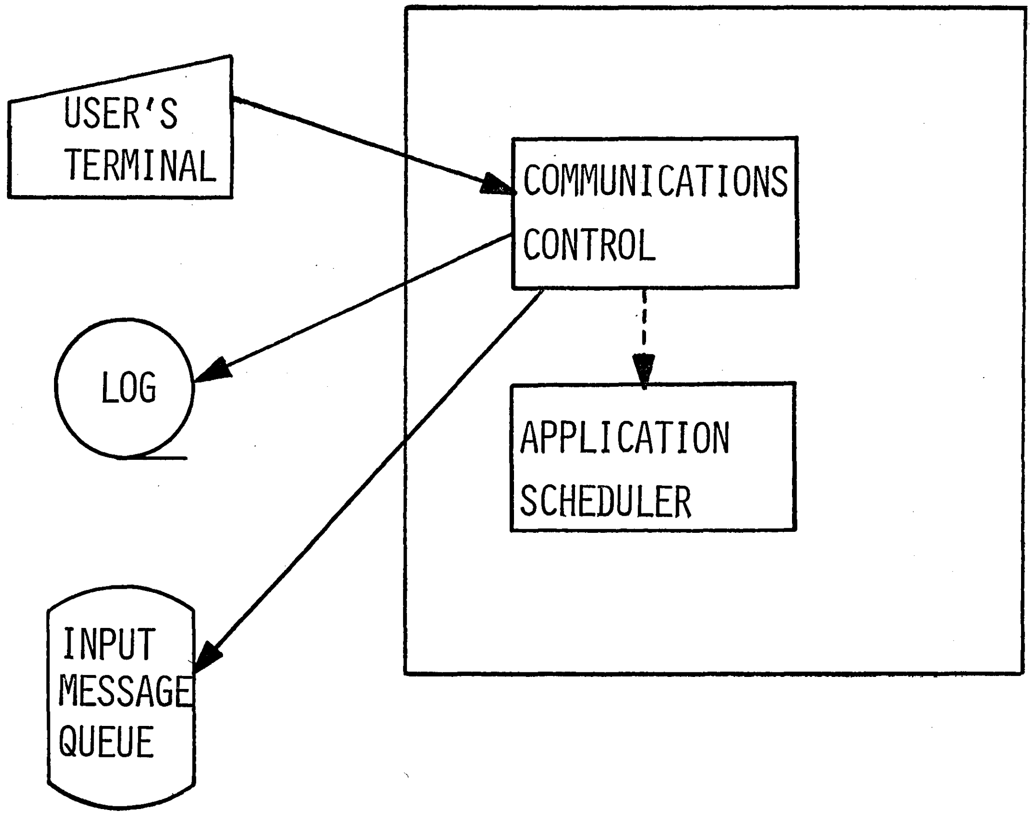

IMS/360
DISPATCHER
SCHEDULER
DATA LANGUAGE/I
MESSAGE REGION
REGION CONTROLLER
ATTACH
APPLICATION PROGRAM
SVC
SVC
MESSAGE QUEUE
APPLICATION PROGRAM


IMS/360
MESSAGE REGION
DISPATCHER
(WAIT)
SVC
REGION
CONTROLLER
DLLO
(DATA LANGUAGE/I BLOCK
LOADER)
SVC
DATA
LANGUAGE/I
(BLOCK MOVER)
DBD
DBD
PCB
PCB



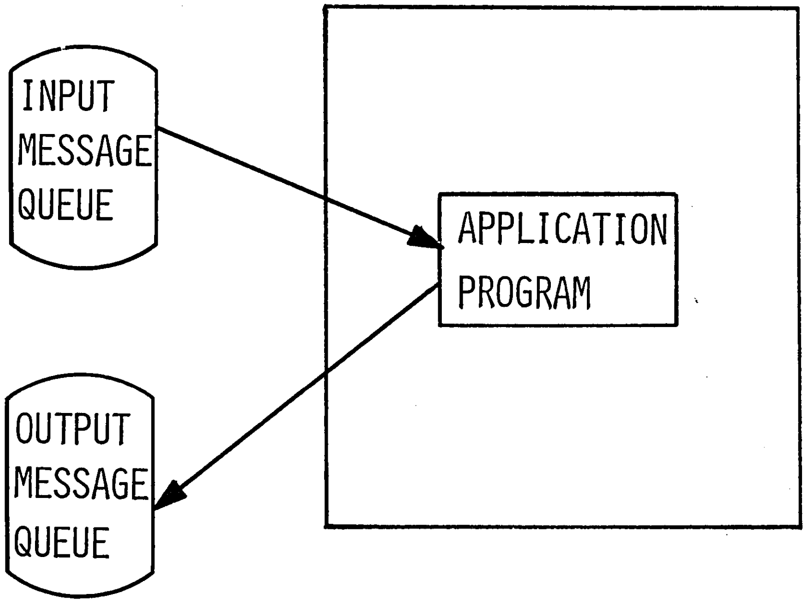
TELEPROCESSING AND RELATED BATCH IMS/360 SYSTEM FLOW

COMMUNICATIONS CONTROL

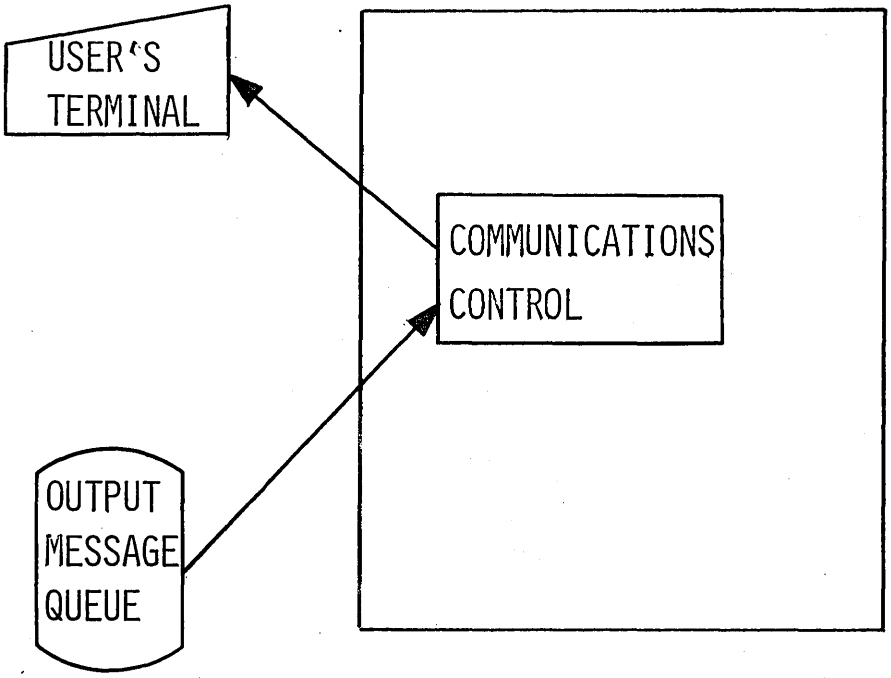


- RECEIVE MESSAGE
- TRANSLATE TO EBCDIC
- DETERMINE DESTINATION
- CHECK SECURITY REQUIREMENTS
- WRITE ON LOG
- QUEUE ON RANDOM ACCESS DEVICE
- NOTIFY APPLICATION SCHEDULER

PROCESSING A MESSAGE

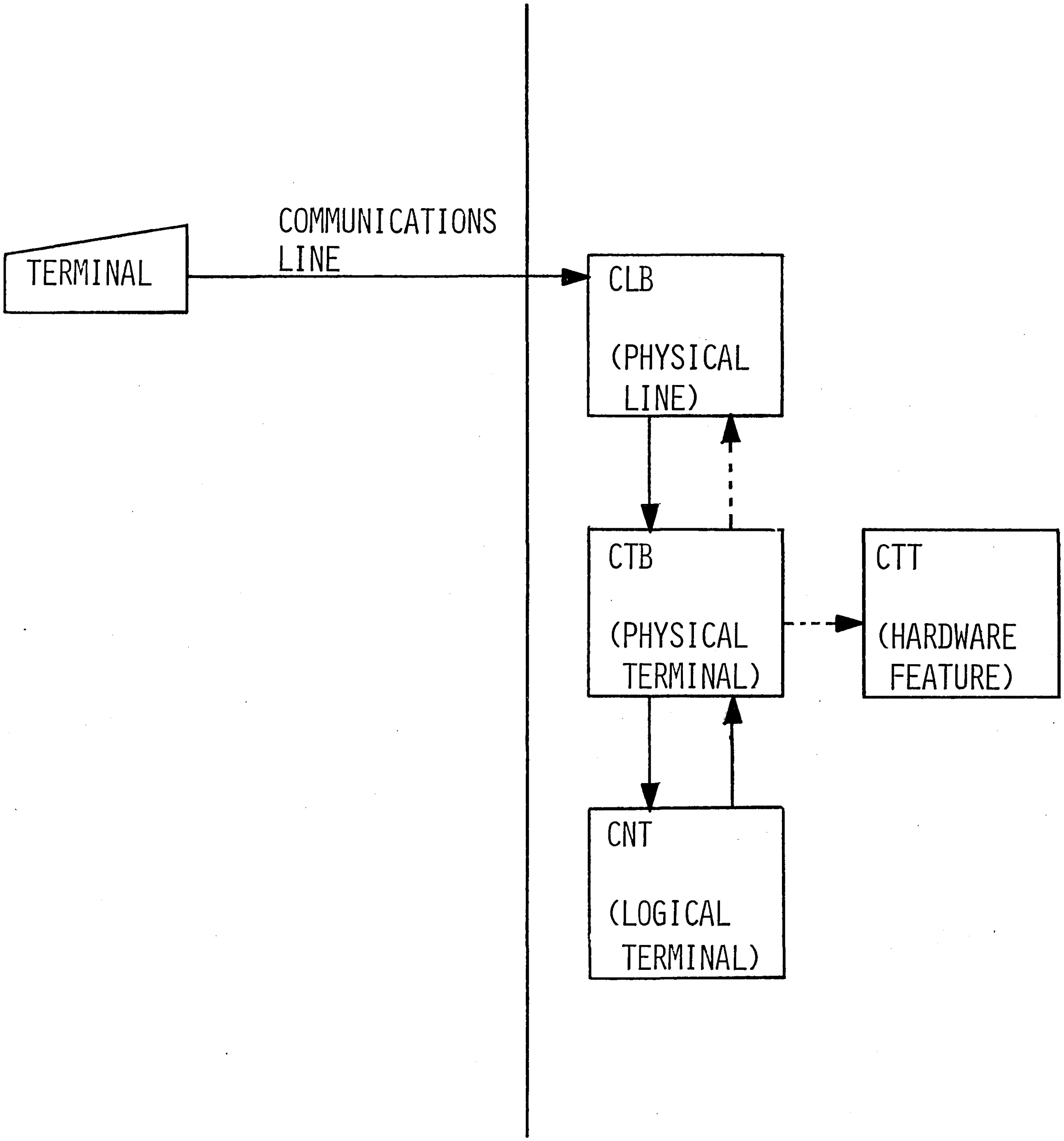


COMMUNICATIONS CONTROL

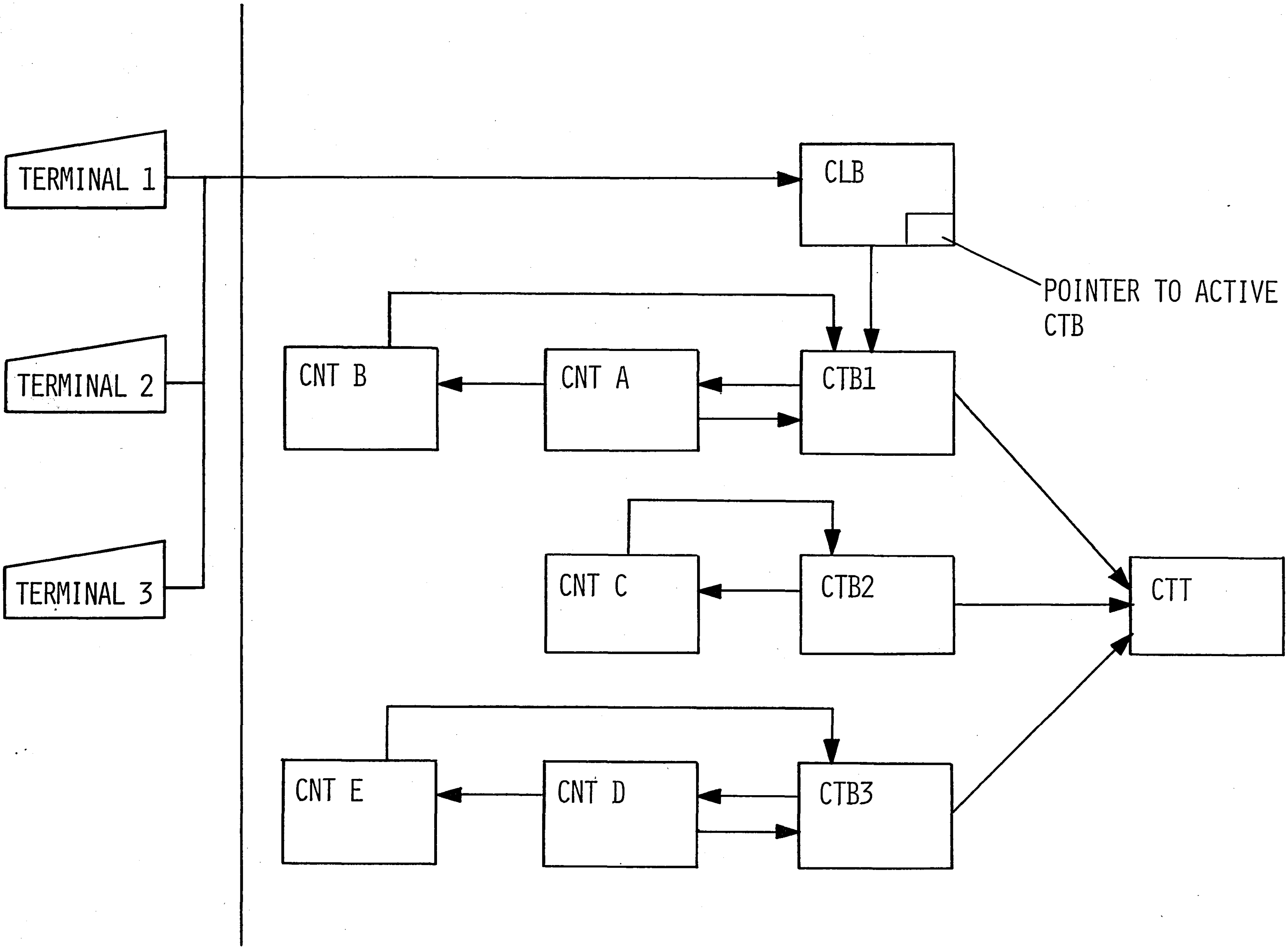


- DEQUEUE MESSAGE
- TRANSLATE TO TERMINAL CODE
- EDIT FOR CONTROL CHARACTER INSERTION
- SEND TO TERMINAL

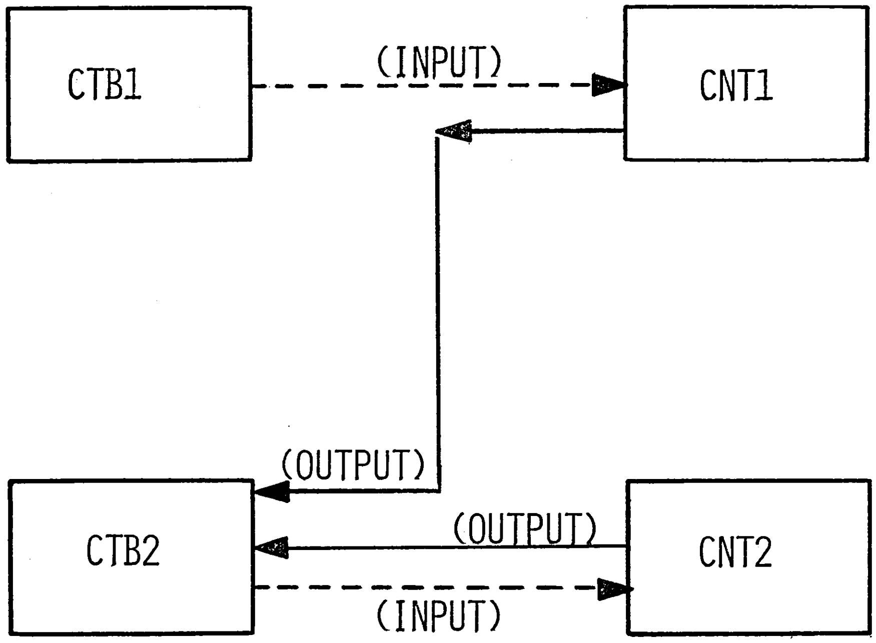
COMMUNICATIONS CONTROL BLOCKS



COMMUNICATIONS CONTROL BLOCKS



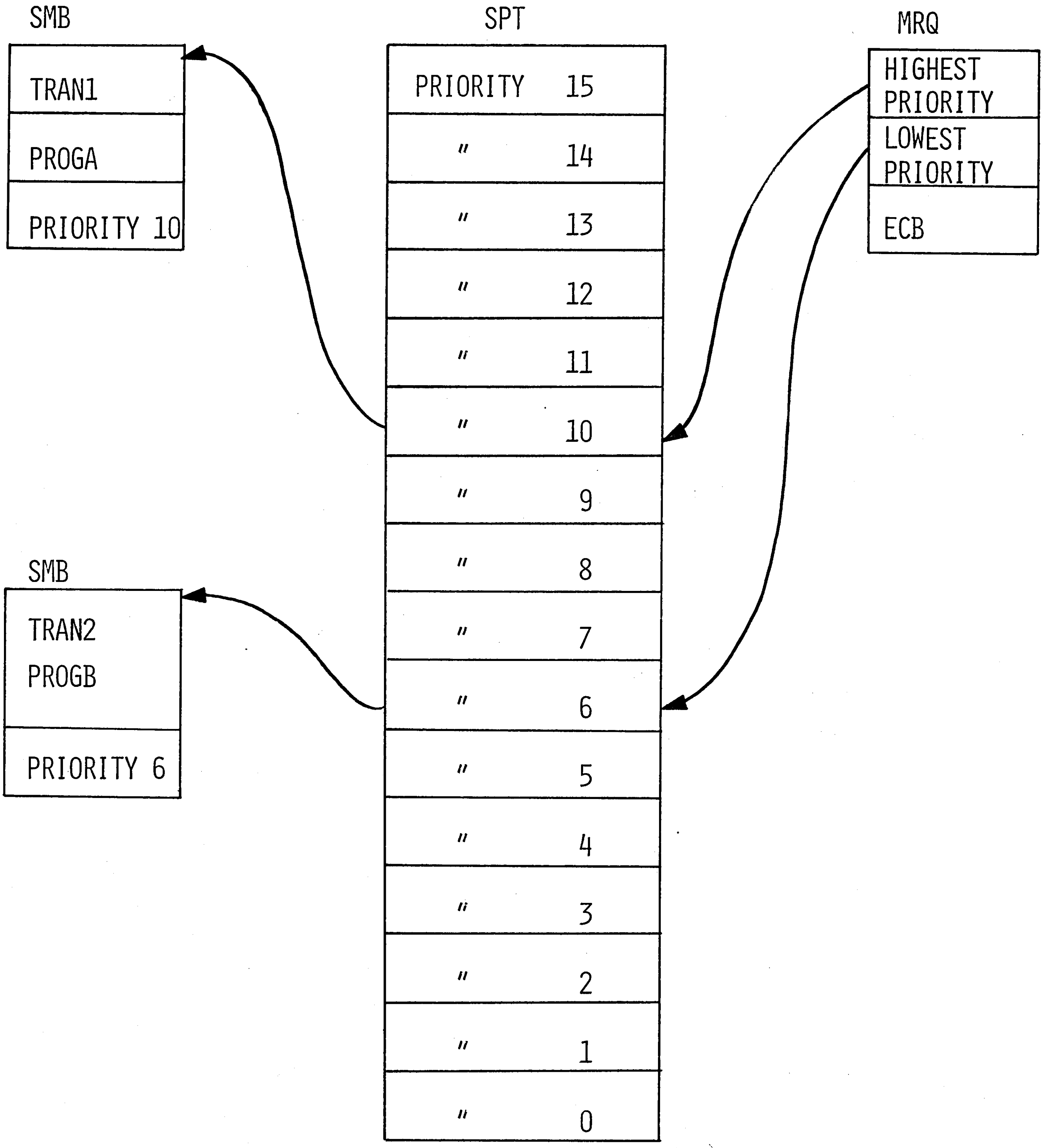
INPUT ON CTB1 - OUTPUT ON CTB2



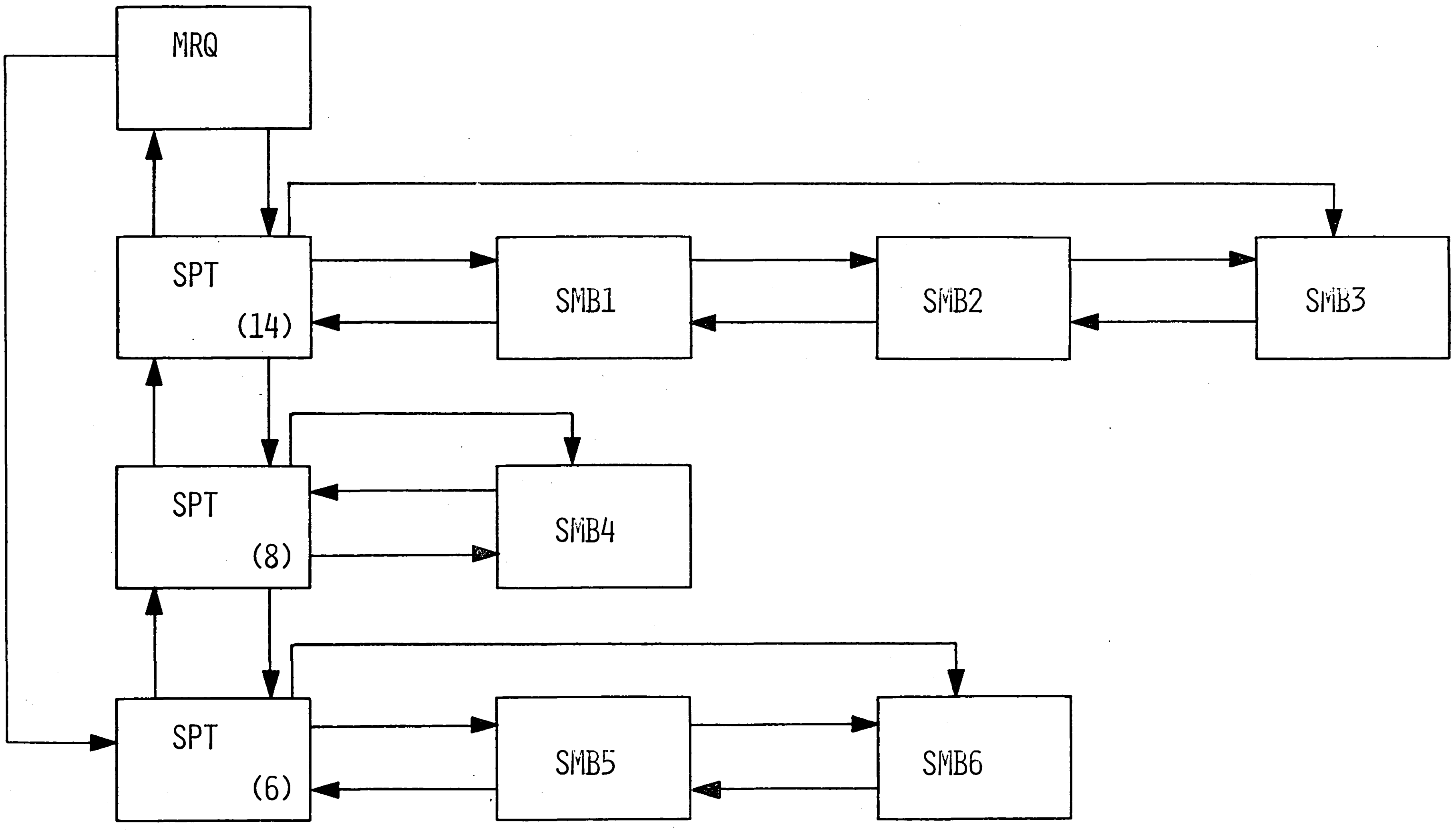
```
/ASSIGN  LTERM CNT1 TO LINE 2 PTERM CTB1 PTERM CTB2
```

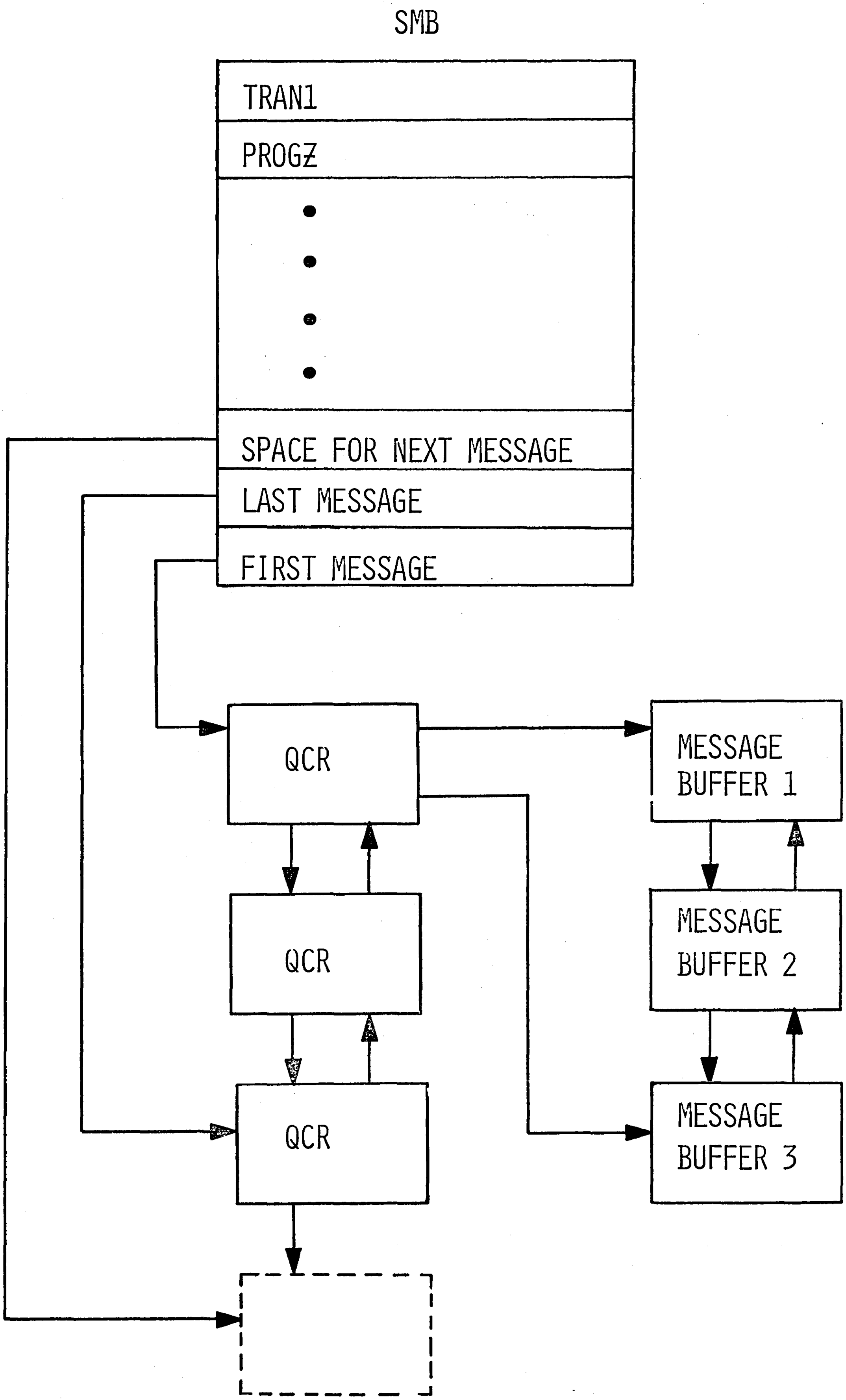
(BOTH CTB'S ARE ON THE SAME LINE IN THIS EXAMPLE)

QUEUING MESSAGES

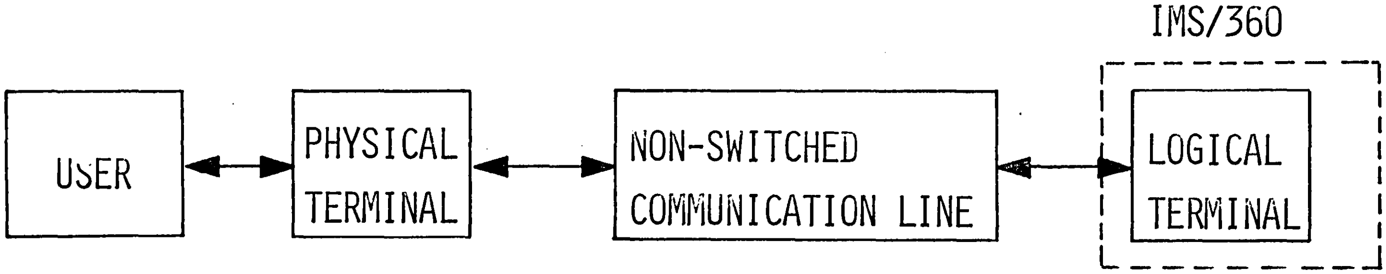


QUEUING MRQ-SPT-SMB

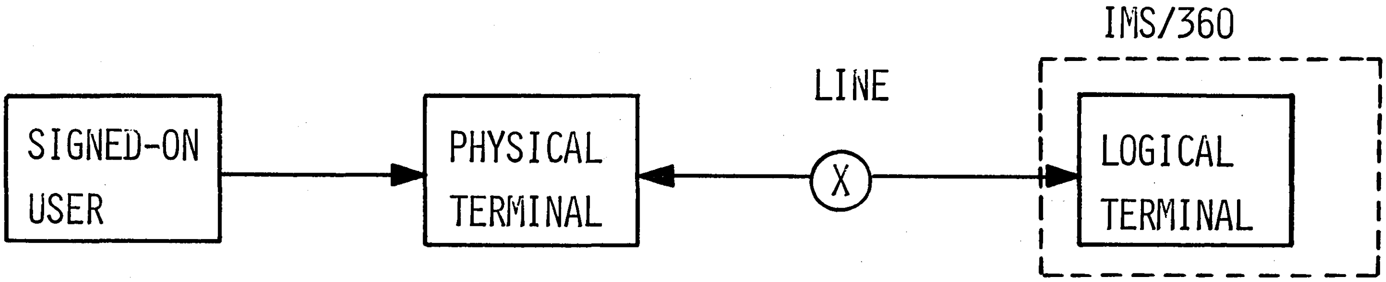



SMB
TRAN1
PROGZ
SPACE FOR NEXT MESSAGE
LAST MESSAGE
FIRST MESSAGE
QCR
QCR
QCR
MESSAGE BUFFER 1
MESSAGE BUFFER 2
MESSAGE BUFFER 3

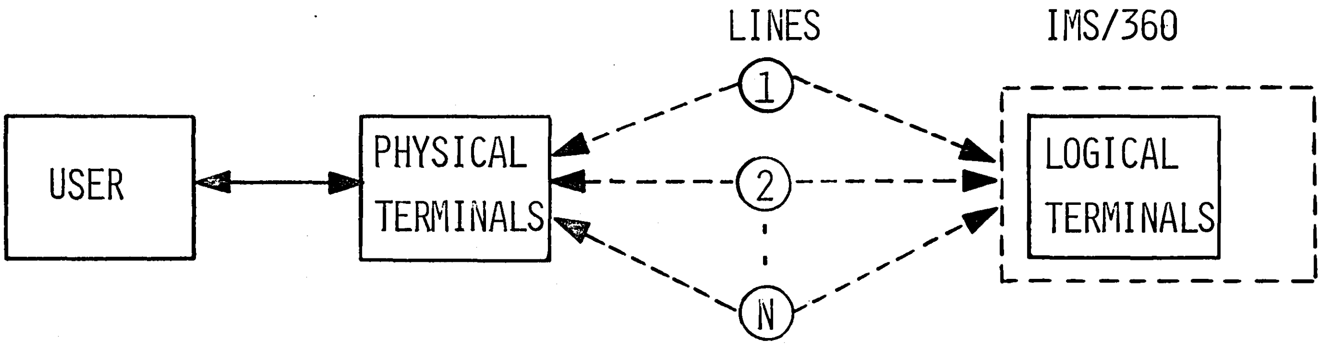
NON-SWITCHED PHYSICAL-LOGICAL RELATIONSHIP



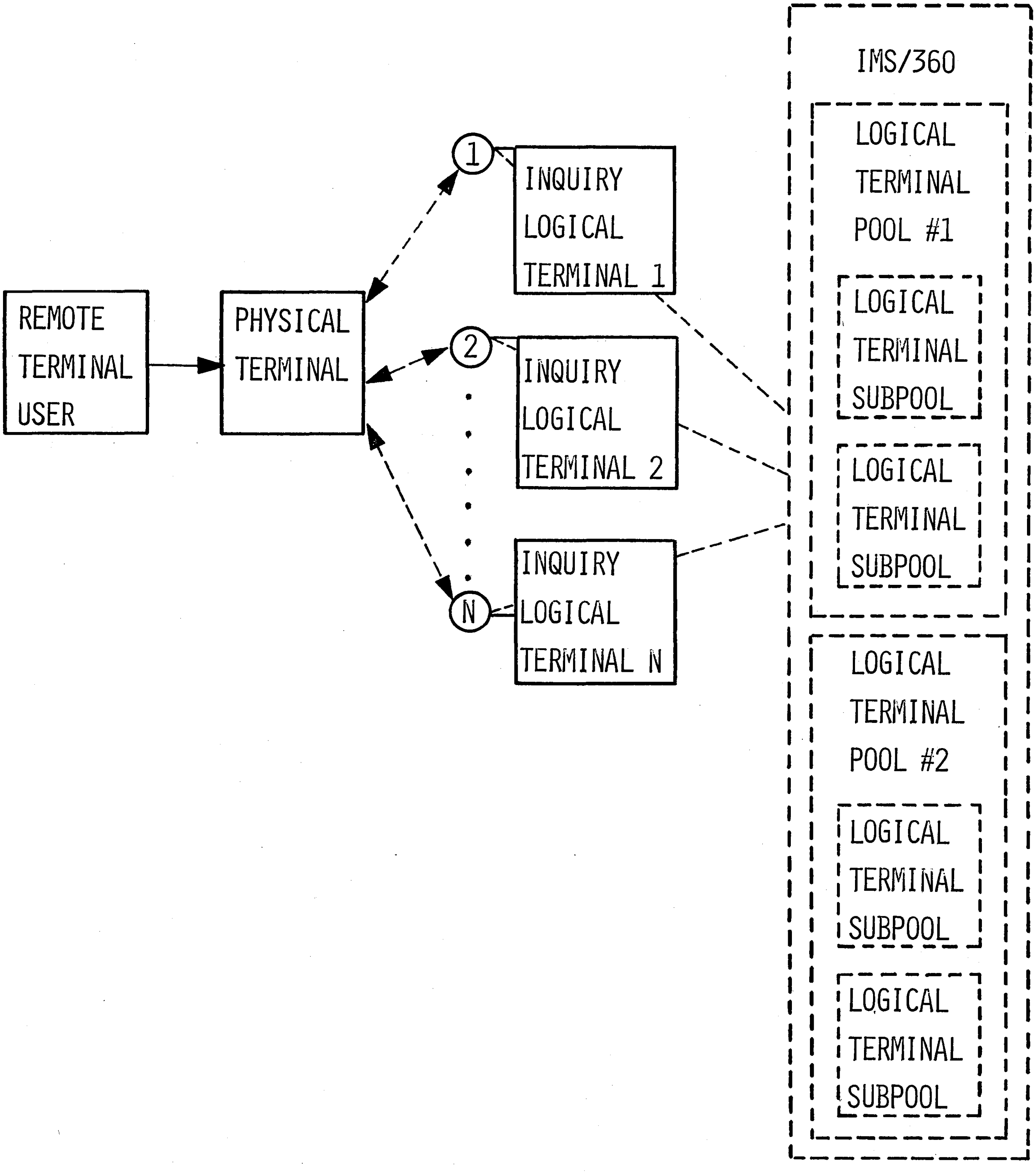
SWITCHED LINE RELATIONSHIP



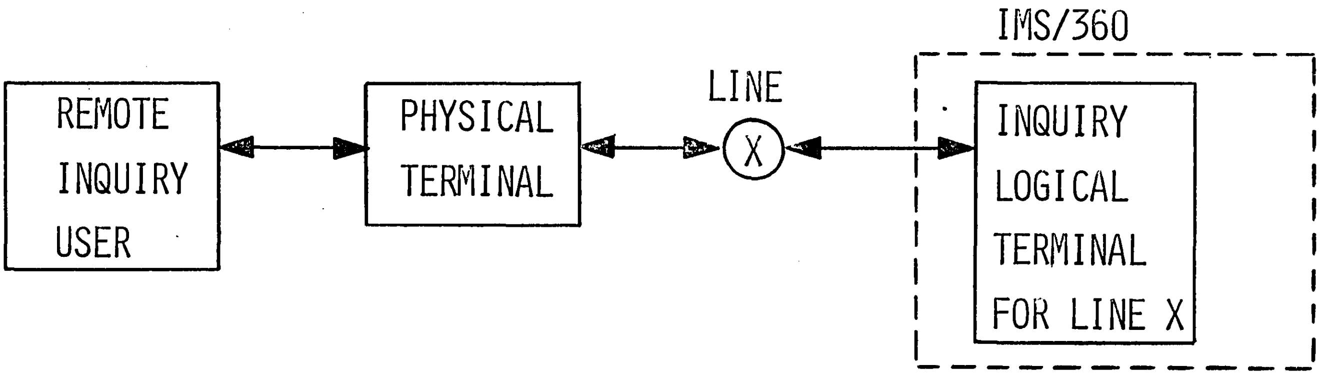
SWITCHED NETWORK



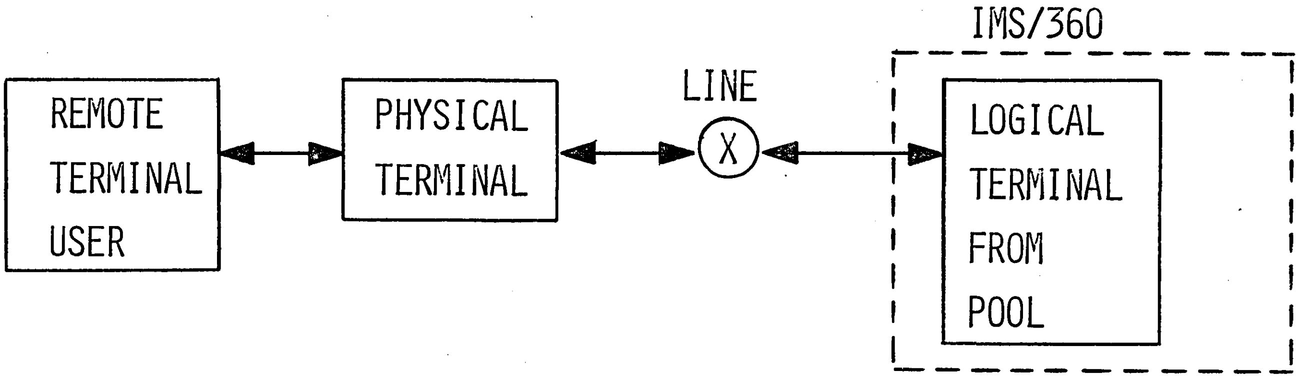
POOLS AND SUBPOOLS



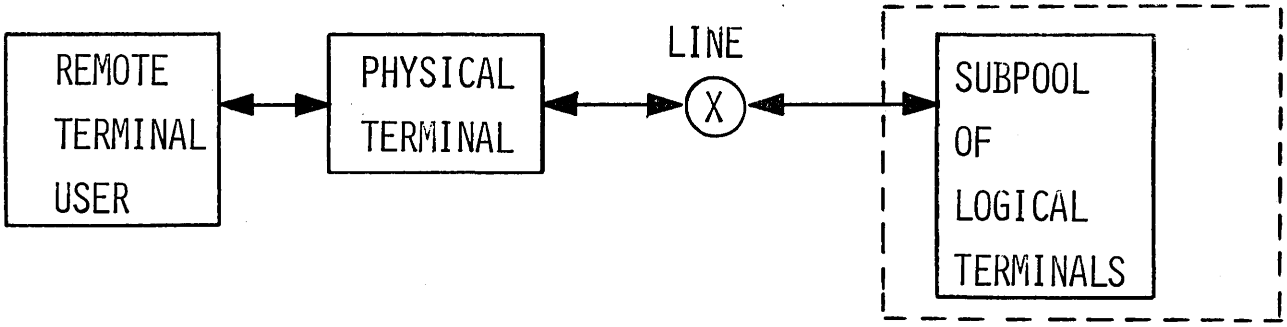
/IAM LTERM INQLOG1



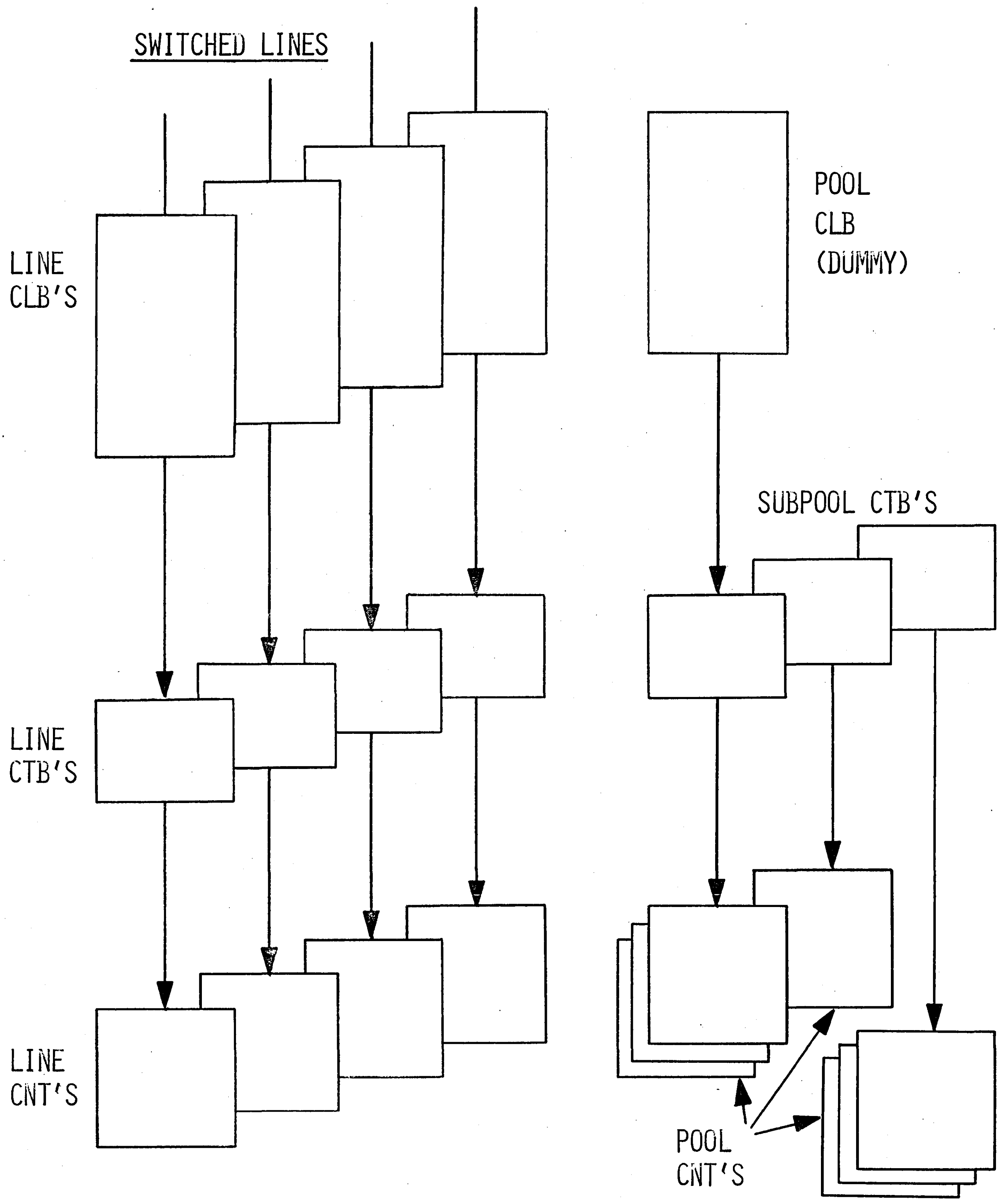
/IAM LTERM LOGPOOL1



/IAM PTERM LTERM ABCX

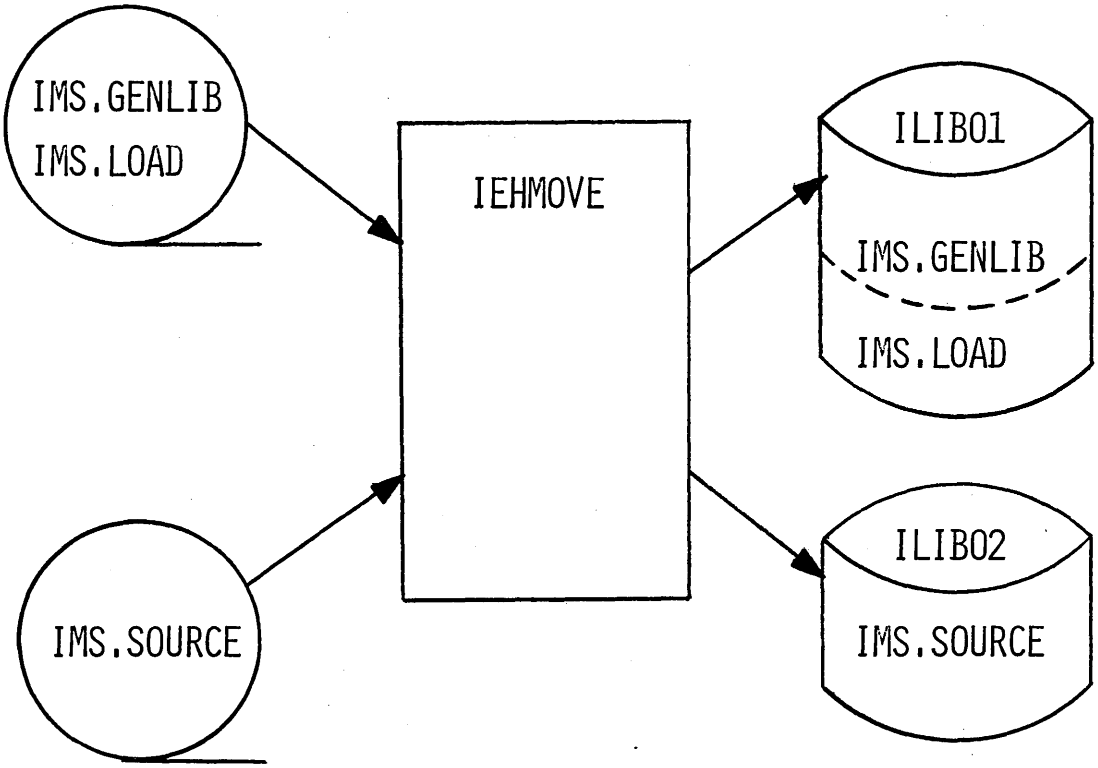


IMS SWITCHED LINE GROUP COMMUNICATIONS BLOCKS

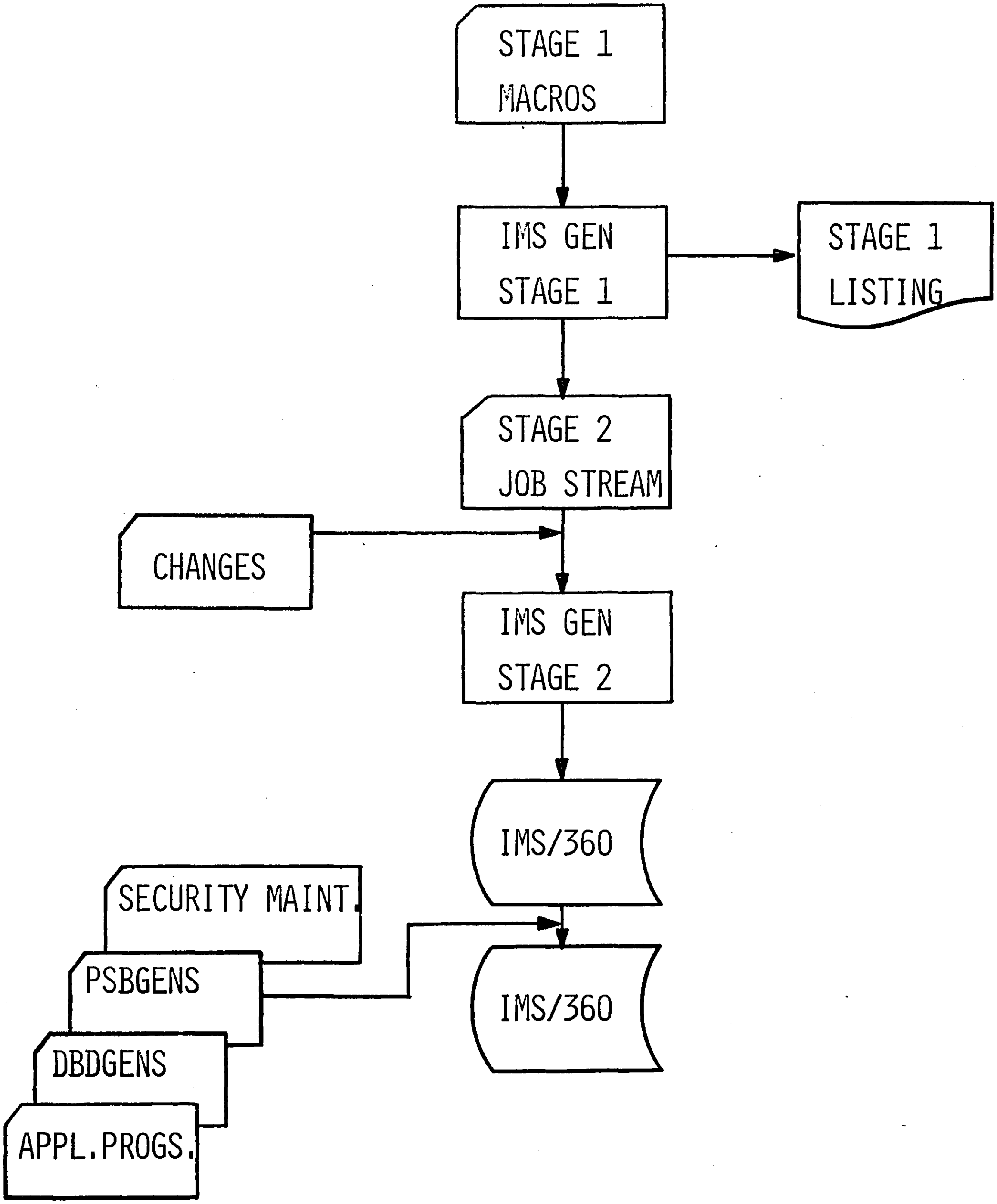


DIAL SIGN ON PROCEDURES

```
TRAN1 TEXT
*SIGN ON REQUIRED

IAM A 1050
*SIGN ON REQUIRED

IAM A SWITCHED NETWORK 1050
*SIGN ON REQUIRED

IAM NOT A 1050
*SIGN ON REQUIRED

HE IS
*SYSTEM DISCONNECT


/IAM LTERM  MASTER
*PTERM/LTERM IN USE, CANNOT PROCESS COMMAND

/IAM LOGTR  TERM01
*REQUIRED KEYWORD NOT PRESENT

/IAM LTERM TRANT1
*LTERM KEYWORD PARAMETER  INVALID
```


IMS.GENLIB
IMS.LOAD
IMS.SOURCE
IEHMOVE
ILIB01
IMS.GENLIB
IMS.LOAD
ILIB02
IMS.SOURCE

IMS GENERATION FLOW



SYSTEM DEFINITION MACROS

	MACRO INSTRUCTION	TYPE OF DEFINITION	
		COMPLETE SYSTEM	BATCH TYPE 3
	IMSCTRL	REQUIRED 1	REQUIRED 1
	MSGQUEUE	REQUIRED 1	NOT ALLOWED
	MACLIB	OPTIONAL 1	OPTIONAL 1
	RESLIB	OPTIONAL 1	OPTIONAL 1
1	PGMLIB	OPTIONAL 1	OPTIONAL 1
	PSBLIB	OPTIONAL 1	OPTIONAL 1
	DBDLIB	OPTIONAL 1	OPTIONAL 1
	PROCLIB	OPTIONAL 1	OPTIONAL 1
	IMSGEN	REQUIRED 1	REQUIRED 1
	APPLCTN	REQUIRED N	NOT ALLOWED
2	DATABASE	REQUIRED N	NOT ALLOWED
	TRANSACT	REQUIRED N	NOT ALLOWED
	LINEGRP	REQUIRED N	NOT ALLOWED
	LINE	REQUIRED N	NOT ALLOWED
	TERMINAL	REQUIRED N	NOT ALLOWED
3	NAME	REQUIRED N	NOT ALLOWED
	MASTTERM	REQUIRED 1	NOT ALLOWED
	POOL	OPTIONAL N	NOT ALLOWED
	SUBPOOL	OPTIONAL N	NOT ALLOWED

	IMSCTRL	SYSTEM = $\left[\begin{Bmatrix}\underline{\text{MVT}}\\ \text{MFT-II}\\ \text{PCP}\end{Bmatrix}\ ,\ \begin{matrix}\underline{\text{ALL}}\\ \text{BATCH}\end{matrix}\right]$
		MAXIQ = NUMBER
		MAXREGN = NUMBER
		COMMSVC = (NUMBER , NUMBER)
		OCENDA = APPENDAGE SUFFIX
		OSAMSVC = NUMBER
		MSGBUFF = NUMBER
		CKPT = $\begin{matrix}\underline{1000}\\ \text{NUMBER}\end{matrix}$

```
IMSCTRL  SYSTEM = (MFT-II,ALL), MAXIO=10,MAXREGN=2,       C
         COMMSVC=(244,245),OCENDA=Z8,OSAMSVC=243,         C
         MSGBUFF=5,CKPT=500
```

NAME	MSGQUEUE	QCRIN = (DDNAME,DSNAME,UNIT,SERIAL),
		QCROUT = (DDNAME,DSNAME,UNIT,SERIAL)
		MSGIN = (DDNAME,DSNAME,UNIT,SERIAL),
		MSGOUT = (DDNAME,DSNAME,UNIT,SERIAL)
		REUSE = (YES,100
		NO,N)

```
         MSGQUEUE    QCRIN  =  (SMSGIN,IMS.SMSGIN,2311,IMS001),
                     QCROUT =  (SMSGOUT,IMS.SMSGOUT,2311,IMS001),
                     MSGIN  =  (MMSGIN,IMS.MMSGIN,2311,IMS002),
                     MSGOUT =  (MMSGOUT,IMS.MMSGOUT,2311,IMS002),
                     REUSE  =  YES,20
```

NAME	MACLIB	UNIT = NAME
		VOLNO = SERIAL
		PDS = IMS.MACLIB
		NAME
		COPY = UTILITY
		ALL

```
          MACLIB        UNIT=2314,VOLNO=IMFMFT,PDS=IMS.MACLIB,
                        COPY=UTILITY
```

PROCLIB	--	IMS.PROCLIB
DBDLIB	--	IMS.DBDLIB
PSBLIB	--	IMS.PSBLIB
PGMLIB	--	IMS.PGMLIB
RESLIB	--	IMS.RESLIB

NAME	IMSGEN	UT1SDS = DSNAME
		ASMPRT = OFF
		ON ,
		LEPRT = (OPTION, OPTION)

```
          IMSGEN        UT1SDS=SYSUT1,ASMPRT=ON,
                           LEPRT=(LIST,XREF)
```

MACRO-INSTRUCTION	NUMBER PER SET	PURPOSE
APPLCTN	1	NAMES APPLICATION PROGRAM. DELIMITS THIS SET OF MACRO-INSTRUCTIONS.
DATABASE	N	NAMES DATA BASES USED BY APPLICATION PROGRAM.
TRANSACT	N	NAMES TRANSACTION CODES WHICH WILL BE PROCESSED BY THE ABOVE APPLICATION PROGRAM.

APPLICATION DESCRIPTION MACRO-INSTRUCTION SET

NAME	APPLCTN	PSB = PSBNAME
		PGMTYPE = { TP
		BATCH }

ANY	APPLCTN	PSB = HIMAKR01

NAME	DATABASE	DBD = DBDNAME
		INTENT = { UPDATE
		SHARE
		EXCLUSIVE }
		LOG = { YES
		NO }

DATA1	DATABASE	DBD = DI21PART,INTENT=SHARE,
		LOG=NO

NAME	TRANSACT	CODE = TRANSACTION CODE
		PRTY = (NORMAL, LIMIT, LIMIT COUNT)
		MSGTYPE = { (MULTSEG,NONRESPONSE)
		(SNGLSEG,RESPONSE) }
		PROCLIM = (SECONDS, COUNT)
		INQUIRY = { NO
		YES }

TRAN1	TRANSACT	CODE=PART,PRTY=(4,10,8),
		MSGTYPE=(SNGLSEG,NONRESPONSE),
		PROCLIM=(2,6)

MACRO-INSTRUCTION	NUMBER PER SET	PURPOSE
LINEGRP	1	NAMES COLLECTION OF TERMINALS WITH LIKE ATTRIBUTES. DELIMITS THIS SET OF MACRO-INSTRUCTIONS.
LINE	N	PROVIDES ADDRESS OF LINE AND DELIMITS TERMINALS ON SAME LINE.
TERMINAL	N	PROVIDES PHYSICAL TERMINAL DATA AND DELIMITS LOGICAL TERMINAL NAME.
NAME	N	PROVIDES LOGICAL TERMINAL NAMES
MASTTERM	1 (PER DEFINITION)	LOGICAL NAME OF MASTER TERMINAL
POOL	1	POOL OF LOGICAL TERMINALS
SUBPOOL	1	SUBPOOL OF LOGICAL TERMINALS

NAME	LINEGRP	DDNAME = NAME
		FEAT = $\left\{\begin{array}{l}\underline{\text{STACTL, NONSWITCH}}\\ \text{TRANSCTL, SWITCHED}\end{array}\right\}$
		UNITYPE = $\left\{\begin{array}{l}\underline{\text{2740}}\\ \text{1052}\end{array}\right\}$

```
LING1         LINEGRP         DDNAME=DD1,FEAT=(TRANSCTL,SWITCHED),
                              UNITYPE=2740
```

TERMINAL TYPE

	1050	1050	
		2740	2740
OPERAND	STACTL SWITCHED	STACTL NONSWITCHED	TRANSCTL SWITCHED
AUTOCALL	OPTIONAL	NOT ALLOWED	OPTIONAL
AUTOANS	OPTIONAL	NOT ALLOWED	OPTIONAL
AUTOPOLL	NOT ALLOWED	OPTIONAL	NOT ALLOWED

NAME	LINE	AUTOCANS
		AUTOCALL
		FEAT = AUTOPOLL
		POLL
		ADDR = HEXNUMBER
		ZONE = NUMBER

L1	LINE	FEAT = POLL,ADDR=03B

NAME	TERMINAL	ADDR = TERMINAL ADDRESS CHARACTER
	TERMINAL	ADDR = E2

NAME	NAME	LTERM NAME
N22	NAME	L2740S2

NAME	MASTTERM	LOGICAL NAME
	MASTTERM	MASTER

NAME	POOL	ZONE = NUMBER
		FEAT = AUTOANS
		AUTOCALL

```
P1        POOL        FEAT = AUTOCALL
```

NAME	SUBPOOL	TELNO = NUMBER

```
          SUBPOOL     TELNO = NUMBER
```

GROUP 1 CREATE RESLIB, MACLIB, AND RESLIB

STEP 1 MOVES LOAD MODULES TO RESLIB

STEP 2 LINK EDITS THE OSAM SVC ROUTINE, IF BATCH ONLY WILL ALSO LINK EDIT DFSIRCOO, DFSIPCOO, AND DFSIDLLO WITH DUMMY INTER-REGION SVC'S

STEP 3 MOVES MACROS TO MACLIB

STEP 4 ADDS IMS PROCEDURES TO PROCLIB

STEP 5 ASSEMBLES THE BATCH SCD

STEP 6 LINK EDITS THE BATCH SCD WITH THE BATCH NUCLEUS

GROUP II DL/I CONTROL BLOCKS

STEPS 7 & 8 ASSEMBLE AND LINK EDIT DFSIDIRO AND INCLUDES PSB DIRECTORY, DMB LISTS, AND DMB DIRECTORY

STEPS 9 & 10 ASSEMBLE AND LINK EDIT DFSISMBO

GROUP III USER SPECIFIED COMMUNICATION CONTROL BLOCKS

STEPS 11 & 12 ASSEMBLE AND LINK EDIT DFSICLLO INCLUDES CLB's, POLLING LISTS, LERB's, LINE DCB's, AND THE BTAM OPEN LIST

STEPS 13 & 14 ASSEMBLE AND LINK EDIT DFSICNTO INCLUDES SKELETON DFSICTMO

STEPS 15 & 16 ASSEMBLE AND LINK EDIT DFSICTBO

GROUP IV SYSTEM SPECIFIED COMMUNICATION CONTROL BLOCKS

STEPS 17 & 18 ASSEMBLES AND LINK EDITS DFSICTTO

STEPS 19 & 20 ASSEMBLES AND LINK EDITS DFSICVBO

STEPS 21 & 22 ASSEMBLE AND LINK EDIT DFSISDBO

GROUP V SYSTEM CONTROL BLOCKS

STEPS 23 & 24 ASSEMBLE AND LINK EDIT DFSISAVO

STEPS 25 & 26 ASSEMBLE AND LINK EDIT DFSIPSTO

STEPS 27 & 28 ASSEMBLE AND LINK EDIT DFSIQUEO
INCLUDES DFSIHLDO

STEPS 29 & 30 ASSEMBLE AND LINK EDIT DFSIOS40
INCLUDES DFSIOS50

STEPS 31 & 32 ASSEMBLE AND LINK EDIT DFSISCDO
INCLUDES DFSINTBO

STEPS 33 & 34 ASSEMBLE AND LINK EDIT DFSISVAO

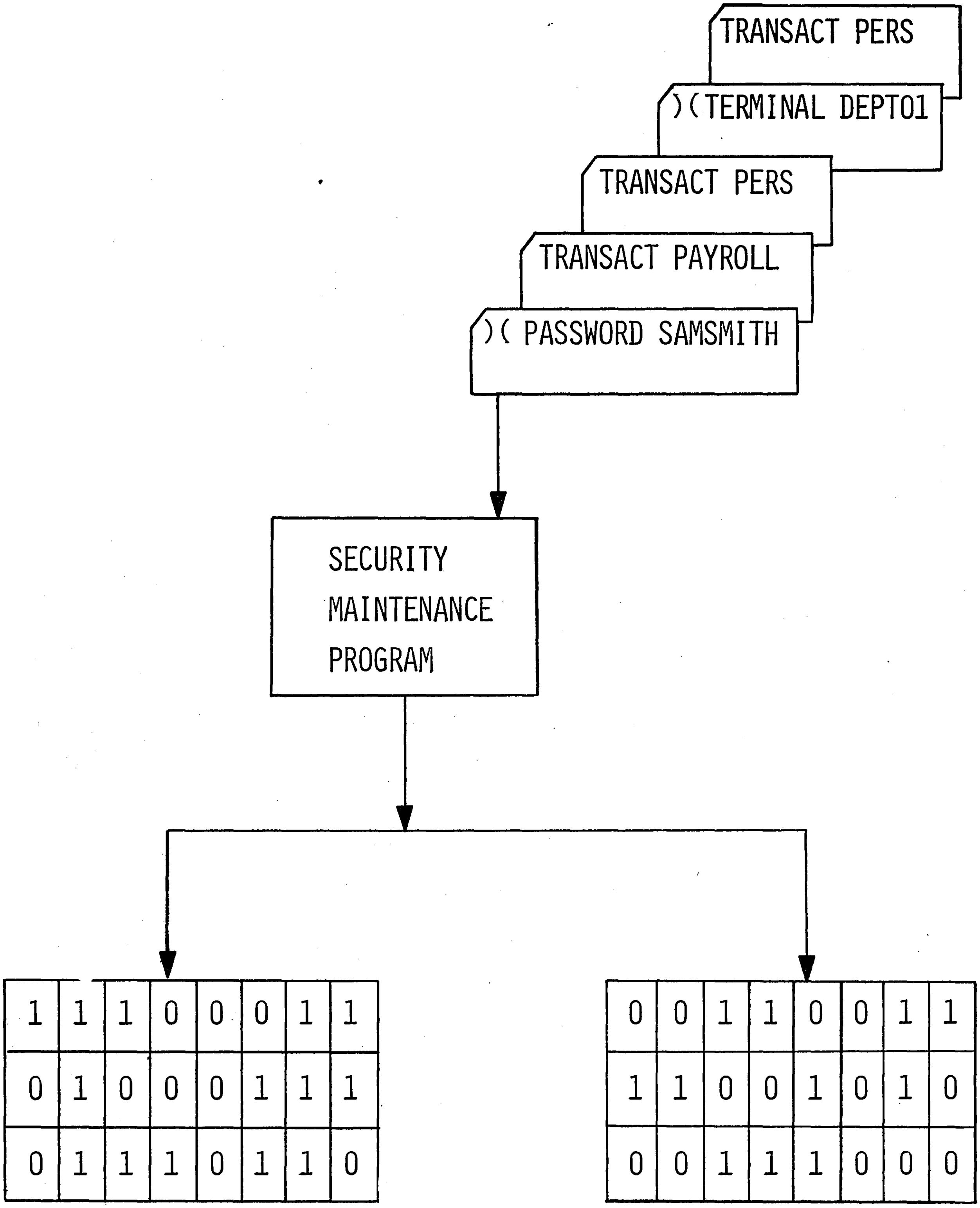
GROUP VI SYSTEM LINK EDITS

STEP 35 LINK EDITS DFSISVFO OR DFSISVVO
LINK EDITS DFSIRCOO
LINK EDITS DFSIPCOO
LINK EDITS DFSIDLLO

STEP 36 LINK EDITS DFSIBLKO

STEP 37 LINK EDITS DFSINUCO

STEP 38 PRODUCES A PDS DIRECTORY LIST OF RESLIB



1	1	1	0	0	0	1	1
0	1	0	0	0	1	1	1
0	1	1	1	0	1	1	0

PASSWORD MATRIX

0	0	1	1	0	0	1	1
1	1	0	0	1	0	1	0
0	0	1	1	1	0	0	0

TERMINAL MATRIX

SECURITY MAINTENANCE

ADD, DELETE, OR CHANGE PASSWORDS FOR THE FOLLOWING RESOURCES:

- TRANSACTION CODES
- TERMINAL COMMAND VERBS
- PROGRAMS
- DATA BASES
- LOGICAL TERMINALS
- PHYSICAL TERMINALS

SPECIFY TERMINAL SECURITY FOR;

- TRANSACTION CODES
- COMMAND VERBS

CONTROL STATEMENT	DATA STATEMENT	OPERAND
)(PASSWORD		NAME
	TERMINAL	LOGICAL TERMINAL NAME
	TRANSACT	NAME
	COMMAND	COMMAND LANGUAGE VERB
	DATABASE	NAME
	PROGRAM	NAME
	PTERM	NAME
)(TERMINAL		LOGICAL TERMINAL NAME
	PASSWORD	NAME
	TRANSACT	NAME
	COMMAND	COMMAND LANGUAGE VERB
)(TRANSACT		NAME
	PASSWORD	NAME
	TERMINAL	LOGICAL TERMINAL NAME
)(COMMAND		COMMAND LANGUAGE VERB
	PASSWORD	NAME
	TERMINAL	LOGICAL TERMINAL NAME
)(DATABASE		NAME
OR		
)(PROGRAM		NAME
OR		
)(PTERM		NAME
	PASSWORD	NAME

PASSWORD SECURITY

PASSWORD PROFILE

```
)( PASSWORD          SAMSMITH
     TRANSACT           PAYROLL
     TRANSACT           PERS
     COMMAND            LOCK
     COMMAND            UNLOCK
     DATABASE           PAYREC
     PROGRAM            PAYPROG
```

RESOURCE PROFILE

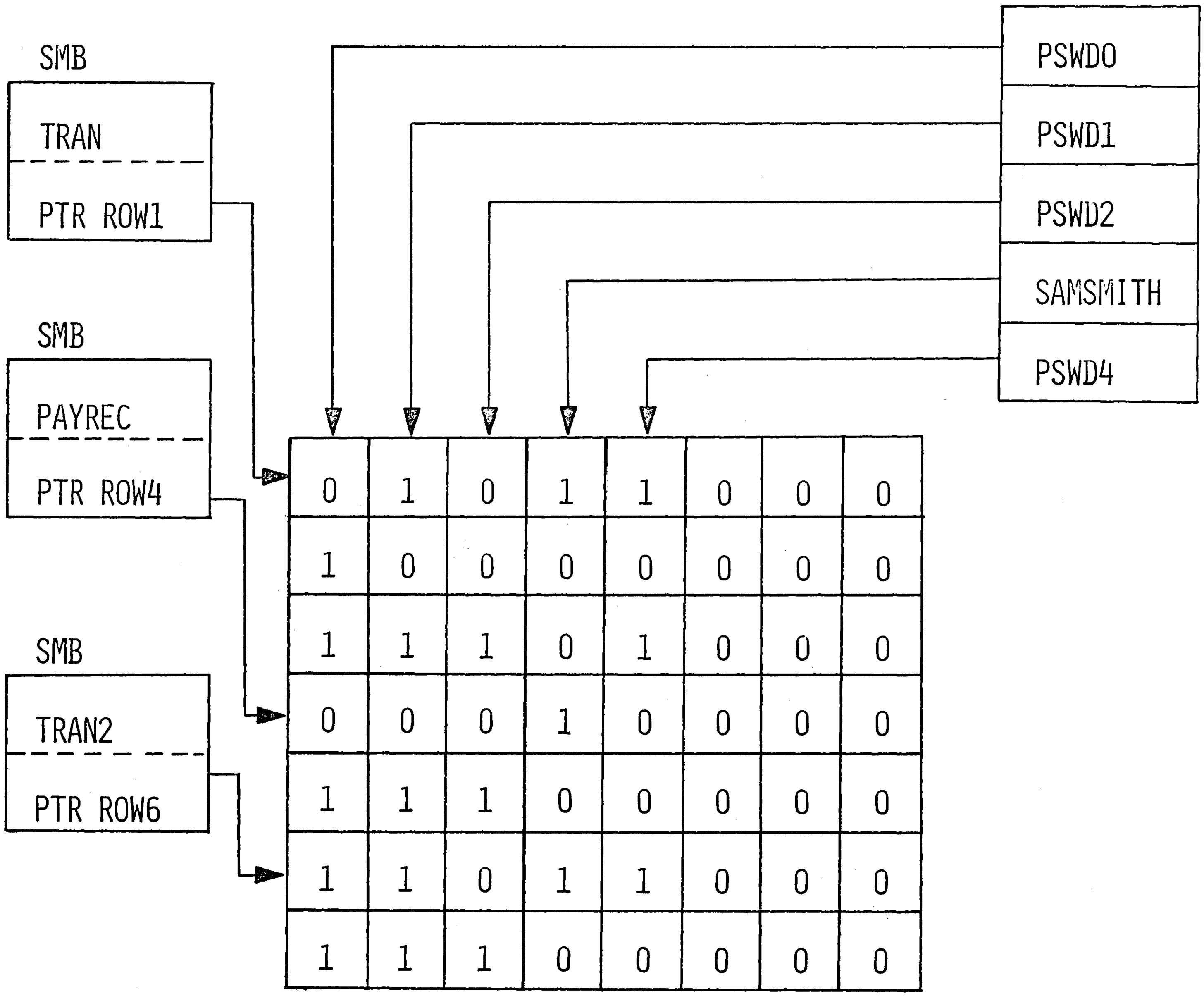
```
)( TRANSACT          PAYROLL
     PASSWORD           SAMSMITH
)( TRANSACT          PERS
     PASSWORD           SAMSMITH
)( COMMAND           LOCK
     PASSWORD           SAMSMITH
)( COMMAND           UNLOCK
     PASSWORD           SAMSMITH
)( DATABASE          PAYREC
     PASSWORD           SAMSMITH
)( PROGRAM           PAYPROG
     PASSWORD           SAMSMITH
```

PASSWORD SECURITY

	PAYPROG	PAYREC	PAYROLL	PERS	LOCK	UNLOCK
ABCXYZ	X			X		
SAMSMITH	X	X	X	X	X	X
V142613			X			
WTWT			X	X		
ZZABC	X		X			X

PASSWORD SECURITY

PAYREC (SAMSMITH) TEXT



0	1	0	1	1	0	0	0
1	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0
0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	0	0	0	0

TERMINAL SECURITY

TRANSACTION AND COMMAND PROFILE

```
)( TRANSACT          PAYROLL
     TERMINAL          DEPT40
     TERMINAL          DEPT65
     TERMINAL          VPPERS
)( TRANSACT          PERS
     TERMINAL          DEPT40
)( COMMAND             HOLD
     TERMINAL          DEPT3L
```

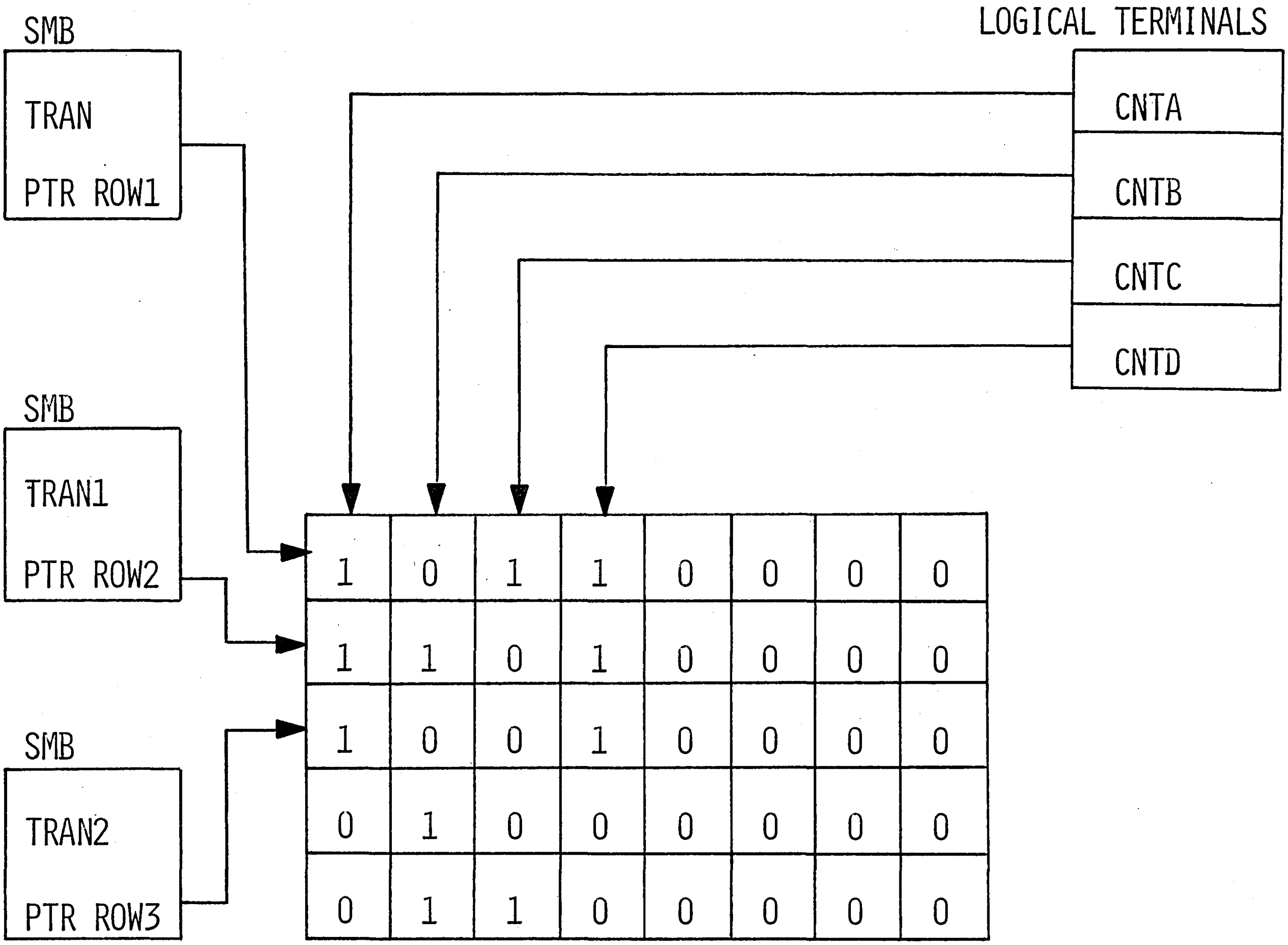
TERMINAL PROFILE

```
)( TERMINAL          DEPT40
     TRANSACT          PAYROLL
     TRANSACT          PERS
)( TERMINAL          DEPT65
     TRANSACT          PAYROLL
)( TERMINAL          VPPERS
     TRANSACT          PAYROLL
)( TERMINAL          DEPT3L
     COMMAND           HOLD
```

TERMINAL SECURITY

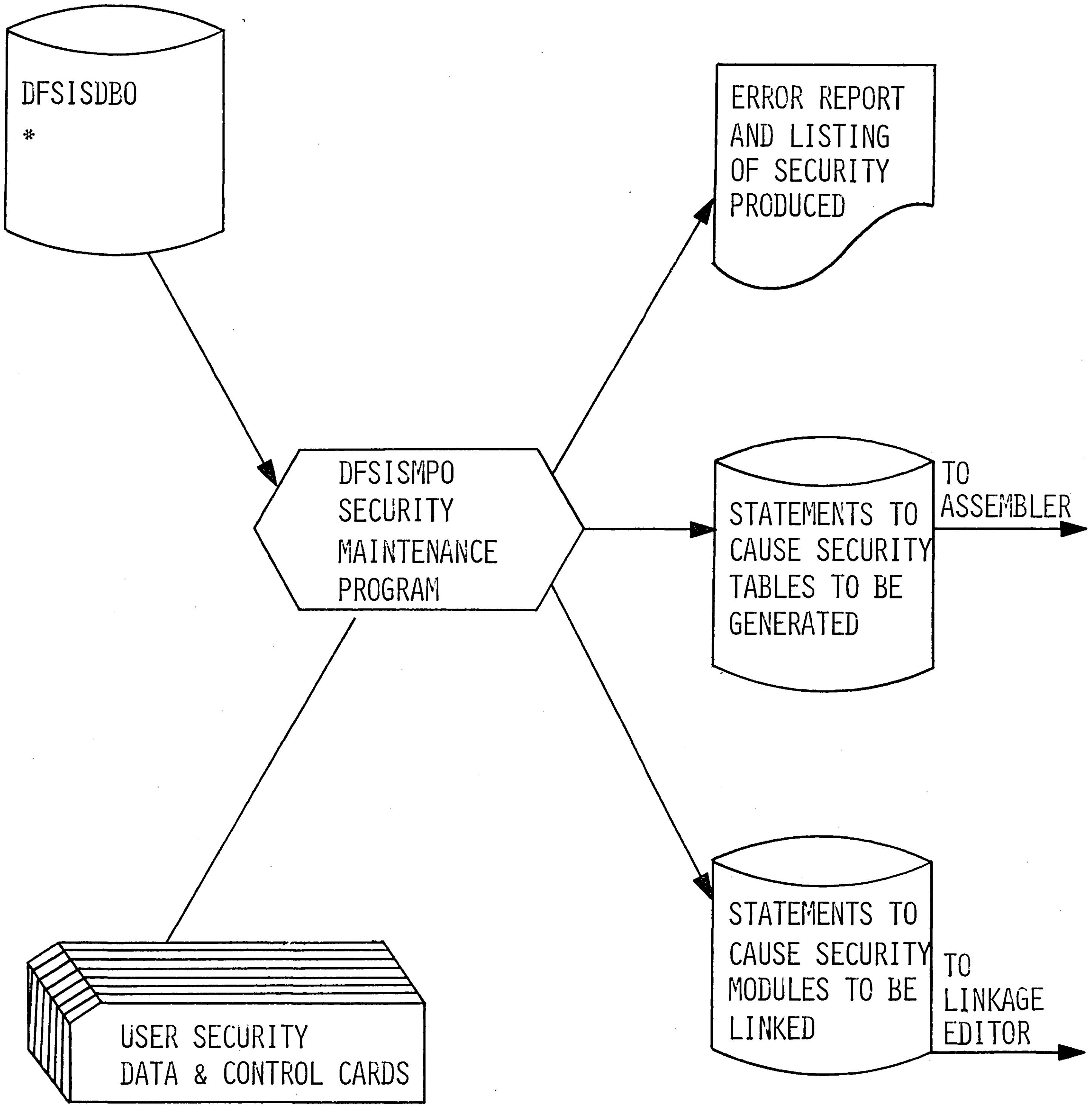
	INVNTORY	PAYROLL	PERS	SAMPLE
DEPT40		X	X	
DEPT65		X		
S274001	X			X
S274002	X			X
S274003	X			X
VPPERS		X		

TERMINAL SECURITY

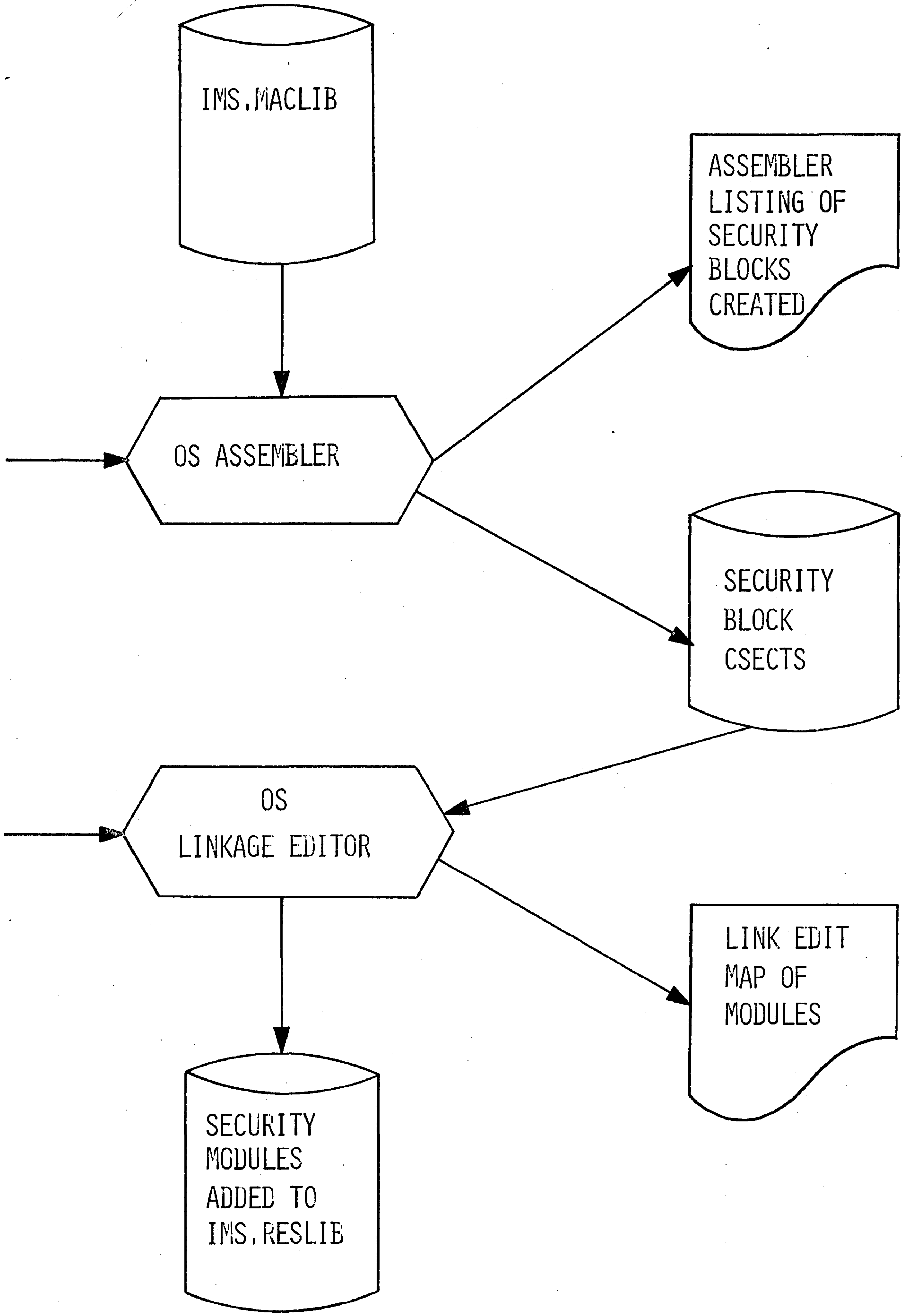


1	0	1	1	0	0	0	0
1	1	0	1	0	0	0	0
1	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0

SECURITY MAINTENANCE -- STEP 1



SECURITY MAINTENANCE -- STEPS 2 AND 3



TERMINAL COMMANDS

(SUGGESTED GROUPINGS)

MASTER TERMINAL ONLY

/ASSIGN

/CHANGE

/CHECKPOINT

/DBDUMP

/DBLOG

/DBNOLOG

/DBRECOVERY

/ERESTART

/NRESTART

/DELETE

/DISPLAY

/PSTOP

/PURGE

/START

/STOP

REMOTE AND MASTER TERMINAL

/BROADCAST

/TEST

/EXCLUSIVE

/END

/LOG

/CANCEL

/LOCK

/IAM

/UNLOCK

/RDISPLAY

/SET

/RESET

COMMAND LANGUAGE FORMAT

```
/VERB      (PASSWORD)  KEYWORD P1 KEYWORD P2, P3, P4. COMMENTS
```

EXAMPLE:

```
/START     (MASXYZ)  LINE 1 PTERM A, B. THIS IS A COMMENT
```

KEYWORDS

LINE

PTERM

LTERM

TRAN OR TRANS

PROG OR PROGRAM OR PGM

DATABASE

PASSWORD OR PSWD

CPRI

LPRI

NPRI

LMCT

REGION OR REG OR MSGREG

```
/CHANGE

/CHANGE PASSWORD ABCD TO PASSWORD EFGH

/CHANGE PASSWORD ABCD TO EFGH


/ASSIGN

/ASSIGN LTERM ABC TO LINE 15 PTERM A

/ASSIGN LTERM ABC TO LINE 2 PTERM A PTERM B

/ASSIGN LTERM ABC TO LINE 3 PTERM A LINE 4 PTERM B

/ASSIGN CPRI 6 TO TRAN ABC

/ASSIGN LPRI 12 TO TRAN XYZ, ABC

/ASSIGN NPRI 10 TO TRAN N24

/ASSIGN LMCT 15 TO TRAN Z31
```

/DELETE

```
/DELETE   PASSWORD  SECURITY FOR TRAN AB1, AB2

/DELETE   PASSWORD  SECURITY FOR LTERM ABC, DEF

/DELETE   PASSWORD  SECURITY FOR LINE 1 PTERM A, B

/DELETE   PASSWORD  SECURITY FOR PROGRAM DFG, RST

/DELETE   PASSWORD  SECURITY FOR DATABASE MN1, MN2


/DELETE   TERMINAL  SECURITY FOR TRAN AB1, AB2
```

```
/DISPLAY
(PASSWORD)       STATUS

                 ACTIVE

*                QUEUE          PRIORITY  { N   }
                                          { ALL }

*                TRAN      { CODE }
                           { ALL  }

*                PGM       { NAME }
                           { ALL  }

*                DATABASE  { NAME }
                           { ALL  }

*                LINE      { NUMBER }
                           { ALL    }

                 LINE NUMBER PTERM     { NUMBER }
                                       { ALL    }

*                LTERM     { NAME }
                           { ALL  }

                 ASSIGNMENT LTERM      { NAME }
                                       { ALL  }

                 ASSIGNMENT LINE NUMBER

                          PTERM        { NUMBER }
                                       { ALL    }

                 MASTER
```

* A SERIES OF PARAMETERS ARE ACCEPTABLE

/PSTOP - /PURGE - /STOP - /START

```
/PSTOP     LINE 1, 2, 3
/START     LINE ALL
/STOP      LINE 1 PTERM A, B
/START     LINE 1 PTERM ALL
/PSTOP     LTERM ABC, DEF
/PURGE     LTERM ALL
/STOP      TRAN AB1, AB2
/START     TRAN ALL
/STOP      PROGRAM GH2, GH3
/START     PROGRAM ALL
/STOP      DATABASE MN1, MN2
/START     REGION
```

NOTE: PROGRAM, DATABASE, REGION - ONLY /START AND /STOP ARE ACCEPTABLE

		LINE &					
	LINE	PTERM	LTERM	TRAN	PROGRAM	DATABASE	REGION
/PSTOP	1, 2, 3	1, 2, 3	2, 3	6, 7	*	*	*
/PURGE	1, 4, 5	1, 4, 5	4, 5	1, 8	*	*	*
/STOP	1, 2, 4	1, 2, 4	2, 4	1, 7	9	10	11

1 - DO NOT QUEUE INPUT
2 - DO NOT SEND OUTPUT
3 - QUEUE OUTPUT
4 - DO NOT QUEUE OUTPUT
5 - SEND OUTPUT
6 - QUEUE INPUT
7 - DO NOT SCHEDULE SMB
8 - SCHEDULE THIS SMB
9 - DO NOT SCHEDULE THIS PROGRAM
10 - DO NOT USE THIS DATA BASE
11 - STOP A MESSAGE REGION

* - NOT ALLOWED

/BROADCAST

```
/BROADCAST TO PTERM ALL
MESSAGE TEXT

/BROADCAST TO ALL
MESSAGE TEXT

/BROADCAST TO LINE ALL
MESSAGE TEXT

/BROADCAST TO LTERM ALL
MESSAGE TEXT

/BROADCAST TO LTERM ABC, DEF
MESSAGE TEXT

/BROADCAST TO ABC, DEF
MESSAGE TEXT

/BROADCAST TO LINE 2
MESSAGE TEXT

/BROADCAST TO LINE 2 PTERM ALL
MESSAGE TEXT

/BROADCAST TO LINE 2, 3
MESSAGE TEXT

/BROADCAST TO LINE 2 PTERM A, B
MESSAGE TEXT
```

```
/TEST

/EXCLUSIVE

/END

/LOG THIS IS A CLASS EXERCISE

/CANCEL
```

/LOCK - /UNLOCK

```
/LOCK      PTERM (PASSWORD)
/LOCK      (PASSWD1) LTERM ABC (PASSWD2), EFG (PASSWD3)
/LOCK      TRAN AB1 (PASSWD4), AB2 (PASSWD5)
/LOCK      PROGRAM GH1 (PASSWD6), GH2 (PASSWD7)
/LOCK      DATABASE MN1 (PASSWD8), MN2 (PASSWD9)
/UNLOCK    DATABASE MN1 (PASSWD8), MN2 (PASSWD9)
```

/IAM

```
/IAM    LTERM P1 (PASSWD)

/IAM    PTERM (PASSWD1)  LTERM P1 (PASSWD3)
```

/RDISPLAY

```
/RDISPLAY  (PASSWD) MASTER
```

COMMANDS

```
/CHECKPOINT

/CHECKPOINT  FREEZE   [ABDUMP]

/CHECKPOINT  DUMPQ    [ABDUMP]

/CHECKPOINT  PURGE    [ABDUMP]

/DBDUMP

/DBDUMP      STOP

/NRESTART

/ERESTART

/DBRECOVERY
```

COMMANDS

```
/CHECKPOINT

/CHECKPOINT  FREEZE   ABDUMP

/CHECKPOINT  DUMPQ

/CHECKPOINT  PURGE

/DBDUMP

/DBDUMP      STOP

/NRESTART    CHKPT 0

/NRESTART    CHKPT 68173/141020

/NRESTART    CHKPT 0 FORMAT ALL

/NRESTART    CHKPT 68165/141050 BLDQ

/NRESTART    CHKPT 68143/11300 BLDQ FORMAT ALL

/ERESTART    CHKPT 68176/105010 SERIAL TAPE50

/ERESTART    CHKPT 68141/091050 BLDQ FORMAT ALL

/DBRECOVERY  DATABASE NAME SERIAL NUMBER, ,NUMBERS
```

SIMPLE CHECKPOINTS

ACTION ORDER	ACTIVITY COUNT	TERMINAL COMMAND	ABNORMAL END
FREE MESSAGE REGIONS	X	X	
LOG BLOCKS AND TABLES	X	X	X
WRITE CHECKPOINT ID TO MASTER TERMINAL	X	X	
WRITE CHECKPOINT ID TO SYSTEM CONSOLE			X
CLOSE LOG			X
RESUME NORMAL PROCESSING	X	X	
CLOSE ALL DATA BASES			X
CLOSE QUEUES			X
WRITE CHECKPOINT ID TO MASTER TERMINAL			X
TERMINATE			X

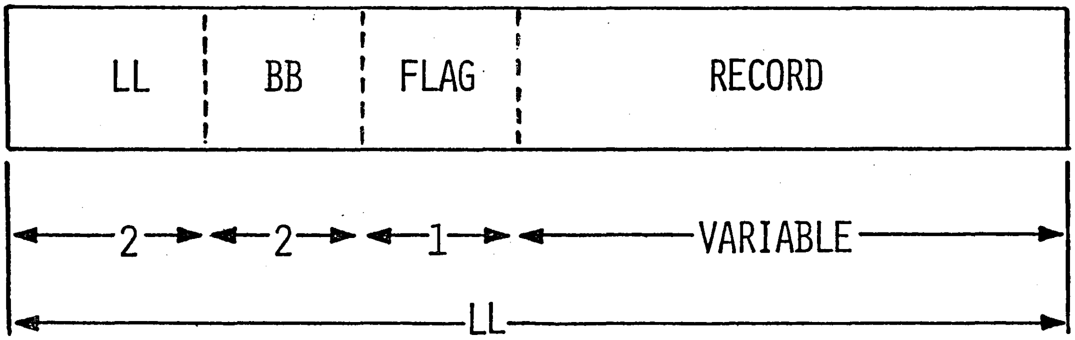
SHUTDOWN CHECKPOINTS

ACTION ORDER	FREEZE	DUMPQ	PURGE
STOP TERMINAL INPUT	X	X	X
STOP TERMINAL OUTPUT	X	X	
PROCESS ALL QUEUED TRANSACTIONS			X
FREE MESSAGE REGIONS	X	X	X
TERMINATE MESSAGE REGIONS	X	X	X
SEND ALL OUTPUT			X
STOP TERMINAL OUTPUT			X
FORCE END OF VOLUME ON LOG TAPE	X	X	X
DUMP QUEUES TO LOG TAPE		X	X
CLOSE QUEUES	X	X	X
CLOSE ALL DATA BASES	X	X	X
LOG BLOCKS AND TABLES	X	X	X
WRITE CHECKPOINT ID TO MASTER TERMINAL	X	X	X
WRITE CHECKPOINT ID TO SYSTEM CONSOLE	X	X	X
CLOSE LOG	X	X	X
TERMINATE	X	X	X

IMS/360 SYSTEM LOG ENTRIES

FOR RESTART	WHEN WRITTEN
● MESSAGE QUEUE CONTROL BLOCKS	WHEN CHANGED
● CHECKPOINT DATA	AT TIME OF CHECKPOINT
● OS/360 DATA SET OPEN OR CLOSE	WHEN DATA SET OPENED OR CLOSED
● CHANGES TO A DATA BASE	INSERT, DELETE, OR REPLACE AGAINST A DATA BASE
FOR RESTART AND STATISTICS	
● MESSAGE FROM TERMINAL	WHEN MESSAGE COMPLETE
● MESSAGE TO TERMINAL OR PROGRAM	WHEN MESSAGE COMPLETE
FOR STATISTICS ONLY	
● ERROR SEGMENTS	WHEN HARDWARE ERROR
● COMPLETION OF SEND RECORD	COMPLETION OF SENDING
● APPLICATION ACCOUNTING RECORD	TERMINATION OF APPL. PROGRAM
● IMS/360 ACCOUNTING RECORD	IMS/360 IS STARTED OR STOPPED

LOG RECORD FORMAT



MESSAGE LOG RECORD

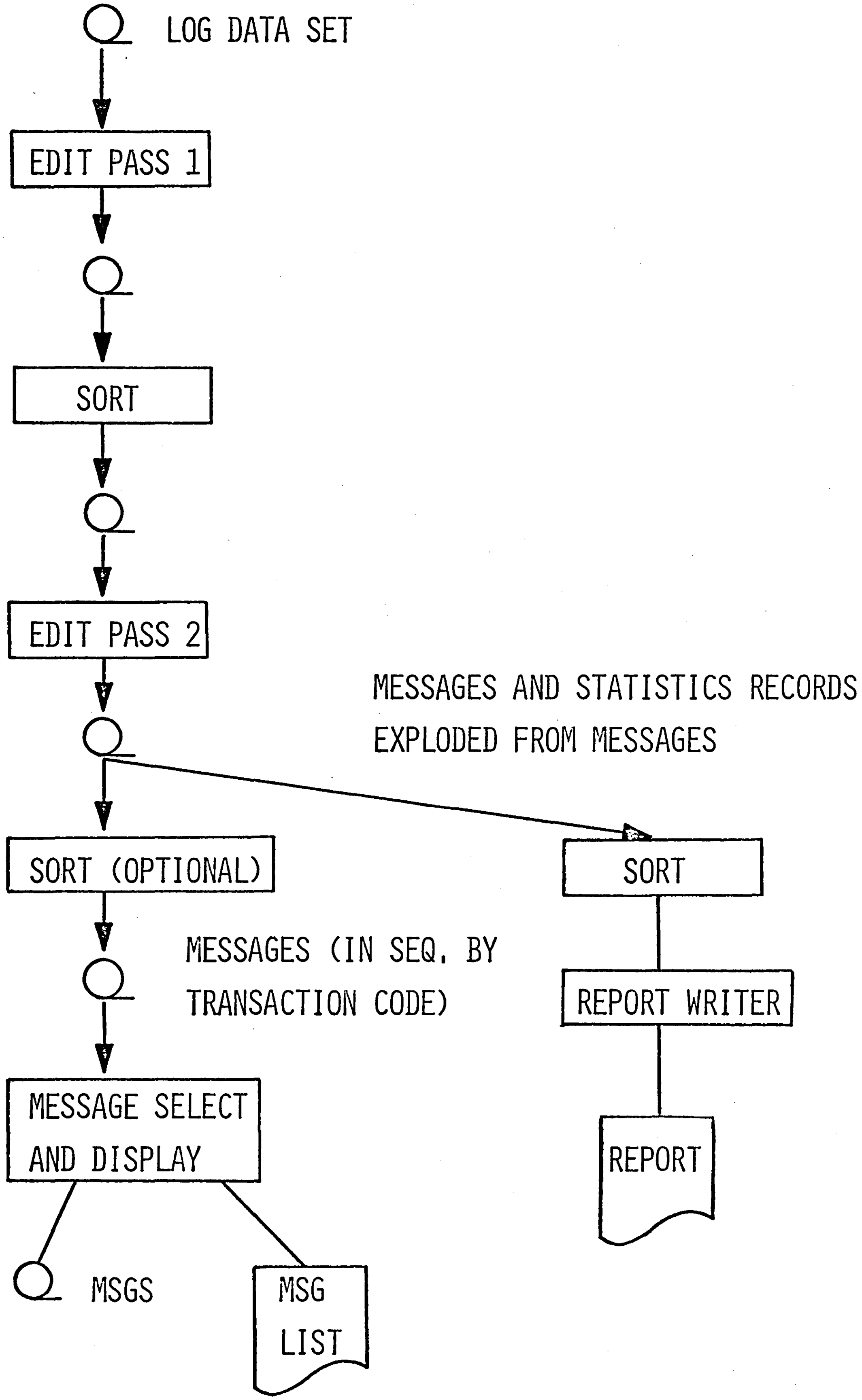
BYTES		
2	RECORD LENGTH	
2	NOT USED	
1	RECORD FLAG	
1	COMMUNICATIONS FLAG	
1	TERMINAL TYPE	
1	LINE NUMBER	
2	TERMINAL ADDRESS	
2	INPUT SEQUENCE NUMBER	
8	INPUT SYMBOLIC NAME	(OUTPUT)
4	DATE	
4	TIME	
	●	
	●	
	●	
	●	
	●	

	RECORD FLAG
INPUT MESSAGES	01
INPUT ERRORS	02
OUTPUT MESSAGES	03
OUTPUT ERRORS	04
STATISTICS	05
ACCOUNTING LOG	06
APPLICATION ACCOUNTING LOG	07

APPLICATION ACCOUNTING LOG RECORD

BYTES		
2	RECORD LENGTH = 108	
2	NOT USED	
1	RECORD CODE = X'07'	
8	PSB NAME	
8	TRANSACTION CODE	
1	PRIORITY	
1	TYPE 00 = MESS 02 = BATCH	
1	PARTITION KEY	
4	ELAPSED TIME	
4	COMPLETION CODE	
8	JOB NAME	
4	NO. OF MESSAGES PROCESSED	DATA BASE
4	GU	
4	GN	
4	GNP	
4	GUH	
4	GNH	
4	GNPH	
4	INSERT	
4	DELETE	
4	REPLACE	
4	MOVE CALLS	
4	GU MESS	MESSAGE
4	GN MESS	
4	INSERT MESS	
4	NOT USED	

PROCESSING THE LOG



TYPES OF STATISTICAL REPORTS

- MESSAGES QUEUED BUT NOT SENT -- BY TERMINAL
- LINE AND TERMINAL LOADING BY TIME OF DAY
- ERROR REPORTS ON BAD TRANSMISSION
- TRANSACTION REPORT
- TRANSACTION RESPONSE REPORT
- APPLICATION ACCOUNTING REPORT
- IMS/360 ACCOUNTING REPORT

```
      MESSAGES -- QUEUED BUT NOT SENT              DATE 05/31/67

TRANSACTION       TOTAL
   CODE           MESSAGES

T19QCA2A                 9
T19QCA2B                19
```

LINE AND TERMINAL REPORT DATE 2/11/69

			TOTAL	TOTAL	AVG,	HOURLY	DISTRIBUTION	
LINE	TERM	R/S	MESSAGES	CHARACTERS	SIZE	00-07	07-08	08-09
001	A0							
T15CASIA		S	12	600	50	1	3	2
		R	14	840	60	1	2	2
002	A0							
T15CAS2A		S	4	320	80	1	0	0
		R	4	80	20	0	0	0
T15CAS2B		S	6	180	30	1	0	1
		R	5	200	40	1	2	1
TRM		S	10	500	50	2	0	2
TOTALS		R	9	280	31	1	2	2
SYSTEM		S	22	1100	50	3	3	4
TOTALS		R	23	1120	48	2	4	4

TRANSACTION REPORT

TRANSACTION		TOTAL	TOTAL	AVG	HOURLY	DISTRIBUTION	
CODE	R/S	MESSAGES	CHARACTERS	SIZE	00-07	07-08	08-09
T21CAS1R	R	5	250	50	0	1	1
T21CAS2S	S	15	1250	83	5	2	3
SYSTEM	S	15	1250	83	5	2	3
TOTALS	R	5	250	50	0	1	1

```
                   TRANSACTION RESPONSE REPORT             DATE 08-31-67

   TYPE       TOTAL      LONGEST   95%       75%       50%       25%       SHORTEST
TRANSACTION   RESPONSES  RESPONSE  RESPONSE  RESPONSE  RESPONSE  RESPONSE  RESPONSE


T231T05M         25     05M 30.0S 05M 00.0S 03M 00.0S 02M 20.0S 01M 00.0S    40.0S
T2359ALL          5         20.0S     15.0S      8.0S      6.0S      4.0S     3.0S
```

```
                    APPLICATION ACCOUNTING REPORT          DATE 01/02/68

                                          DATA BASE COUNTS
PROGRAM TRANSACTION MESSAGE - - - - COUNTS            MOVE    BAD        TOT MESS    AVR
NAME     NAME     PRI  QTY   GU   GN  ISRT   GU   GN    CALL    CC         CPU TIME    TIME


PSB00001 TRANS001 01    71   142  14   71    81   42      1      1         10.65S      0.15S
                  02    81   162  16   81    91   31      1      0         12.15S      0.15S
                  **   152   304  30  152   172   73      2      1         22.80S      0.15S
    SYSTEM TOTAL       152   304  30  152   172   73      2      1
```

```
                    IMS ACCOUNTING REPORT               DATE 1-22-68



IMS  CPU TIME FOR DAY 1/20/68 IS 07H 47M 07.9S OR 28,027.9S

IMS  CPU TIME FOR DAY 1/21/68 IS 06H 30M 29.5S OR 23,429.5S

IMS  CPU TIME FOR DAY 1/22/68 IS 07H 40M 39.5S OR 27,639.5S

IMS  CPU TOTAL TIME           IS 21H 58M 16.9S OR 28,096.9S
```

CONTROL CARDS

```
TRANS  CODE=(TRANSCOD,I,O), (TRANSA,I)

TRANS  CODE=(TRANSA,I)

TRANS  CODE=(INV*,,O)

TRANS  CODE=(ALL,I,O)

SYM NAME=(TERMA,I,O)

SYM NAME=(TERMPAY,I,O,TERM)

TERM  ADDR=(3,A,I,O), (42,C,,O,21,A)

TERM  ADDR=(I,ALL,I,O)

TIME=(68014,1620,68015,1900)

NON PRINT=HEX
```

SORTING THE LOG TAPE

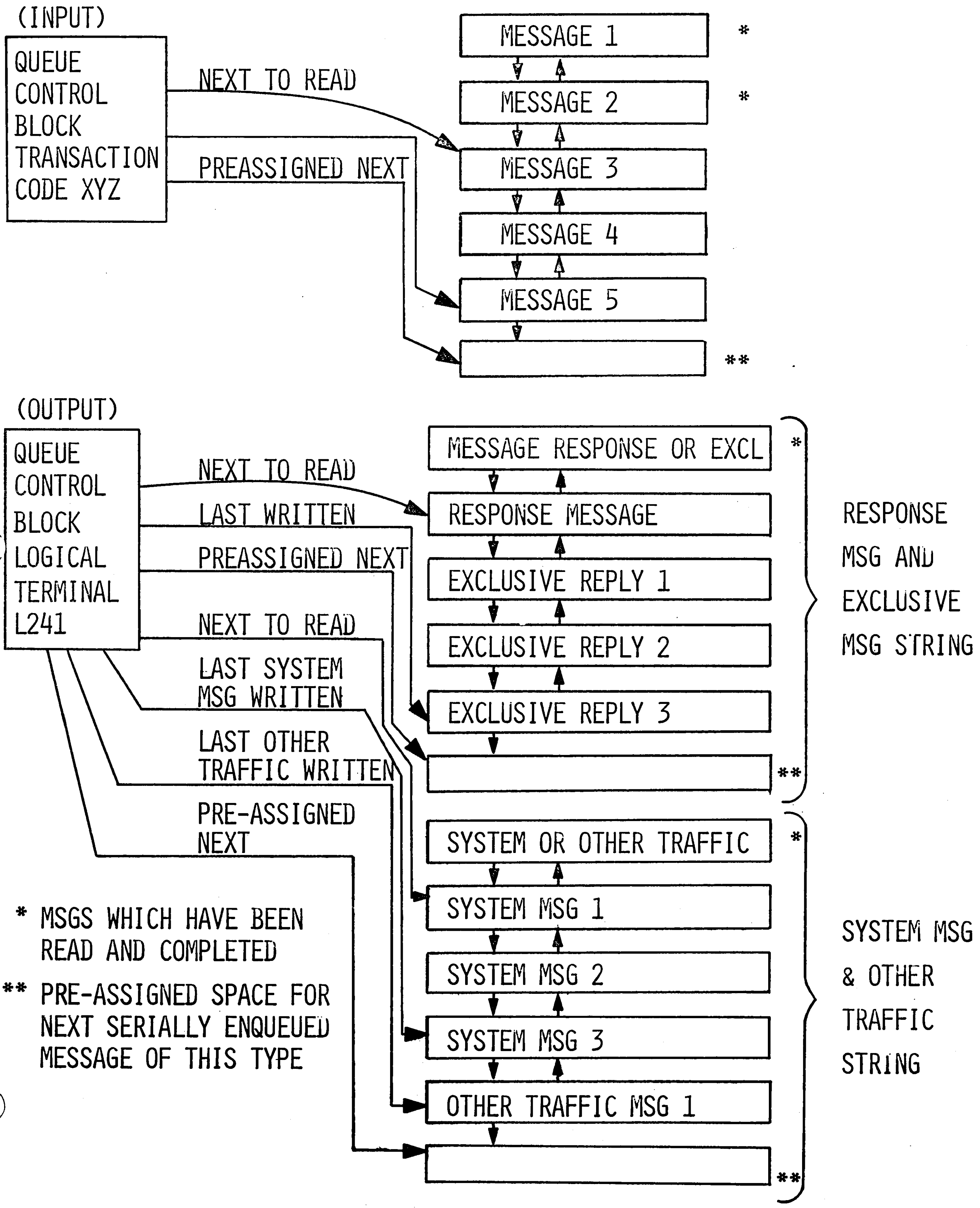
BY DATE

```
SORT FIELD=(5,1,CH,A,9,4,PD,A,13,36,CH,A)
```

WITHOUT DATE

```
SORT FIELD=(5,1,CH,A,13,36,CH,A)
```

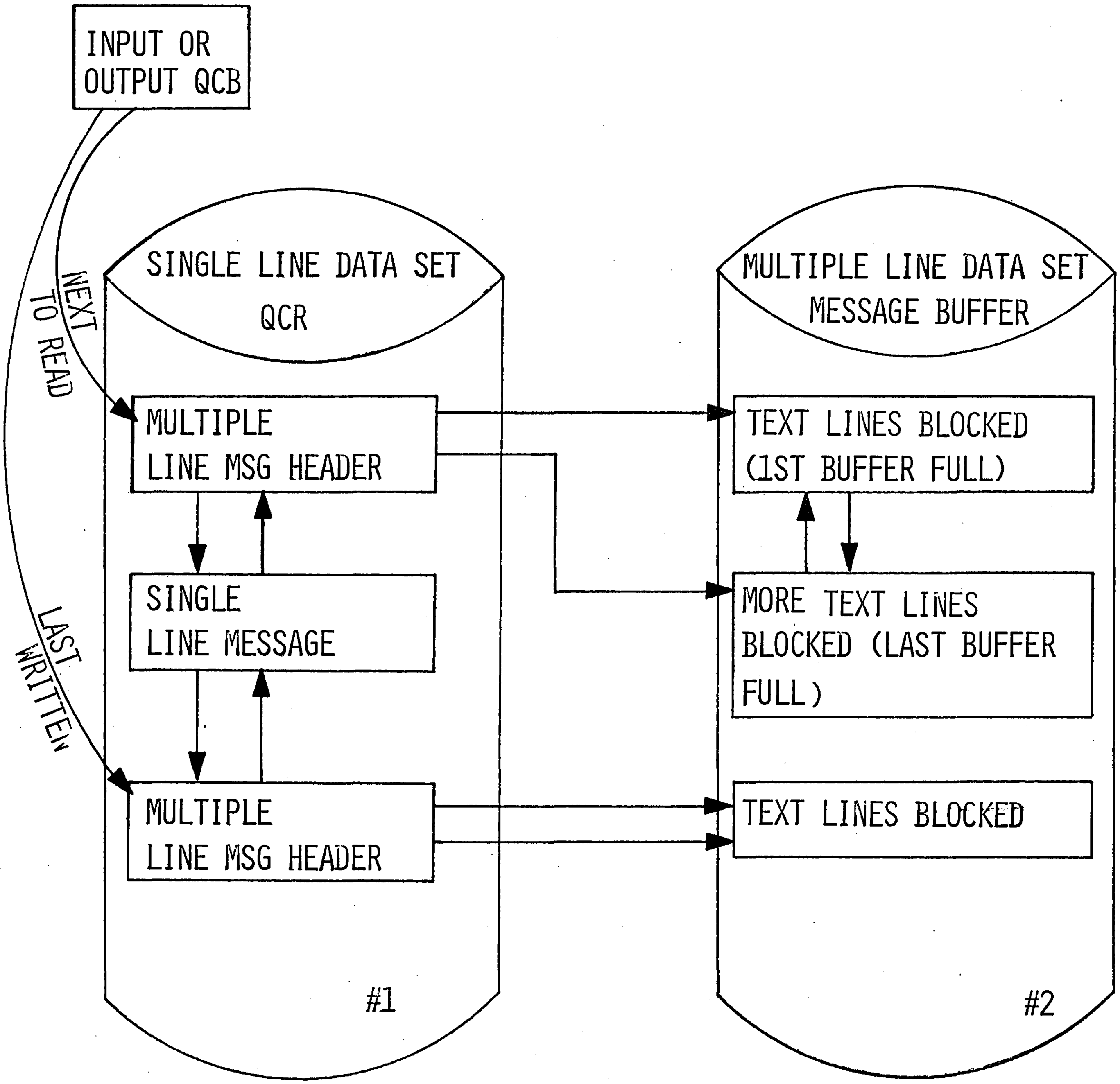
MESSAGE QUEUES



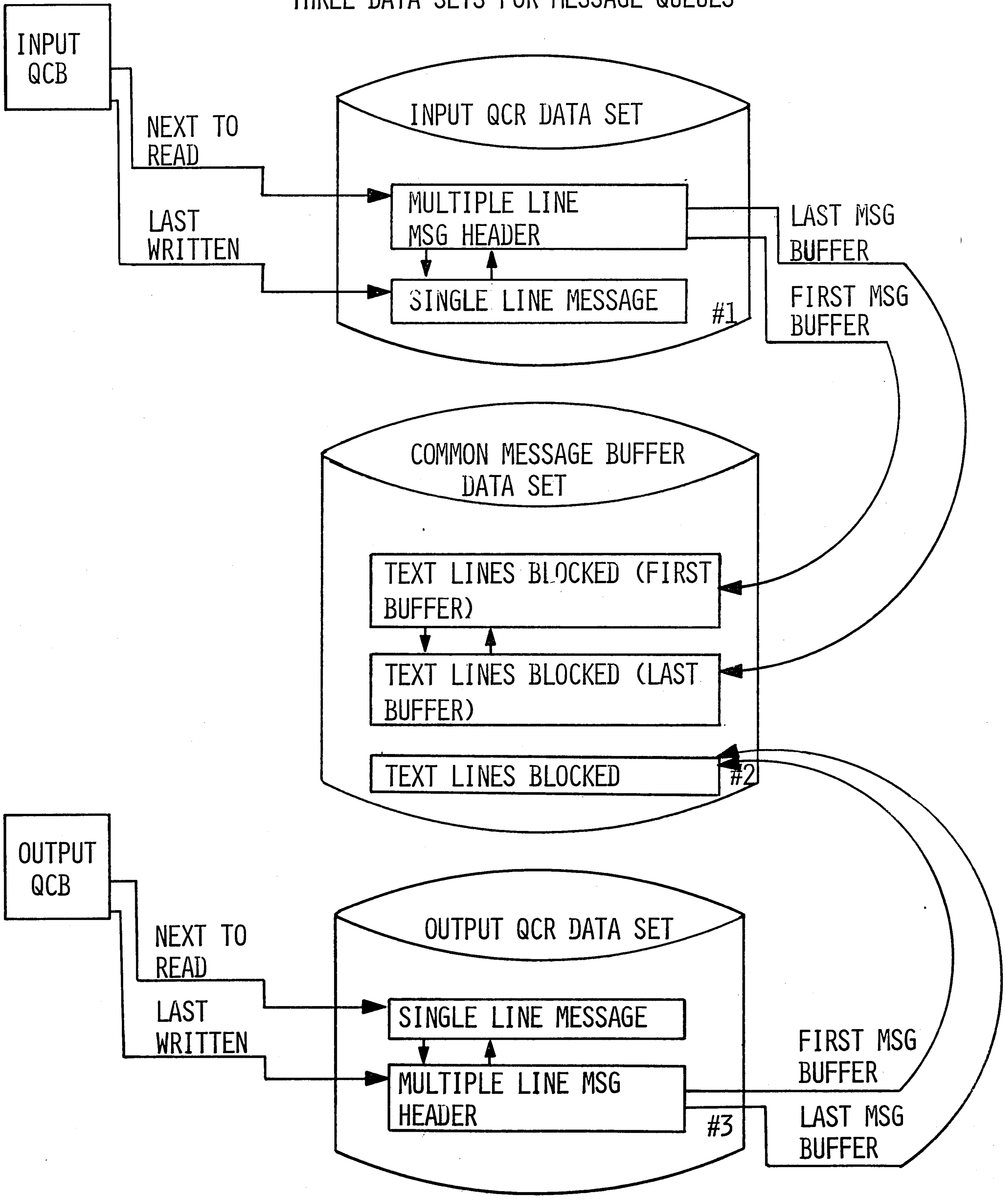
* MSGS WHICH HAVE BEEN READ AND COMPLETED

** PRE-ASSIGNED SPACE FOR NEXT SERIALLY ENQUEUED MESSAGE OF THIS TYPE

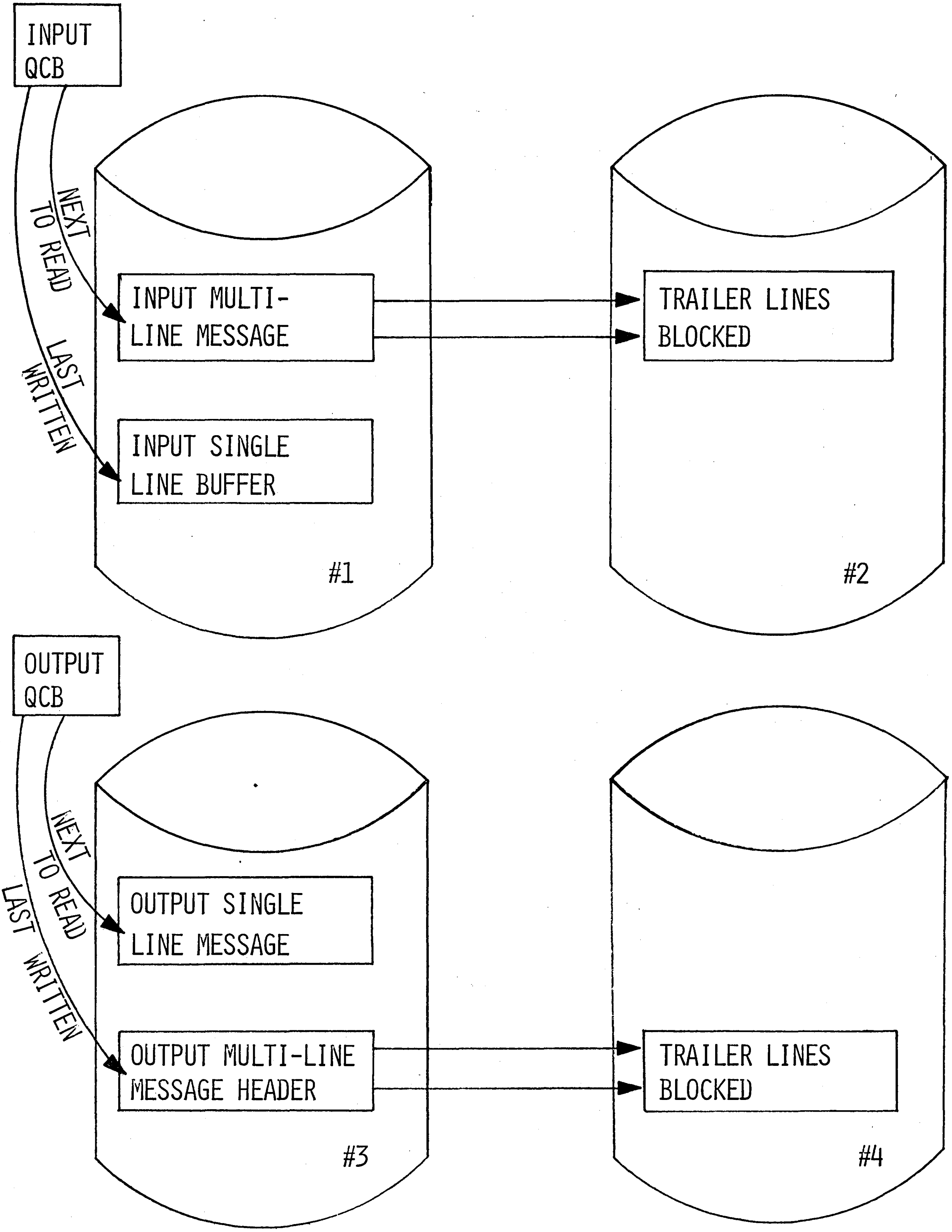
TWO DATA SETS FOR MESSAGE QUEUES



THREE DATA SETS FOR MESSAGE QUEUES



FOUR DATA SETS FOR MESSAGE QUEUES



MESSAGE QUEUE SPACE ALLOCATION
(NO REUSE ALLOWED)

4 MESSAGE QUEUE DATA SETS

5 MESSAGE LINES PER BLOCK IN MSG BUFFER D/S
TEXT + PREFIX $\cong$ 200 CHARACTERS
SINGLE PREFIX PER PHYSICAL RECORD

50,000 INPUT MSGS/DAY (12 HOURS)

50,000 OUTPUT

10,000 OF INPUT MSGS ARE MULTIPLE LINE (AVERAGE 5 LINES)
25,000 OF OUTPUT MSGS ARE MULTIPLE LINE (AVERAGE 10 LINES)
9 MULTIPLE LINE OSAM (MSG BUFFER) RCDS PER 2314 TRACK
20 SINGLE LINE OSAM (QCR) RCDS PER 2314 TRACK

INPUT QCR $= \dfrac{50{,}000}{20} =$ 2500 TRACKS OR 125 CYL

OUTPUT QCR $= \dfrac{50{,}000}{20} =$ 2500 TRACKS OR 125 CYL

INPUT MSG BUFFER $= \dfrac{10{,}000}{9} =$ 1112 TRACKS OR 56 CYL

OUTPUT MSG BUFFER $= \dfrac{25{,}000 \times 2}{9} =$ 5,556 TRACKS OR 278 CYL

TOTAL = 584 CYL

MESSAGE QUEUE SPACE ALLOCATION
(REUSE ALLOWED)

200 TRANSACTION TYPES
200 LOGICAL TERMINALS

AVERAGE HOURLY MESSAGE TURN-OVER

$$\frac{50{,}000}{12 \times 200} = 21 \text{ MESSAGES PROCESSED PER QCB PER HOUR}$$

TO RUN 3 HOURS BEFORE REUSE ALLOCATE:

INPUT QCR $\frac{3x21x200}{20} = \frac{12600}{20}$ = 630 TRACKS OR 32 CYL

OUTPUT QCR $\frac{3x21x200}{20} = \frac{12600}{20}$ = 630 TRACKS OR 32 CYL

INPUT MSG $\frac{12600}{5x9} = \frac{3150}{9}$ = 350 TRACKS OR 18 CYL

OUTPUT MSG $\frac{12600}{9}$ = 1400 TRACKS OR 70 CYL

TOTAL 152 CYL

DISK I/O OPERATIONS - WRITING

1ST 3 HOURS (NO REUSE)

INPUT QCR	12600 (WRITE)
OUTPUT QCR	12600 (WRITE)
INPUT MESSAGE	3150 (WRITE)
OUTPUT MESSAGE	12600 (WRITE)
TOTAL	40950

2ND 3 HOURS (REUSE)

			(BUILD MSG STRINGS)	
	WRITE	READ	WRITE	READ
INPUT QCR	12600	12600		
OUTPUT QCR	12600	12600		
INPUT MESSAGE	3150	3150	3150	3150
OUTPUT MESSAGE	12600	12600	6300	6300
SUBTOTAL	40950	40950	9450	9450

AVERAGE QCR STRING OBTAINED = $\frac{3x21+6x21}{2}$ -2 = 94-2=92

NUMBER OF QCR SEARCHES = $\frac{3x\ 12600}{92}$ = 411

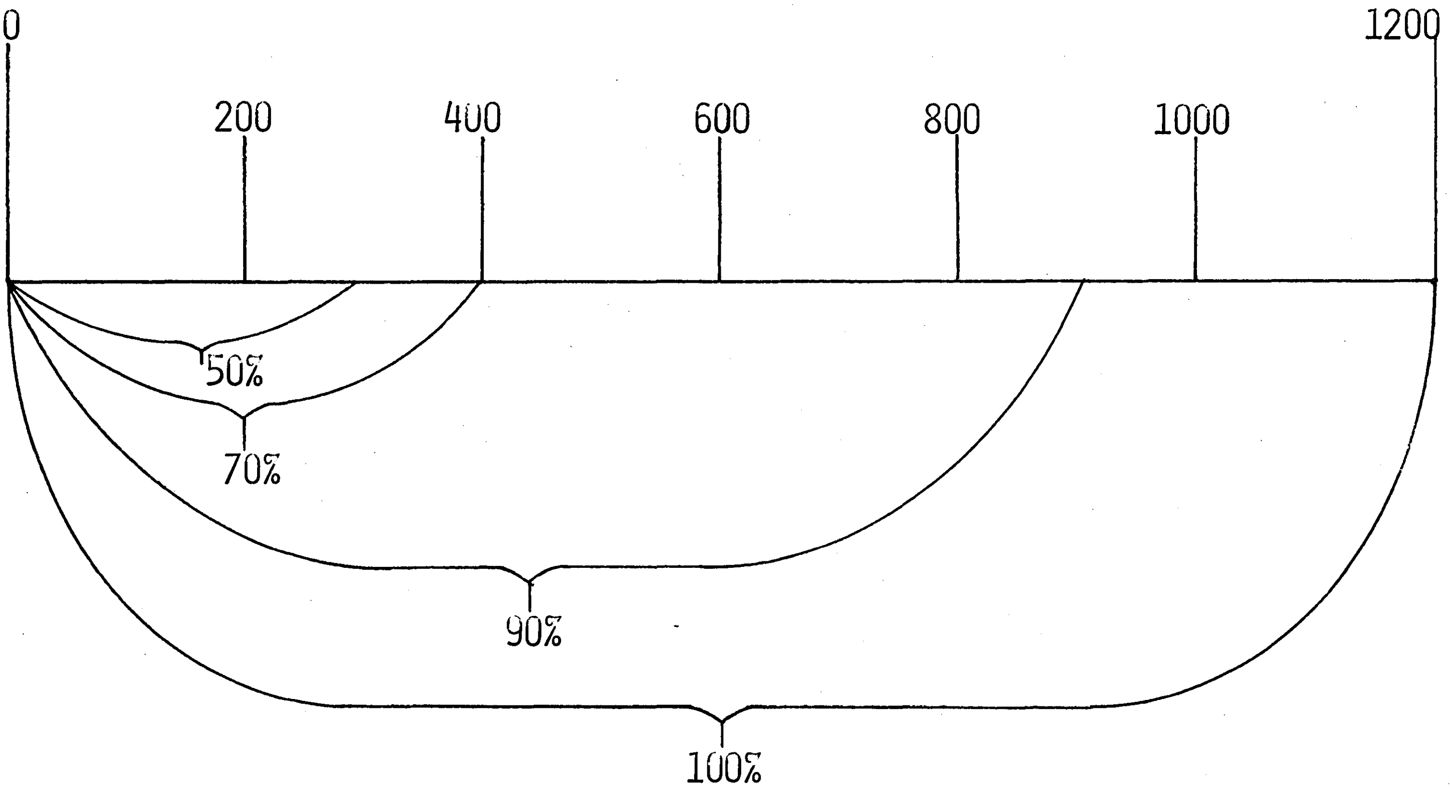
411 x 2 READS = 822

411 x 1 WRITE = 411

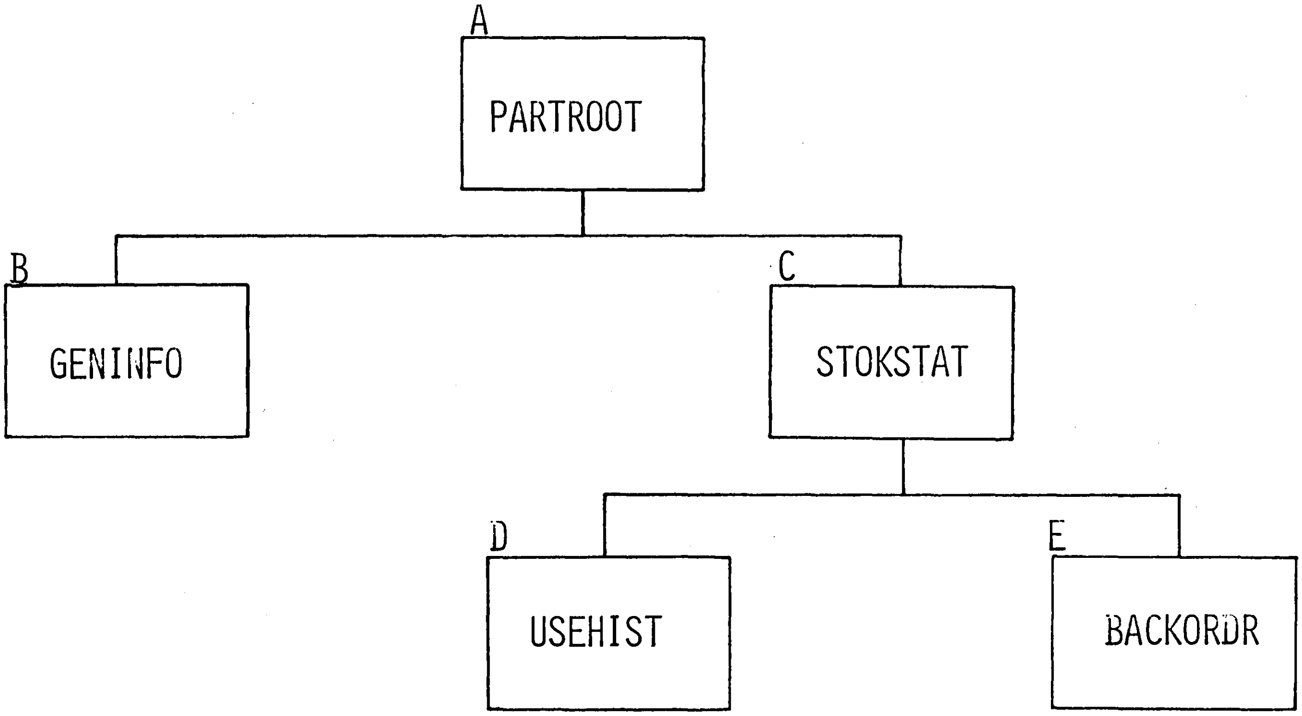
SUBTOTAL 1233

TOTAL = 1233 + 40950 + 40950 + 9450 + 9450 = 102,033

RECORD LENGTH DISTRIBUTION



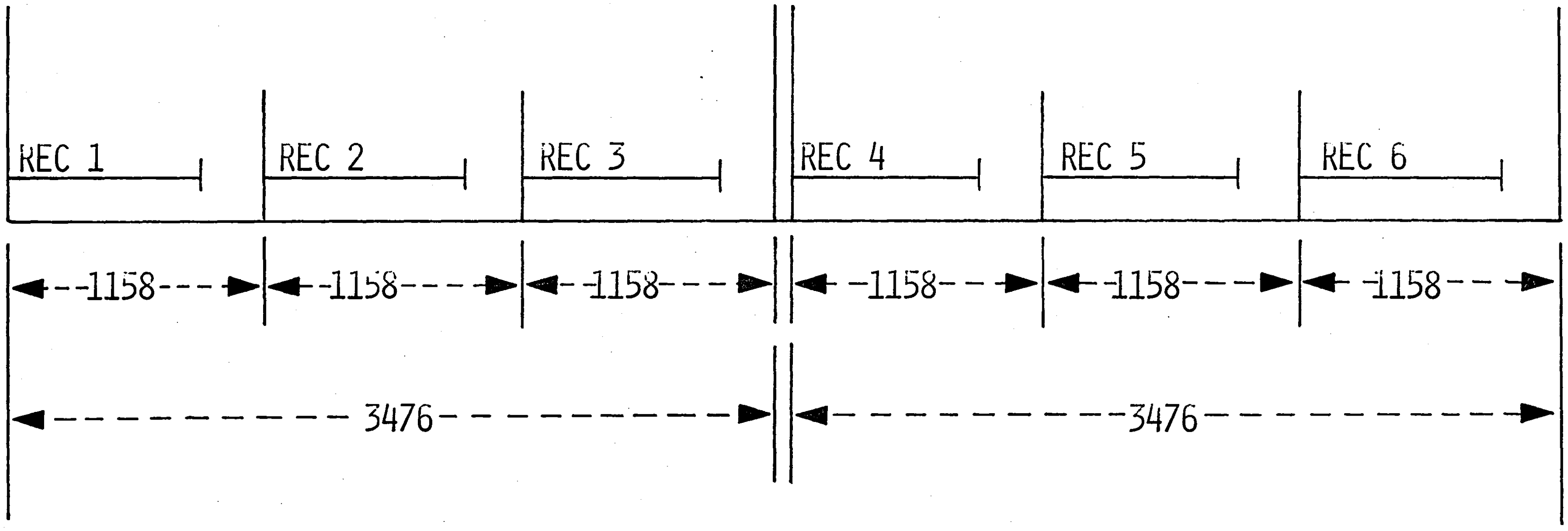
DATA BASE RECORD



SEGMENT NAME	OCCURRENCE PER DATA BASE RECORD	SEGMENT LENGTH INCLUDING 2 BYTES FOR CONTROL
PARTROOT	1	20
GENINFO	4 (90% HAVE 4 OR LESS)	8
STOKSTAT	10 (90% HAVE 10 OR LESS)	15
USEHIST	2	5
BACKORDR	3 (90% HAVE 3 OR LESS)	20

$$\text{PRIME LRECL} = \overset{A}{20} + \overset{B}{(4x8)} + \overset{C}{10(15} + \overset{D}{(2x5)} + \overset{E}{3(20)}\ ) + 7 = 909$$

CALCULATED LRECL RANGE	LRECL SIZE USED BY IMS/360	LRECL'S PER BLOCK
1739 - 3476	3476	1
1159 - 1738	1738	2
870 - 1158	1158	3
696 - 869	869	4
580 - 695	695	5



BASIC STORAGE REQUIREMENTS

	BYTES
IMS RESIDENT NUCLEUS (LESS CONTROL BLOCKS)	60,000
DATA LANGUAGE/I ACTION MODULES	12,800
OSAM ACCESS METHOD MODULES	2,900
ISAM AND IMS ISAM SIMULATOR MODULES	7,800
BTAM (LESS DEVICE SUPPORT MODULES)	6,600
	90,100

BTAM DEVICE SUPPORT

REQUIREMENTS FOR EACH LINE GROUP DESCRIBED AT IMS/360 SYSTEM DEFINITION TIME.

1050 NON-SWITCHED WITHOUT AUTOPOLL	224
1050 NON-SWITCHED WITH AUTOPOLL	234
1050 SWITCHED	328
2740 WITH DIAL, TRANSMIT CONTROL, AND CHECKING	304
2740 WITH STATION CONTROL AND CHECKING	204
2740 WITH STATION CONTROL, CHECKING, AND AUTOPOLL	224

CONTROL BLOCKS STORAGE REQUIREMENTS

	BYTES
EACH COMMUNICATION LINE AND BUFFER	500
EACH COMMUNICATION TERMINAL	75
EACH TRANSACTION TYPE	60
EACH MESSAGE PROGRAM	500
EACH OPEN DATA BASE	1,000

ASSUMPTIONS

MFT-II SYSTEM

12	2740 LINES
4	1050 LINES
5	OPEN DATA BASES
16	TOTAL DATA BASES
2	MESSAGE REGIONS CONCURRENT
10	PSB's RESIDENT
	LARGEST DATA BASE BUFFER IS 3,000 BYTES
30	PROGRAMS
50	TRANSACTION TYPES
3	MAX REGIONS
4	MAX I/O

ESTIMATE OF STORAGE REQUIREMENTS

BASIC STORAGE REQUIREMENTS (60,000 + 30,100)		90,100
2740 STATION CONTROL, AUTOPOLL AND CHECKING		224
1050 NON-SWITCHED WITH AUTOPOLL		192
IMS/360 CONTROL BLOCKS		18,000
QCR BUFFER POOL (16 X 176)		2,816
MESSAGE BUFFER POOL (5 X 880)		4,400
1050 AND 2740 OUTPUT BUFFERS (5 X 500)		2,500
DATA BASE BUFFERS 2 (5 X 3000)		30,000
PSB POOL (10 X 1,000)		10,000
DMB POOL (2 X 1,000) + (3 X 500)		3,500
	TOTAL	161,732

CLASS EXERCISES

Write the necessary IMS/360 Definition Cards to define a system to operate in the following environment:

1. MVT System

2. Three message processing regions

3. Five transaction codes (TRANS01, through TRANS05). Each invokes a different program (STUDNT01 through STUDNT05 respectively). All use the same data base (DI21PART). Priorities and processing limits should be selected by the student.

4. The teleprocessing system consists of three nonswitched lines, each with a 2740, and one switched line to accommodate a 2740. Line addresses are 032 through 035. All terminal addresses are E2.

5. There is to be one pool-subpool combination with three logical terminals (LTERM1, LTERM2, and LTERM3).

6. The student is to supply all other necessary information to generate a system.

JCL will be provided by the instructor.

Section 5
Instructor Materials

Instructor Materials

INSTRUCTORS' NOTE:

The information contained in this section is for use by the instructor in preparing for a hands-on class and in assigning the class problems.

Generating A System

An IMS/360 system must be generated for use in running class problems. The following is a list of considerations in generating the system.

1. If the system is to be used for teleprocessing by the class, transaction codes must be included in system definition.
2. Catalog the procedures to be used by the class.
3. Allocate adequate space for the DBD, PSB, and Program Libraries to be used by the class.
4. If teleprocessing is to be used, the sample problem which accompanies the IMS/360 system should be run to ensure proper system generation. This will entail generating the data base to be used by the class along with the Data Base Description.
5. If a teleprocessing system is not generated, the sample problem data base description and data base must still be generated for use by the class.
6. If the programs provided in this section of the Education Guide are to be used in the class, they should be punched and run against the established sample problem data base. It will be necessary to generate the PSB's to be used with the program. They will be the same as the one on page 5.28 with the appropriate language specified.

Assigning Class Problems

In order to allow the student experience in writing data base descriptions, program specification blocks, programs, and system definitions, it is suggested that prepunched JCL be given each student or student group. This avoids JCL errors and helps eliminate the bottleneck at the key punch. A list of the suggested JCL can be found on page 5.6.

The length of the courses taught from this Education Guide does not allow a student to write a complete program. For this reason, it is suggested that the student be given one of the programs which start on page 5.7. He can then be required to modify the program to perform a specific function. It is important to require specific functions of the student. If he is simply given the program and told to modify it as he wishes, it is difficult to monitor his progress.

The COBOL, PL/I, and Assembler language program presented in Appendix B are written to run with the sample problem data base provided with the IMS/360 system. A write-up of this problem can be found in the IMS/360 Systems Operation Manual. A complete list of the part numbers in the data base can be found starting on page 5.31.

The programs beginning on page 5.6 should only be used as teaching aids, they should not be used to represent the standard means of writing IMS/360 programs. Each of these programs reads a card which has the key of the root segment (PARTROOT) punched in it, retrieves the PARTROOT and STANINFO segment from the data base, prints out the information contained in the segments, and reads another card.

Page 5.20 begins a message processing program which is written in COBOL and a message input/output simulator. The message processing program does the same functions as the COBOL program on page 5.7 but the input now appears to be coming from a terminal and the output is directed to a terminal. This technique is presented here to allow message processing programs to be written and checked out without any teleprocessing equipment. Only COBOL is supplied here. A similar technique could be developed for PL/I or Assembler. Once a message processing program is checked out with the simulator, the only change required is to call Data Language/I instead of the message simulator. Other examples of message processing programs can be found in the Systems Operation Manual.

Machine Time Required

It is suggested that the instructor run each class problem which he plans to assign to the student. Machine time requirements can vary widely depending on the equipment available and the mode of operation.

As a guide line, 2 minutes of 360/65 time should be allowed for any COBOL, PL/I, or Assembler jobs to be compiled, link edited and executed. Two minutes should also be allowed for DBD or PSB generation. System definition runs will require around 5 minutes of 360/65 time.

Message Simulator

The message simulator will simulate input from a terminal by reading cards and formatting them to look like they came from a terminal. Input may be single line or multiple line; however, the length of the line is limited to that which can be contained in an 80 column card.

Output to a terminal is also simulated. This is accomplished by writing to the system output device (SYSOUT).

The format of the calls to the message simulator from the program which is being checked out are the same as the calls which would be made to Data Language/I for input or output to a terminal. The only difference is that the message simulator is called instead of Data Language/I.

The format of the input card is a character count in 1-4, an H for header or B for body in 5, the terminal name in 6-13, followed by the text.

The character count in 1-4 should include an additional count of 4 for the four characters which normally precede an input or output message.

The output which follows was produced from the message simulator which is included in this section. The first line of output shows that the message type is IH for input header. Message count is 30. The terminal name is TERM0001. The call which was issued was a Get Unique.

The second line of output is the actual text of the message. The third line shows an insert call to send a message to a terminal.

```
   MESSAGE TYPE = IH, MESSAGE COUNT = 0030, TERMINAL = TERM0001   GU
PART     023008027
   MESSAGE TYPE = O , MESSAGE COUNT = 0068, TERMINAL = TERM0001  ISRT
 PART=023008027          DESC=CARD FRONT             PROC CODE=46
   MESSAGE TYPE = O , MESSAGE COUNT = 0080, TERMINAL = TERM0001  ISRT
 INV CODE=A  MAKE DEPT=72-46  PLAN REV NUM=    MAKE TIME= 26  COMM
```

Assignment of Student Numbers

Each student should be assigned a number to be used as a suffix for all computer runs. This will help identify output and make any library entries unique.

For Data Base Descriptions:

```
//DBDGEN01  JOB  01,STUDENT01,MSGLEVEL=1
//          EXEC DBDGEN,MBR=DBASE01
```

The underlined number is the student number. JCL for each student will be identical except for this number.

For Program Specification Block Generation:

```
//PSBGEN01  JOB  01,STUDENT01,MSGLEVEL=1
//          EXEC PSBGEN,MBR=STUDENT01
```

The same conventions should be used for compiling, link editing, and executing the application program.

SUGGESTED JCL FOR STUDENTS

DATA BASE DESCRIPTION

```
//DBDGEN01     JOB    01,STUDENT01,MSGLEVEL=01
//             EXEC   DBDGEN,MBR=DBASE01
//C.SYSIN      DD    *
/*
```

PROGRAM SPECIFICATION BLOCK

```
//PSBGEN01     JOB    01,STUDENT01,MSGLEVEL=01
//             EXEC   PSBGEN,MBR=STUDNT01
//C.SYSIN      DD    *
/*
```

BATCH COMPILE AND GO

```
//BATCH01      JOB    01,STUDENT01,MSGLEVEL=01
//JOBLIB       DD     DSN=ICS.LOAD,DISP=SHR
//             DD     DSN=CLA.PGMLIB,DISP=SHR
//             EXEC   IMSCOBGO,MBR=STUDNT01
//C.SYSIN      DD     *
/*
```

COMPILE AND LINK

```
//MSGPR01      JOB    01,STUDENT01,MSGLEVEL=01
//             EXEC   IMSCOBOL,MBR=STUDNT01
//C.SYSIN      DD    *
/*
```

IMS/360 SYSTEM DEFINITION

```
//IMSDEF01     JOB    01,STUDENT01,MSGLEVEL=01
//             EXEC   ASMFC,REGION=96K
//ASM.SYSLIB DD       DSN=ICS.MACLIB,DISP=SHR
//ASM.SYSPRINT  DD    SYSOUT=3
//ASM.SYSPUNCH  DD    SYSOUT=3
//ASM.SYSIN  DD       *
/*
```

ALTERNATIVE PROCEDURES

```
//             EXEC   DLIBATCH,PSB=STUDNT01
//             EXEC     IMSPLI,MBR=STUDNT01
//             EXEC   IMSPLIGO,MBR=STUDNT01
//             EXEC   IMSASSEM,MBR=STUDNT01
```

BATCH PROCESSING -- COBOL EXAMPLE

```
//BATCH01   JOB   848,CABANISS,MSGLEVEL=1,MSGCLASS=3,CLASS=A,PRTY=8
//JOBLIB DD   DSN=CLA.PGMLIB,DISP=SHR
//         DD   DSN=ICS.LOAD,DISP=SHR
//         EXEC   IMSCOBGO,MBR=STUDNT01
//C.SYSIN    DD    *
001010 IDENTIFICATION DIVISION.
001020   PROGRAM-ID.  'IMSTEST'.
001030 ENVIRONMENT DIVISION.
001040 INPUT-OUTPUT SECTION.
001050 FILE-CONTROL.
001060     SELECT MESSAGE-FILE  ASSIGN TO 'TESTIN'  UTILITY.
001070     SELECT TEST-OUTPUT-FILE  ASSIGN  TO 'TESTOUT' UTILITY.
001080 DATA DIVISION.
001090 FILE SECTION.
001100 FD  MESSAGE-FILE
001110     RECORDING MODE IS F
001120     DATA RECORD IS INPUT-MESSAGE.
001130     01  INPUT-MESSAGE             PICTURE IS X(80).
001140 FD  TEST-OUTPUT-FILE
001150     BLOCK CONTAINS 10 RECORDS
001160     DATA RECORD IS PRINT-LINE.
001170     01  PRINT-LINE                PICTURE IS X(133).
001490     01  TEST-OUTPUT-TEXT.
001500         02 TEST-OUTPUT-CHAR  OCCURS 130 TIMES
001510                                            PICTURE X.
01013  WORKING-STORAGE SECTION.
01014      77  GET-UNIQUE  PICTURE XXXX VALUE 'GU  '.
01015      77  GET-NEXT    PICTURE XXXX VALUE 'GN  '.
01016      77  INSERT      PICTURE XXXX VALUE 'ISRT'.
01017      77  SEG-NOT-FOUND PICTURE XX VALUE 'GE'.
01020      77  ERROR-SWITCH  PICTURE X   VALUE ' '.
02016      01  LINE-OUTPUT.
               02 BLANK-SPACE  PICTURE X VALUE ' '.
02019          02  OUTPUT-TEXT  PICTURE X(132) VALUE SPACES.
03001      01  PART-NOT-FOUND-MSG.
03002          02  FILLER PICTURE X(12) VALUE 'PART NUMBER '.
               02 FILL-PART-1 PICTURE X(17).
03004          02  FILLER PICTURE X(16) VALUE ' NOT ON DATABASE'.
         01  MESSAGE-IN-WORK-AREA.
               02  CARD-INPUT.
                 03  ROOT-KEY   PICTURE  X(17).
03006      01  FIRST-LINE.
03007          02  FILLER PICTURE X(5) VALUE 'PART='.
               02 FILL-PART-2 PICTURE X(17).
03009          02  FILLER PICTURE X(7) VALUE '  DESC='.
03010          02  FILL-DESCR PICTURE X(20).
03011          02  FILLER PICTURE X(12) VALUE '  PROC CODE='.
03012          02  FILL-PROC-CODE PICTURE XX VALUE SPACES.
03014      01  SECOND-LINE.
03015          02  FILLER  PICTURE X(9) VALUE 'INV CODE='.
03016          02  FILL-INV-CODE PICTURE X VALUE SPACE.
03017          02  FILLER  PICTURE X(12) VALUE '  MAKE DEPT='.
```

```
03018          02  FILL-MAKE-1 PICTURE XX VALUE SPACES.
03019          02  FILLER  PICTURE X  VALUE '-'.
03020          02  FILL-MAKE-2 PICTURE XX VALUE SPACES.
04001          02  FILLER  PICTURE X(15) VALUE '  PLAN REV NUM='.
04002          02  FILL-PLAN-REV-NUM PICTURE XX VALUE SPACES.
04003          02  FILLER  PICTURE X(12) VALUE '  MAKE TIME='.
04004          02  FILL-MAKE-TIME  PICTURE ZZZ.
04005          02  FILLER  PICTURE X(12) VALUE '  COMM CODE='.
04006          02  FILL-COMM-CODE PICTURE XXXX VALUE SPACES.
04012      01  ROOT-FORMAT.
04013          02  FILLER  PICTURE X(26).
04014          02  DESCRIPTION PICTURE X(20).
04015          02  FILLER  PICTURE XXXX.
           01  STANINFO-FORMAT.
05002          02  FILLER  PICTURE X(18).
05003          02  PROCUREMENT-CODE  PICTURE XX.
05004          02  INVENTORY-CODE  PICTURE X.
05005          02  PLANNING-REVISION-NUMBER  PICTURE XX.
05006          02  FILLER  PICTURE X(24).
05007          02  MAKE-DEPT  PICTURE XX.
05008          02  MAKE-COST-CTR  PICTURE XX.
05009          02  FILLER  PICTURE XX.
05010          02  COMMODITY-CODE  PICTURE XXXX.
05011          02  FILLER  PICTURE XXXX.
05012          02  MAKE-SPAN  PICTURE S999.
05013          02  FILLER  PICTURE X(21).
05014      01  ROOT-SSA.
               02  FILLER  PICTURE X(19) VALUE 'PARTROOT(PARTKEY  ='
               02  FILL-PART-3 PICTURE X(17).
05017          02  FILLER  PICTURE X VALUE ')'.
05018      01  STANINFO-SSA PICTURE X(22)  VALUE
05019                     'STANINFO(STANKEY  =02)'.
06001      01  STATUS-CODE-MSG.
06002          02  FILLER PICTURE X(23) VALUE 'UNRESOLVED STATUS CODE '.
06003          02  FILL-STATUS  PICTURE XX.
06004          02  FILLER PICTURE X(4) VALUE ' ON '.
06005          02  FILL-FUNCTION PICTURE X(4).
06011  LINKAGE SECTION.
07001      01  DATABASE.
07002          02  DBASE-NAME  PICTURE X(8).
07003          02  SEGMENT-INDR PICTURE XX.
07004          02  DBASE-STATUS PICTURE XX.
07005          02  PROC-OPTIONS PICTURE XXXX.
07006          02  DLI-RESERVED PICTURE XXXX.
07007          02  SEG-FEEDBACK PICTURE X(8).
07008          02  KEY-FEEDBACK-LENGTH  PICTURE XXXX.
07009          02  NO-OF-SENSEG-TYPES   PICTURE XXXX.
07010          02  KEY-FEEDBACK.
                   03  ROOT-KEY1.
07012                  04  FILLER  PICTURE XX.
07013                  04  PARTNUM PICTURE X(15).
07014              03  STANINFO-KEY PICTURE XX.
08001  PROCEDURE DIVISION.
08002  START-OUT.
08003      ENTER LINKAGE.
```

```
                 ENTRY 'DLITCBL' USING DATABASE.
08005       ENTER COBOL.
         OPEN-FILES.
              OPEN INPUT MESSAGE-FILE
              OUTPUT TEST-OUTPUT-FILE.
         READ-MESSAGE-FILE.
                READ MESSAGE-FILE INTO MESSAGE-IN-WORK-AREA
               AT END GO TO EXIT-RTN.
               MOVE ROOT-KEY TO FILL-PART-3.
100069      ENTER LINKAGE.
09016        CALL 'CBLTDLI' USING GET-UNIQUE, DATABASE, ROOT-FORMAT,
09017                                   ROOT-SSA.
09018       ENTER COBOL.
             IF DBASE-STATUS = '  ' GO TO PROCESS-FIRST-LINE.
             IF DBASE-STATUS NOT = SEG-NOT-FOUND GO TO DBASE-ERROR.
             MOVE ROOT-KEY TO FILL-PART-1.
10003       MOVE PART-NOT-FOUND-MSG TO OUTPUT-TEXT.
                 PERFORM TERM-OUT-RTN.
             GO TO READ-MESSAGE-FILE.
11007  PROCESS-FIRST-LINE.
             MOVE ROOT-KEY TO FILL-PART-2.
11009       MOVE DESCRIPTION TO FILL-DESCR.
11010       MOVE GET-NEXT TO FILL-FUNCTION.
11011       ENTER LINKAGE.
11012        CALL 'CBLTDLI' USING GET-NEXT, DATABASE, STANINFO-FORMAT,
11013                                   ROOT-SSA, STANINFO-SSA.
11014       ENTER COBOL.
11015       IF DBASE-STATUS = '  ' MOVE PROCUREMENT-CODE TO
11016           FILL-PROC-CODE, GO TO PROCESS-SECOND-LINE.
11017       IF DBASE-STATUS = SEG-NOT-FOUND
11018           PERFORM PROCESS-SECOND-LINE.
11019       GO TO DBASE-ERROR.
12001  PROCESS-SECOND-LINE.
12003       MOVE FIRST-LINE TO OUTPUT-TEXT.
12005       PERFORM TERM-OUT-RTN.
12006  EXIT-FROM-PROCESS-2ND-LINE. EXIT.
120061 PTCH-3.
12007       MOVE INVENTORY-CODE TO FILL-INV-CODE.
12008       MOVE MAKE-DEPT TO FILL-MAKE-1.
12009       MOVE MAKE-COST-CTR TO FILL-MAKE-2.
12010       MOVE PLANNING-REVISION-NUMBER TO FILL-PLAN-REV-NUM.
12011       MOVE MAKE-SPAN TO FILL-MAKE-TIME.
12012       MOVE COMMODITY-CODE TO FILL-COMM-CODE.
12014       MOVE SECOND-LINE TO OUTPUT-TEXT.
12016       GO TO TERM-OUT-RTN.
10005  TERM-OUT-RTN.
             WRITE PRINT-LINE FROM LINE-OUTPUT.
       GO-BACK.
           GO TO READ-MESSAGE-FILE.
08014  EXIT-RTN.
           CLOSE  MESSAGE-FILE.
           CLOSE  TEST-OUTPUT-FILE.
08015       ENTER LINKAGE.
08016        RETURN.
08017       ENTER COBOL.
```

```
100135 COMMON-ERROR.
10015       MOVE STATUS-CODE-MSG TO OUTPUT-TEXT.
100165 EXIT-FROM-ERROR-HANDLER. EXIT.
100166 PTCH-2.
10017       GO TO TERM-OUT-RTN.
10018  DBASE-ERROR.
10019       MOVE DBASE-STATUS TO FILL-STATUS.
10020       GO TO COMMON-ERROR.
/*
//G.DI21PARO   DD   DSN=IMS.DI21PARO,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.DI21PART   DD   DSN=IMS.DI21PART,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.TESTOUT  DD  SYSOUT=3
//G.TESTIN   DD   *,DCB=BLKSIZE=80
02252252-003
023008027
0230080270000000
02JAN1N976B
023007228
023009225
02300928
023009270
023856124
/*
```

BATCH PROCESSING -- PL/I EXAMPLE

```
//PLICAB    JOB   848,CABANISS,MSGLEVEL=1,MSGCLASS=3,CLASS=A,PRTY=8
//JOBLIB   DD   DSN=ICS.LOAD,DISP=SHR
//       DD   DSN=CLA.PGMLIB,DISP=SHR
//       EXEC    IMSPLIGO,MBR=STUDNT03
//C.SYSIN    DD    *
 DLITPLI: PROC(DATABASE) OPTIONS(MAIN);
 DECLARE  GET_UNIQUE     CHARACTER(4) INITIAL('GU  ');
 DECLARE  GET_NEXT       CHARACTER(4) INITIAL('GN  ');
 DECLARE  INSERT         CHARACTER(4) INITIAL('ISRT');
 DECLARE  CARD_IN        CHARACTER(80);
 DECLARE  SEG_NOT_FOUND  CHARACTER(2) INITIAL('GE');
 DECLARE  FOUR FIXED BINARY  INITIAL(4);
 DECLARE  ERROR_SWITCH   CHARACTER(1) INITIAL(' ');
 DECLARE FIVE FIXED BINARY  INITIAL(5);
 DECLARE  1 PART_NOT_FOUND_MSG,
             2 FILL35         CHARACTER(12)  INITIAL('PART NUMBER'),
             2 FILL_PART_1    CHARACTER(17),
             2 FILL36        CHARACTER(16) INITIAL(' NOT ON DATABASE');
 DECLARE  1 LINE_OUTPUT,
             2 OUTPUT_TEXT    CHARACTER(132);
 DECLARE  1 MESSAGE_IN_WORK_AREA,
             2 CARD_INPUT,
                3 ROOT_KEY    CHARACTER(17);
 DECLARE  1 FIRST_LINE,
             2 FILL1          CHARACTER(5)  INITIAL('PART='),
             2 FILL_PART_2    CHARACTER(17),
             2 FILL2          CHARACTER(7)  INITIAL('  DESC='),
             2 FILL_DESCR     CHARACTER(20),
             2 FILL3          CHARACTER(12) INITIAL('  PROC CODE='),
             2 FILL_PROC_CODE CHARACTER(2)  INITIAL('  ');
 DECLARE  1 SECOND_LINE,
             2 FILL4          CHARACTER(9)  INITIAL('INV CODE='),
             2 FILL_INV_CODE  CHARACTER(1)  INITIAL(' '),
             2 FILL5          CHARACTER(12) INITIAL('  MAKE DEPT='),
             2 FILL_MAKE_1    CHARACTER(2)  INITIAL('  '),
             2 FILL6          CHARACTER(1)  INITIAL('-'),
             2 FILL_MAKE_2    CHARACTER(2)  INITIAL('  '),
             2 FILL7          CHARACTER(15) INITIAL('  PLAN REV NUM='),
             2 FILL_PLAN_REV_NUM  CHARACTER(2) INITIAL('  '),
             2 FILL8          CHARACTER(12) INITIAL('  MAKE TIME='),
             2 FILL_MAKE_TIME PICTURE 'ZZZ',
             2 FILL9          CHARACTER(12) INITIAL('  COMM CODE='),
             2 FILL_COMM_CODE CHARACTER(4)  INITIAL('    ');
 DECLARE  1 ROOT_FORMAT,
             2 FILL10         CHARACTER(26),
             2 DESCRIPTION    CHARACTER(20),
             2 FILL11         CHARACTER(4);
 DECLARE  1 STANINFO_FORMAT,
             2 FILL12         CHARACTER(18),
             2 PROCUREMENT_CODE CHARACTER(2) INITIAL('  '),
             2 INVENTORY_CODE CHARACTER(2)   INITIAL(' '),
             2 PLANNING_REVISION_NUMBER CHARACTER(2) INITIAL('  '),
            2 FILL13         CHARACTER(24),
            2 MAKE_DEPT      CHARACTER(2)   INITIAL('  '),
            2 MAKE_COST_CTR  CHARACTER(2)   INITIAL('  '),
            2 FILL14         CHARACTER(2)   INITIAL('  '),
            2 COMMODITY_CODE CHARACTER(4)   INITIAL('    '),
            2 FILL15         CHARACTER(4)   INITIAL('    '),
            2 MAKE_SPAN      PICTURE 'S999',
            2 FILL16         CHARACTER(21);
DECLARE  1 ROOT_SSA,
            2 FILL17 CHARACTER(19) INITIAL('PARTROOT(PARTKEY  ='),
            2 FILL_PART_3    CHARACTER(17),
            2 FILL18         CHARACTER(1)   INITIAL(')');
DECLARE  STANINFO_SSA CHARACTER(22) INITIAL('STANINFO(STANKEY  =02)');
DECLARE  1 STATUS_CODE_MSG,
            2 FILL19 CHARACTER(22) INITIAL('UNRESOLVED STATUS CODE'),
            2 FILL_STATUS    CHARACTER(2)   INITIAL('  '),
            2 FILL20         CHARACTER(4)   INITIAL(' ON '),
            2 FILL_FUNCTION  CHARACTER(4);
DECLARE  1 DATABASE,
            2 DBASE_NAME     CHARACTER(8),
            2 SEGMENT_INDR   CHARACTER(2),
            2 DBASE_STATUS   CHARACTER(2),
            2 PROC_OPTIONS   CHARACTER(4),
            2 DLI_RESERVED   CHARACTER(4),
            2 SEG_FEEDBACK   CHARACTER(4),
            2 KEY_FEEDBACK_LENGTH CHARACTER(4),
            2 NO_OF_SENSEG_TYPES  CHARACTER(4),
            2 KEY_FEEDBACK,
                3 ROOT_KEY1,
                    4 FILL21 CHARACTER(2),
                    4 PARTNUM CHARACTER(15),
                3 STANINFO_KEY CHARACTER(2);
READ_CARDS:   GET FILE (TESTIN) EDIT (CARD_IN) (A(80));
              ON ENDFILE (TESTIN) GO TO EXIT_RTN;
              GET STRING (CARD_IN) EDIT (ROOT_KEY) (A(17));
              FILL_PART_3=ROOT_KEY;
CALL_DB:      CALL PLITDLI(FOUR,GET_UNIQUE,DATABASE,ROOT_FORMAT,
                           ROOT_SSA);
              IF DBASE_STATUS='  ' THEN GO TO PROCESS_FIRST_LINE;
              IF DBASE_STATUS¬=SEG_NOT_FOUND THEN GO TO DBASE_ERROR;
              FILL_PART_1=ROOT_KEY;
            OUTPUT_TEXT=STRING(PART_NOT_FOUND_MSG);
            PUT SKIP FILE (TESTOUT) EDIT (LINE_OUTPUT) (A);
            GO TO READ_CARDS;
PROCESS_FIRST_LINE: FILL_PART_2=ROOT_KEY;
                    FILL_DESCR=DESCRIPTION;
                    FILL_FUNCTION=GET_NEXT;
                    CALL PLITDLI(FIVE,GET_NEXT,DATABASE,
                   STANINFO_FORMAT,ROOT_SSA,STANINFO_SSA);
                    IF DBASE_STATUS='  ' THEN
                   FILL_PROC_CODE=PROCUREMENT_CODE;
                     GO TO PROCESS_SECOND_LINE;
                    IF DBASE_STATUS=SEG_NOT_FOUND THEN
                     GO TO PROCESS_SECOND_LINE;
                    GO TO DBASE_ERROR;
```

```
 PROCESS_SECOND_LINE: OUTPUT_TEXT=STRING(FIRST_LINE);
             PUT SKIP FILE (TESTOUT) EDIT (LINE_OUTPUT) (A);
                      FILL_INV_CODE=INVENTORY_CODE;
                      FILL_MAKE_1=MAKE_DEPT;
                      FILL_MAKE_2=MAKE_COST_CTR;
                    FILL_PLAN_REV_NUM=PLANNING_REVISION_NUMBER;
                      FILL_MAKE_TIME=MAKE_SPAN;
                      FILL_COMM_CODE=COMMODITY_CODE;
                      OUTPUT_TEXT=STRING(SECOND_LINE);
 TERM_OUT_RTN:   PUT SKIP FILE (TESTOUT) EDIT (LINE_OUTPUT) (A);
                    GO TO READ_CARDS;
 EXIT_RTN:     RETURN;
 COMMON_ERROR:       OUTPUT_TEXT=STRING(STATUS_CODE_MSG);
                    GO TO TERM_OUT_RTN;
 DBASE_ERROR:       FILL_STATUS=DBASE_STATUS;
                    GO TO COMMON_ERROR;
 END DLITPLI;
/*
//G.DI21PARO   DD   DSN=IMS.DI21PARO,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.DI21PART   DD   DSN=IMS.DI21PART,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.TESTOUT  DD  SYSOUT=3,DCB=BLKSIZE=132
//G.TESTIN   DD   *,DCB=BLKSIZE=80
02252252-003
023008027
02300802700000000
02JAN1N976B
023007228
023009225
02300928
023009270
023856124
/*
```

BATCH PROCESSING -- ASSEMBLER EXAMPLE

```
//ASGOCAB    JOB   848,CABANISS,MSGLEVEL=1,MSGCLASS=3,CLASS=A,PRTY=8
//JOBLIB   DD    DSN=ICS.LOAD,DISP=SHR
//       DD    DSN=CLA.PGMLIB,DISP=SHR
//           EXEC  IMSASSEM,MBR=STUDNT05
//C.SYSIN    DD    *
DLITCBL  CSECT
         USING *,12                 USE A BASE REGISTER
PROLOGUE DS    0H
         STM   14,12,12(13)         SAVE REGISTERS
         LR    12,15                SET UP BASE REGISTER
         LR    15,13                SAVE ADDRESS OF CALLER'S SAVE AREA
         LA    13,SAVE0001     LOAD REGISTER 13 WITH SAVE AREA ADDRESS
         ST    15,4(13)             FORWARD CHAIN
         ST    13,8(15)             BACK CHAIN
         L     15,16(15)            RESTORE WORKING REGISTER
         B     SAVE0001+72          BRANCH AROUND SAVE AREA
SAVE0001 DS    18F
         L     8,SAVE0001+4         ADDRESS OF PREVIOUS SAVE
         L     8,24(8)              ADDRESS OF FIRST PCB
         MVC   PCBPTR(4),0(8)
         NI    PCBPTR,X'7F'
***OPEN-FILES
         OPEN  (CARD,(INPUT),PRINT,(OUTPUT))
***READ-MESSAGE-FILE
GETCARD  GET   CARD,INPUT           GET INPUT CARD
         MVC   FILLP3,ROOTKEY       MOVE KEY TO SSA
         MVC   FUNCTION,GETUNQ      MOVE FUNCTION CODE
         MVC   IOAREA,AROOTFMT      MOVE IO AREA TO CALL LIST
         MVC   PCOUNT,=F'4'
         LA    1,TPCOUNT
         CALL  CBLTDLI              GO GET SEGMENT
         L     9,PCBPTR             ADDRESS OF DATABASE PCB
         CLC   10(2,9),=X'4040'
         BE    PROFLIN              GO PROCESS FIRST LINE
         CLC   10(2,9),SEGNOTF      IF STATUS CODE NOT FOR SEGMENT
         BNE   DBASEROR              NOT FOUND THEN ERROR
         MVC   OUTTEXT(132),BLANKS
         MVC   FILLP1,ROOTKEY       SET UP PRINT FOR SEGMENT NOT FOUND
         MVC   OUTTEXT(45),PARTNF   MOVE TO OUTPUT AREA
         PUT   PRINT,BLNKSP
         B     GETCARD              GET NEXT INPUT CARD
***PROCESS-FIRST-LINE
PROFLIN  MVC   FILLP2,ROOTKEY
         MVC   FILDESCR,DESCRPT
         MVC   FILLFUN,GETNEXT
         MVC   FUNCTION,GETNEXT
         MVC   PCOUNT,=F'5'         SET PARM COUNT TO 5
         MVC   IOAREA,ASTANINF      SET UP ADDRESS OF IO AREA
         MVC   OUTTEXT(132),BLANKS
         LA    1,TPCOUNT            ADDRESS OF PARAMETER LIST
         CALL  CBLTDLI              GO GET SEGMENT
         L     9,PCBPTR             POINTER TO PCB
         CLC   10(2,9),=X'4040'    COMPARE TO BLANKS
         BNE   JUMP1
         MVC   FILLPCD,PROCCDE
         B     PROSECLN
JUMP1    CLC   10(2,9),=C'GE'
         BE    PROSECLN            SEGMENT NOT FOUND
         B     DBASEROR
***PROCESS-SECOND-LINE
PROSECLN MVC   OUTTEXT(63),FIRSTLN
         PUT   PRINT,BLNKSP        PRINT FIRST LINE
         MVC   FILINVC,MKDPT       SETUP
         MVC   FILLMK1,MKDPT          SECOND
         MVC   FILLMK2,MKCOSTCR           LINE
         MVC   FILPRN,PLANRVNO
         MVC   FILMT,MAKSPN
         MVC   FILCOCD,COMMCODE
         MVC   OUTTEXT(132),BLANKS
         MVC   OUTTEXT(74),SECLINE
         PUT   PRINT,BLNKSP
         B     GETCARD
***EXIT-RTN
EXITRTN  CLOSE (CARD,,PRINT)       EXIT
EPILOGUE DS    0H
         L     13,4(13)            LOAD REGISTER 13 WITH ADDRESS
         LM    14,12,12(13)        RESTORE REGISTERS
         MVI   12(13),X'FF'        INDICATE RETURN
         LA    15,0(0,0)           SET UP RETURN CODE IN REGISTER 15
         BR    14                  RETURN
***COMMON-ERROR
CERROR   MVC   OUTTEXT(33),STATCOMS
         PUT   PRINT,BLNKSP
         B     GETCARD
**DBASE-ERROR
DBASEROR L     9,PCBPTR
         MVC   FILLSTAT,10(9)
         B     CERROR
RETURN   B     EXITRTN
***WORKING STORAGE SECTION
GETUNQ   DC    C'GU  '
GETNEXT  DC    C'GN  '
SEGNOTF  DC    C'GE'
ERSWTCH  DC    C' '
***LINE-OUTPUT
BLNKSP   DC    C' '
OUTTEXT  DC    CL132' '
***PART-NOT-FOUND-MSG
PARTNF   DC    C'PART NUMBER '
FILLP1   DS    CL17
         DC    C' NOT ON DATABASE'
***MESSAGE-IN-WORK-AREA
INPUT    DS    0F
ROOTKEY  DS    CL17
         DS    CL63
***FIRST-LINE
FIRSTLN  DC    C'PART='
```

```
FILLP2   DS    CL17
         DC    C'  DESC='
FILDESCR DS    CL20
         DC    C'  PROC CODE='
FILLPCD  DC    C'  '
***SECOND-LINE
SECLINE  DC    C'INV CODE='
FILINVC  DC    C' '
         DC    C'  MAKE DEPT='
FILLMK1  DC    C'  '
         DC    C'-'
FILLMK2  DC    C'  '
         DC    C'  PLAN REV NUM='
FILPRN   DC    C'  '
         DC    C'  MAKE TIME='
FILMT    DC    C'   '
         DC    C'  COMM CODE='
FILCOCD  DC    C'    '
***ROOT-FORMAT
ROOTFMT  DS    CL26
DESCRPT  DS    CL20
         DC    C'    '
***STANINFO-FORMAT
STANINF  DS    CL18
PROCCDE  DC    C'  '
INVCODE  DC    C' '
PLANRVNO DC    C'  '
         DS    CL24
MKDPT    DC    C'  '
MKCOSTCR DC    C'  '
         DS    CL2
COMMCODE DC    C'    '
         DS    CL4
MAKSPN   DS    CL4
         DS    CL21
***ROOT-SSA
ROOTSSA  DC    C'PARTROOT(PARTKEY  ='
FILLP3   DS    CL17
         DC    C')'
PCOUNT   DS    F
TPCOUNT  DC    A(PCOUNT)
         DC    A(FUNCTION)
PCBPTR   DS    F
IOAREA   DS    F
FLAG1    DC    A(ROOTSSA)
FLAG2    DC    A(STANISSA)
FUNCTION DS    CL4
AROOTFMT DC    A(ROOTFMT)
ASTANINF DC    A(STANINF)
BLANKS   DC    CL132' '
***STANINFO-SSA
STANISSA DC    C'STANINFO(STANKEY  =02)'
***STATUS-CODE-MSG
STATCOMS DC    C'UNRESOLVED STATUS CODE '
FILLSTAT DC    C'  '
```

```
         DC    C' ON '
FILLFUN  DC    C'    '
CARD     DCB   BLKSIZE=80,BUFL=160,DDNAME=TESTIN,DSORG=PS,EODAD=RETURN,X
               LRECL=80,MACRF=GM,RECFM=F
PRINT    DCB   DDNAME=TESTOUT,DSORG=PS,MACRF=PM,LRECL=133,RECFM=FA
***LINKAGE SECTION
DBPCB    DSECT
DBDNAME  DS    CL8
SEGLEVEL DS    CL2
STATCODE DS    CL2
PROCOPTS DS    CL4
RESVDLI  DS    CL4
SEGNAME  DS    CL8
LGTHFDBK DS    CL4
NOSENSEG DS    CL4
KEYFDBK  DS    CL19
         END
/*
//L.SYSLIN DD DSN=&&LIN,DISP=(OLD,DELETE)
//          DD DSN=ICS.PROCLIB(DLITCBL),DISP=SHR
//          DD DDNAME=SYSIN
//      EXEC    DLIBATCH,PSB=STUDNT05
//G.DI21PARO    DD    DSN=IMS.DI21PARO,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.DI21PART    DD    DSN=IMS.DI21PART,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.TESTOUT  DD    SYSOUT=3,DCB=BLKSIZE=133
//G.TESTIN   DD    *,DCB=BLKSIZE=80
02252252-003
023008027
0230080270000000
02JAN1N976B
023007228
023009225
02300928
023009270
023856124
/*
```

FIELD DEFINITIONS FOR DATA BASE SEGMENTS

```
000980 01  SEG-RET-AREA.
001000     02 FILLER                 PICTURE X(02).
001020     02 PART-NO                PICTURE X(15).
001040     02 FILLER                 PICTURE X(09).
001060     02 DESC                   PICTURE X(15).
001080     02 FILLER                 PICTURE X(119).
001100 01  STAN-INFO-RET  REDEFINES SEG-RET-AREA.
001120     02 FILLER                 PICTURE X(18).
001140     02 PROC-CODE              PICTURE XX.
001160 01  STOCK-STATUS-RET  REDEFINES STAN-INFO-RET.
001180     02 FILLER                 PICTURE XX.
001200     02 SS-AREA                PICTURE X.
001220     02 SS-DEPT                PICTURE XX.
001240     02 SS-PROJ                PICTURE XXX.
001260     02 SS-DIV                 PICTURE XX.
001280     02 FILLER                 PICTURE X(10).
001300     02 SS-UNIT-PRICE          PICTURE 9(6)V999.
001320     02 FILLER                 PICTURE X(05).
001340     02 SS-UNIT-OF-MEAS        PICTURE X(04).
001360     02 FILLER                 PICTURE X(33).
001380     02 SS-STOCK-DATE          PICTURE X(03).
001400     02 FILLER                 PICTURE X(15).
001420     02 SS-CUR-REQMTS-1        PICTURE S9(7)V9.
001440     02 SS-UNPL-REQMTS         PICTURE S9(7)V9.
001460     02 SS-ON-ORDER            PICTURE S9(7)V9.
001480     02 SS-IN-STOCK            PICTURE S9(7)V9.
001500     02 SS-PLAN-DISB           PICTURE S9(7)V9.
001520     02 SS-UNPL-DISB           PICTURE S9(7)V9.
001540     02 FILLER                 PICTURE X(23).
001560 01  BACK-ORDER-RET            REDEFINES STOCK-STATUS-RET.
001580     02 FILLER                 PICTURE X(02).
001600     02 WORK-ORDER             PICTURE X(08).
001620     02 FILLER                 PICTURE X(53).
001640     02 WO-QTY                 PICTURE S9(07)V9.
001660 01  CYCLE-COUNT-RET           REDEFINES BACK-ORDER-RET.
001680     02 FILLER                 PICTURE X(02).
001700     02 PHYSICAL-COUNT         PICTURE S9(07)V9.
001720     02 FILLER                 PICTURE X(04).
001740     02 TOTAL-STOCK            PICTURE S9(07)V9.
```

The above COBOL field definitions may be used if one desires to print out segments other than the PARTROOT and STANINFO segments of the sample problem data base.

SAMPLE OUTPUT

THE OUTPUT PRESENTED BELOW IS INDICATIVE OF THE OUTPUT WHICH CAN BE EXPECTED FROM THE PRECEDING COBOL, PL/I, OR ASSEMBLER PROGRAM.

```
PART=02252252-003       DESC=COUPLING                  PROC CODE=74
INV CODE=2  MAKE DEPT=20-0   PLAN REV NUM=    MAKE TIME=300  COMM CODE=6
PART=023008027          DESC=CARD FRONT                PROC CODE=46
INV CODE=A  MAKE DEPT=24-6   PLAN REV NUM=    MAKE TIME=600  COMM CODE=4
PART NUMBER 0230080270000000 NOT ON DATABASE
PART=02JAN1N976B        DESC=DIODE CODE-A              PROC CODE=74
INV CODE=2  MAKE DEPT=20-0   PLAN REV NUM=    MAKE TIME=300  COMM CODE=2
PART=023007228          DESC=HOUSING                   PROC CODE=22
INV CODE=2  MAKE DEPT=20-0   PLAN REV NUM=    MAKE TIME=     COMM CODE=4
PART NUMBER 023009225        NOT ON DATABASE
PART NUMBER 02300928         NOT ON DATABASE
PART=023009270          DESC=HOUSING                   PROC CODE=22
INV CODE=2  MAKE DEPT=20-0   PLAN REV NUM=    MAKE TIME=     COMM CODE=8
PART NUMBER 023856124        NOT ON DATABASE
```

COBOL MESSAGE PROCESSING PROGRAM WITH MESSAGE SIMULATOR

```
//EXEC02  JOB  848,CABANISS,MSGLEVEL=1,MSGCLASS=3,CLASS=A,PRTY=8
//JOBLIB   DD   DSN=CLA.PGMLIB,DISP=SHR
//        DD   DSN=ICS.LOAD,DISP=SHR
//          EXEC   IMSCOBOL,MBR=CAB01
//C.SYSIN       DD   *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. 'CAB'.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  A PICTURE X(24).
       LINKAGE SECTION.
       01  DB-PCB.
             02  DATA-BAS-DESC   PICTURE X(71).
       PROCEDURE DIVISION.
             ENTER LINKAGE.
                ENTRY 'DLITCBL' USING DB-PCB.
             ENTER COBOL.
             ENTER LINKAGE.
                CALL 'TEST' USING A, DB-PCB.
             ENTER COBOL.
             STOP RUN.
/*
//          EXEC   IMSCOBOL,MBR=CAB02
//C.SYSIN    DD    *
01001  IDENTIFICATION DIVISION.
01002  PROGRAM-ID.  'CLASSEX'
01004     REMARKS.  IMS DEMONSTRATION OF TERMINAL ABILITY
01005               TO DISPLAY A PART NUMBER SEGMENT AND
01006               SOME OF ITS STANDARD INFORMATION WHICH
01007               ARE ON AN IMS ISAM/OSAM INVENTORY DATABASE.
01008  ENVIRONMENT DIVISION.
01009  CONFIGURATION SECTION.
01010  SOURCE-COMPUTER. IBM-360.
01011  OBJECT-COMPUTER. IBM-360.
01012  DATA DIVISION.
01013  WORKING-STORAGE SECTION.
01014      77  GET-UNIQUE  PICTURE XXXX VALUE 'GU  '.
01015      77  GET-NEXT    PICTURE XXXX VALUE 'GN  '.
01016      77  INSERT      PICTURE XXXX VALUE 'ISRT'.
01017      77  SEG-NOT-FOUND PICTURE XX VALUE 'GE'.
01019      77  NO-MSG-THERE  PICTURE XX  VALUE 'QC'.
01020      77  ERROR-SWITCH  PICTURE X   VALUE ' '.
02012      01  LINE-INPUT.
02013          02  INPUT-COUNT PICTURE S99 COMPUTATIONAL.
02014          02  FILLER      PICTURE S99 COMPUTATIONAL.
               02  TRAN-SACT   PICTURE X(8).
               02  FILLER      PICTURE X.
               02  ROOT-KEY    PICTURE X(17).
               02  INPUT-TEXT  PICTURE X(106)  VALUE SPACES.
02016      01  LINE-OUTPUT.
02017          02  OUTPUT-COUNT PICTURE S99 COMPUTATIONAL.
02018          02  FILLER       PICTURE S99 COMPUTATIONAL VALUE ZERO
020185         02  CARR-RETURN PICTURE X VALUE 'N'.
02019          02  OUTPUT-TEXT  PICTURE X(132) VALUE SPACES.
```

```
03001     01  PART-NOT-FOUND-MSG.
03002         02  FILLER PICTURE X(12) VALUE 'PART NUMBER '.
              02  FILL-PART-1 PICTURE X(17).
03004         02  FILLER PICTURE X(16) VALUE ' NOT ON DATABASE'.
03005         02  CR-1   PICTURE X.
03006     01  FIRST-LINE.
03007         02  FILLER PICTURE X(5) VALUE 'PART='.
              02  FILL-PART-2 PICTURE X(17).
03009         02  FILLER PICTURE X(7) VALUE '  DESC='.
03010         02  FILL-DESCR PICTURE X(20).
03011         02  FILLER PICTURE X(12) VALUE '  PROC CODE='.
03012         02  FILL-PROC-CODE PICTURE XX VALUE SPACES.
03013         02  CR-2  PICTURE X.
03014     01  SECOND-LINE.
03015         02  FILLER  PICTURE X(9) VALUE 'INV CODE='.
03016         02  FILL-INV-CODE PICTURE X VALUE SPACE.
03017         02  FILLER  PICTURE X(12) VALUE '  MAKE DEPT='.
03018         02  FILL-MAKE-1 PICTURE XX VALUE SPACES.
03019         02  FILLER  PICTURE X  VALUE '-'.
03020         02  FILL-MAKE-2 PICTURE XX VALUE SPACES.
04001         02  FILLER  PICTURE X(15) VALUE '  PLAN REV NUM='.
04002         02  FILL-PLAN-REV-NUM PICTURE XX VALUE SPACES.
04003         02  FILLER  PICTURE X(12) VALUE '  MAKE TIME='.
04004         02  FILL-MAKE-TIME  PICTURE ZZZ.
04005         02  FILLER  PICTURE X(12) VALUE '  COMM CODE='.
04006         02  FILL-COMM-CODE PICTURE XXXX VALUE SPACES.
04007         02  CR-3  PICTURE X.
04008     01  NO-STANINFO-MSG.
04009         02  FILLER PICTURE X(29)
04010                 VALUE 'THERE IS NO STANINFO SEGMENT.'.
04011         02  CR-4  PICTURE X.
04012     01  ROOT-FORMAT.
04013         02  FILLER  PICTURE X(26).
04014         02  DESCRIPTION PICTURE X(20).
04015         02  FILLER  PICTURE XXXX.
04020     01  STANINFO-FORMAT-DEFINITION PICTURE X(85) VALUE SPACES.
05001     01  STANINFO-FORMAT REDEFINES STANINFO-FORMAT-DEFINITION.
05002         02  FILLER  PICTURE X(18).
05003         02  PROCUREMENT-CODE  PICTURE XX.
05004         02  INVENTORY-CODE  PICTURE X.
05005         02  PLANNING-REVISION-NUMBER  PICTURE XX.
05006         02  FILLER  PICTURE X(24).
05007         02  MAKE-DEPT  PICTURE XX.
05008         02  MAKE-COST-CTR  PICTURE XX.
05009         02  FILLER  PICTURE XX.
05010         02  COMMODITY-CODE  PICTURE XXXX.
05011         02  FILLER  PICTURE XXXX.
05012         02  MAKE-SPAN  PICTURE S999.
05013         02  FILLER  PICTURE X(21).
05014     01  ROOT-SSA.
              02  FILLER  PICTURE X(19) VALUE 'PARTROOT(PARTKEY  ='.
              02  FILL-PART-3  PICTURE X(17).
05017         02  FILLER  PICTURE X VALUE ')'.
05018     01  STANINFO-SSA PICTURE X(22)  VALUE
05019                   'STANINFO(STANKEY  =02)'.
```

```
06001      01  STATUS-CODE-MSG.
06002          02  FILLER PICTURE X(23) VALUE 'UNRESOLVED STATUS COD '
06003          02  FILL-STATUS  PICTURE XX.
06004          02  FILLER PICTURE X(4) VALUE ' ON '.
06005          02  FILL-FUNCTION PICTURE X(4).
06006          02  CR-5  PICTURE X.
06011  LINKAGE SECTION.
06012      01  TERMINAL.
06013          02  LTERM-NAME   PICTURE X(8).
06014          02  FILLER       PICTURE XX.
06015          02  TERM-STATUS PICTURE XX.
06016          02  TERM-PREFIX.
06017              03  FILLER  PICTURE X.
06018              03  JULIAN-DATE PICTURE S9(5) COMPUTATIONAL-3.
06019              03  TIME-OF-DAY PICTURE S9(7) COMPUTATIONAL-3.
06020              03  FILLER  PICTURE XXXX.
07001      01  DATABASE.
07002          02  DBASE-NAME  PICTURE X(8).
07003          02  SEGMENT-INDR PICTURE XX.
07004          02  DBASE-STATUS PICTURE XX.
07005          02  PROC-OPTIONS PICTURE XXXX.
07006          02  DLI-RESERVED PICTURE XXXX.
07007          02  SEG-FEEDBACK PICTURE X(8).
07008          02  KEY-FEEDBACK-LENGTH  PICTURE XXXX.
07009          02  NO-OF-SENSEG-TYPES   PICTURE XXXX.
07010          02  KEY-FEEDBACK.
                   03  ROOT-KEY1.
07012                  04  FILLER  PICTURE XX.
07013                  04  PARTNUM PICTURE X(15).
07014              03  STANINFO-KEY PICTURE XX.
08001  PROCEDURE DIVISION.
08002  START-OUT.
08003      ENTER LINKAGE.
08004       ENTRY 'TEST' USING TERMINAL, DATABASE.
08005      ENTER COBOL.
       READ-MESG-FILE.
08009      ENTER LINKAGE.
08010       CALL 'GEORGEI' USING GET-UNIQUE, TERMINAL, LINE-INPUT.
08011      ENTER COBOL.
08012      IF TERM-STATUS = '  ' GO TO TERM-OK.
08013      IF TERM-STATUS NOT = NO-MSG-THERE GO TO ERROR-HANDLER.
08014  EXIT-RTN.
08015      ENTER LINKAGE.
08016       RETURN.
08017      ENTER COBOL.
08018  TERM-OK.
           MOVE  ROOT-KEY TO FILL-PART-3.
09015      ENTER LINKAGE.
09016       CALL 'CBLTDLI' USING GET-UNIQUE, DATABASE, ROOT-FORMAT,
09017                                 ROOT-SSA.
09018      ENTER COBOL.
09019      IF  DBASE-STATUS = '  ' GO TO PROCESS-FIRST-LINE.
09020      IF  DBASE-STATUS NOT = SEG-NOT-FOUND GO TO DBASE-ERROR.
           MOVE ROOT-KEY TO FILL-PART-1.
10002      MOVE CARR-RETURN TO CR-1.
```

```
10003      MOVE PART-NOT-FOUND-MSG TO OUTPUT-TEXT.
10004      MOVE 50 TO OUTPUT-COUNT.
10005  TERM-OUT-RTN.
10006      MOVE INSERT TO FILL-FUNCTION.
100069     ENTER LINKAGE.
10007      CALL 'GEORGEI' USING INSERT, TERMINAL, LINE-OUTPUT.
10008      ENTER COBOL.
100085 EXIT-FROM-TERM-OUT. EXIT.
100086 PTCH-1.
10009      GO TO READ-MESG-FILE.
10010  ERROR-HANDLER.  MOVE TERM-STATUS TO FILL-STATUS.
10011      IF ERROR-SWITCH NOT = ' ' DISPLAY STATUS-CODE-MSG UPON
10012           CONSOLE, GO TO EXIT-RTN.
10013      MOVE 'E' TO ERROR-SWITCH.
100135 COMMON-ERROR.
10014      MOVE CARR-RETURN TO CR-5.
10015      MOVE STATUS-CODE-MSG TO OUTPUT-TEXT.
10016      MOVE 38 TO OUTPUT-COUNT.
100165 EXIT-FROM-ERROR-HANDLER. EXIT.
100166 PTCH-2.
10017      GO TO TERM-OUT-RTN.
10018  DBASE-ERROR.
10019      MOVE DBASE-STATUS TO FILL-STATUS.
10020      GO TO COMMON-ERROR.
11007  PROCESS-FIRST-LINE.
           MOVE ROOT-KEY TO FILL-PART-2.
11009      MOVE DESCRIPTION TO FILL-DESCR.
11010      MOVE GET-NEXT TO FILL-FUNCTION.
11011      ENTER LINKAGE.
11012       CALL 'CBLTDLI' USING GET-NEXT, DATABASE, STANINFO-FORMAT,
11013                                ROOT-SSA, STANINFO-SSA.
11014      ENTER COBOL.
11015      IF DBASE-STATUS = '  ' MOVE PROCUREMENT-CODE TO
11016          FILL-PROC-CODE, GO TO PROCESS-SECOND-LINE.
11017      IF DBASE-STATUS = SEG-NOT-FOUND
11018          PERFORM PROCESS-SECOND-LINE, GO TO EXIT-RTN.
11019      GO TO DBASE-ERROR.
12001  PROCESS-SECOND-LINE.
12002      MOVE CARR-RETURN TO CR-2.
12003      MOVE FIRST-LINE TO OUTPUT-TEXT.
12004      MOVE 68 TO OUTPUT-COUNT.
12005      PERFORM TERM-OUT-RTN.
120055     IF TERM-STATUS NOT = '  ' PERFORM ERROR-HANDLER THRU
120056         COMMON-ERROR, PERFORM TERM-OUT-RTN.
120057     IF TERM-STATUS NOT = '  ' GO TO ERROR-HANDLER.
12006  EXIT-FROM-PROCESS-2ND-LINE. EXIT.
120061 PTCH-3.
12007      MOVE INVENTORY-CODE TO FILL-INV-CODE.
12008      MOVE MAKE-DEPT TO FILL-MAKE-1.
12009      MOVE MAKE-COST-CTR TO FILL-MAKE-2.
12010      MOVE PLANNING-REVISION-NUMBER TO FILL-PLAN-REV-NUM.
12011      MOVE MAKE-SPAN TO FILL-MAKE-TIME.
12012      MOVE COMMODITY-CODE TO FILL-COMM-CODE.
12013      MOVE CARR-RETURN TO CR-3.
12014      MOVE SECOND-LINE TO OUTPUT-TEXT.
```

```
12015      MOVE 80 TO OUTPUT-COUNT.
12016      GO TO TERM-OUT-RTN.
/*
//        EXEC    IMSCOBOL,MBR=CAB03
//C.SYSIN      DD   *
001010 IDENTIFICATION DIVISION.
001020 PROGRAM-ID. 'IMSTEST'.
001030 ENVIRONMENT DIVISION.
001040 INPUT-OUTPUT SECTION.
001050 FILE-CONTROL.
001060     SELECT MESSAGE-FILE  ASSIGN TO 'TESTIN'  UTILITY.
001070     SELECT TEST-OUTPUT-FILE  ASSIGN  TO 'TESTOUT' UTILITY.
001080 DATA DIVISION.
001090 FILE SECTION.
001100 FD  MESSAGE-FILE
001110     RECORDING MODE IS F
001120     DATA RECORD IS INPUT-MESSAGE.
001130     01  INPUT-MESSAGE            PICTURE IS X(80).
001140 FD  TEST-OUTPUT-FILE
001150     BLOCK CONTAINS 10 RECORDS
001160     DATA RECORD IS PRINT-LINE.
001170     01  PRINT-LINE               PICTURE IS X(133).
001180 WORKING-STORAGE  SECTION.
001190     77  OPEN-SWITCH  PICTURE X   VALUE ' '.
001200     77  END-SWITCH   PICTURE X   VALUE ' '.
001210     77  MESSAGE-SIZE-WORK  PICTURE S9(4) VALUE 0
001220                        USAGE COMPUTATIONAL.
001230     77  BAD-FUNCTION-CODE  PICTURE XX      VALUE 'QA'.
001240     77  NO-DATA-CODE       PICTURE XX      VALUE 'QC'.
001250     77  REC-SWT    PICTURE X  VALUE ' '.
001260     77  MESS-OUT  PICTURE X  VALUE ' '.
001261     77  C-329     PICTURE  S9(6)     VALUE  329
001262                   USAGE COMPUTATIONAL.
001270     01  MESSAGE-IN-WORK-AREA.
001280         02   HEADER-DATA-IN.
001290              03   MESSAGE-COUNT            PICTURE  9(4).
001300              03   MESSAGE-TYPE             PICTURE  X.
001310              03   TERMINAL-NAME            PICTURE  X(8).
001320         02   MESSAGE-TEXT.
001330              03  FILLER PICTURE X OCCURS 67 TIMES.
001350     01  TEST-OUTPUT-HEADER.
001360         02  FILLER PICTURE X(18) VALUE
001370              '   MESSAGE TYPE = '.
001380         02 FILLER.
001390            03 IN-OR-OUT-MESSAGE            PICTURE X.
001400            03 HEAD-OR-BODY                 PICTURE X.
001410         02 FILLER  PICTURE X(18)           VALUE
001420              ', MESSAGE COUNT = '.
001430         02 OUTPUT-COUNT                    PICTURE  9999.
001440         02 FILLER  PICTURE X(13)           VALUE
001450              ', TERMINAL = '.
001460         02 OUTPUT-TERMINAL                 PICTURE  X(8).
001470         02   FILLER  PICTURE  XX  VALUE SPACES.
001480         02   OUT-FUN  PICTURE  XXXX.
001490     01  TEST-OUTPUT-TEXT.
```

```
001500         02 TEST-OUTPUT-CHAR  OCCURS 130 TIMES
001510                                            PICTURE X.
001520 LINKAGE SECTION.
001530     01  INOUT-PCB.
001540         02   IO-TERMINAL         PICTURE X(8).
001550         02   IO-RESERVE          PICTURE XX.
001560         02   IO-STATUS           PICTURE XX.
001570         02  I-PREFIX   PICTURE X(12).
001580     01  FUNCTION   PICTURE  XXXX.
001590     01  IO-AREAS-RECORD.
001600         02  RCC PICTURE S9(4) USAGE COMPUTATIONAL.
001610          02  RCC-ZEROS   PICTURE  XX.
001620         02  TEXT.
001630              03  FILLER PICTURE X OCCURS 130 TIMES.
001650 PROCEDURE DIVISION.
001660     ENTER LINKAGE.
001670     ENTRY 'GEORGEI' USING FUNCTION, INOUT-PCB, IO-AREAS-RECORD.
001680     ENTER COBOL.
001690 OPEN-FILES.
001700     IF OPEN-SWITCH = '1' GO TO PROCESS-X.
001705         MOVE 0 TO TALLY.
001710     OPEN INPUT MESSAGE-FILE
001720          OUTPUT TEST-OUTPUT-FILE.
001730     MOVE '1' TO OPEN-SWITCH.
001740 PROCESS-X.
001750     IF FUNCTION = 'GU  ' GO TO GET-HEADER.
001760     IF FUNCTION = 'GN  ' GO TO GET-BODY.
001770     IF FUNCTION = 'ISRT' GO TO WRITE-REPLY.
001780     MOVE BAD-FUNCTION-CODE  TO IO-STATUS.
001790 RETURN-TO-APPLICATION.
001806     ENTER LINKAGE.
001810     RETURN.
001820     ENTER COBOL.
001830 FORMAT-INPUT-MESSAGE.
001840     MOVE 'I' TO IN-OR-OUT-MESSAGE.
001850     MOVE MESSAGE-TYPE TO HEAD-OR-BODY.
001860     MOVE MESSAGE-COUNT TO OUTPUT-COUNT.
001870     MOVE TERMINAL-NAME TO OUTPUT-TERMINAL.
001880     MOVE MESSAGE-TEXT  TO TEST-OUTPUT-TEXT.
001890 SET-UP-FOR-USER.
001900     MOVE MESSAGE-COUNT TO RCC.
001910     MOVE LOW-VALUES  TO RCC-ZEROS.
001920     MOVE TERMINAL-NAME TO IO-TERMINAL.
001930     MOVE MESSAGE-TEXT TO     TEXT.
001940      MOVE '  ' TO IO-STATUS.
001950 READ-MESSAGE-FILE.
001960     IF  END-SWITCH = '1' GO TO FINISH-UP.
001990     READ MESSAGE-FILE INTO MESSAGE-IN-WORK-AREA
002000                   AT END MOVE '1' TO END-SWITCH
002010                   GO TO READ-MESSAGE-FILE.
002020     COMPUTE  MESSAGE-SIZE-WORK = MESSAGE-COUNT - 4.
002040     PERFORM  FORMAT-INPUT-MESSAGE.
002050     PERFORM  WRITE-TEST-OUTPUT-FILE.
002060 WRITE-TEST-OUTPUT-FILE.
002070         MOVE  FUNCTION  TO  OUT-FUN.
```

```
002080     WRITE PRINT-LINE  FROM  TEST-OUTPUT-HEADER.
002090     WRITE PRINT-LINE  FROM  TEST-OUTPUT-TEXT.
002100 GET-HEADER.
002110     IF  REC-SWT NOT = 'H'
002130              PERFORM  READ-MESSAGE-FILE
002150              GO TO REC-GOT.
002170     COMPUTE  MESSAGE-SIZE-WORK = MESSAGE-COUNT - 4.
002180     PERFORM  FORMAT-INPUT-MESSAGE.
002190     PERFORM  WRITE-TEST-OUTPUT-FILE.
002200 REC-GOT.
002210     IF MESSAGE-TYPE NOT = TO 'H' GO TO GET-HEADER.
002220     PERFORM  SET-UP-FOR-USER. MOVE ' ' TO REC-SWT.
002230     GO  TO RETURN-TO-APPLICATION.
002240 GET-BODY.
002250     PERFORM  READ-MESSAGE-FILE.
002260      IF MESSAGE-TYPE = 'B' NEXT SENTENCE ELSE
002270              MOVE 'H' TO  REC-SWT
002280              MOVE   'QD'        TO IO-STATUS
002290              GO TO RETURN-TO-APPLICATION.
002300     PERFORM  SET-UP-FOR-USER.
002310     GO TO RETURN-TO-APPLICATION.
002320 WRITE-REPLY.
002330     MOVE IO-TERMINAL TO OUTPUT-TERMINAL.
002340         COMPUTE   MESSAGE-SIZE-WORK  =  RCC - 4.
002350     MOVE  RCC TO  OUTPUT-COUNT.
002360     MOVE  'O' TO  IN-OR-OUT-MESSAGE.
002370     MOVE ' ' TO HEAD-OR-BODY.
002380     MOVE  TEXT TO   TEST-OUTPUT-TEXT.
002390     MOVE  MESS-OUT  TO  IO-STATUS.
002400     PERFORM  WRITE-TEST-OUTPUT-FILE.
002410 FINISH-UP.
002420     IF FUNCTION = 'GU  '  MOVE 'QC' TO IO-STATUS.
002430     IF FUNCTION = 'GN  '  MOVE 'QD' TO IO-STATUS.
002440     GO TO RETURN-TO-APPLICATION.
/*
//        EXEC       IMSLINK
//L.SYSLMOD    DD   DSN=CLA.PGMLIB,DISP=SHR
//L.SYSOBJ   DD   DSN=CLA.PGMLIB,DISP=SHR
//        DD   DSN=ICS.LOAD,DISP=SHR
//L.SYSIN   DD   *
        INCLUDE   SYSOBJ(CAB01)
        INCLUDE   SYSOBJ(CAB02)
        INCLUDE   SYSOBJ(CAB03)
        ENTRY   DLITCBL
        NAME      STUDNT02(R)
/*
//        EXEC  DLIBATCH,PSB=STUDNT02
//         DD   DSN=ICS.DBDLIB,DISP=SHR
//G.IMS    DD    DSN=CLA.PSBLIB,DISP=SHR
//G.DI21PARO   DD   DSN=IMS.DI21PARO,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.DI21PART   DD   DSN=IMS.DI21PART,DISP=SHR,VOL=SER=IMSDBS,UNIT=2314
//G.TESTOUT  DD  SYSOUT=3
//G.TESTIN   DD   *,DCB=BLKSIZE=80
0030HTERM0001PART      023008027
0030HTERM0002PART      023009228
```

```
0030HTERM0003PART      0268663-104
0030HTERM0004PART      02652540-002
0030HTERM0008PART      02989036-001
0030HTERM0005PART      02975105-001
0030HTERM0023PART      02974810-010
0030HTERM0099PART      02968534-001
0030HTERM1258PART      02958007-180
/*
```

SUGGESTED SOLUTIONS FOR EXERCISES

PSB GENERATION

```
//PSBGEN01    JOB     01,STUDENT01,MSGLEVEL=1
//            EXEC    PSBGEN,MBR=STUDNT01
//C.SYSIN     DD       *
          PCB    TYPE=DB,DBDNAME=DI21PART,PROCOPT=A,KEYLEN=43
          SENSEG   PARTROOT
          SENSEG  STANINFO,PARTROOT
          SENSEG   STOKSTAT,PARTROOT
          SENSEG  CYCCOUNT,STOKSTAT
          SENSEG  BACKORDR,STOKSTAT
          PSBGEN   LANG=COBOL,PSBNAME=STUDNT01
          END
/*
```

DBD GENERATION

```
//DBDGEN01    JOB     01,STUDENT01,MSGLEVEL=1
//            EXEC    DBDGEN,MBR=DBASE01
//C.SYSIN     DD       *
          DBD     NAME=DBASE01,ACCESS=ISAM
          DMAN   DD1=PARTB,DEV1=2311,DLIOF=BASE1
          SEGM   NAME=PARTROOT,PARENT=0,BYTES=50,FREQ=250
          FLDK   NAME=PARTKEY,TYPE=C,BYTES=17,START=1
          SEGM   NAME=STANINFO,PARENT=PARTROOT,BYTES=85,FREQ=1
          FLDK   NAME=STANKEY,TYPE=C,BYTES=2,START=1
          SEGM   NAME=STOKSTAT,PARENT=PARTROOT,BYTES=160,FREQ=2
          FLDK   NAME=STOCKEY,TYPE=C,BYTES=16,START=1
          SEGM   NAME=CYCCOUNT,PARENT=STOKSTAT,BYTES=25,FREQ=1
          FLDK   NAME=CYCLKEY,TYPE=C,BYTES=2,START=1
          SEGM   NAME=BACKORDR,PARENT=STOKSTAT,BYTES=75,FREQ=0
          FLDK   NAME=BACKKEY,TYPE=C,BYTES=10,START=1
          DBDGEN
          FINISH
          END
/*
```

SUGGESTED SOLUTION FOR SYSTEM DEFINITION

```
//IMSDEF01 JOB    848,CABANISS,MSGLEVEL=1,MSGCLASS=3,CLASS=A,PRTY=8
//         EXEC   ASMFC,REGION=96K
//ASM.SYSLIB   DD   DSN=ICS.MACLIB,DISP=SHR
//ASM.SYSPRINT  DD   SYSOUT=3
//ASM.SYSPUNCH  DD   SYSOUT=3
//ASM.SYSIN DD *
    IMSCTRL         SYSTEM=(MVT,ALL),MAXIO=10,MAXREGN=3,MSGBUFF=10,     X
               COMMSVC=(244,245),OSAMSVC=243,OCENDA=Z8,CKPT=500
  APPLCTN           PSB=STUDNT01
    DATABASE        DBD=DI21PART
      TRANSACT      CODE=TRANS01,PRTY=(2,6,3),PROCLIM=(5,10)
  APPLCTN           PSB=STUDNT02
    DATABASE        DBD=DI21PART
      TRANSACT      CODE=TRANS02,PRTY=(2,6,3),PROCLIM=(5,10)
  APPLCTN           PSB=STUDNT03
    DATABASE        DBD=DI21PART
      TRANSACT      CODE=TRANS03,PRTY=(2,6,3),PROCLIM=(5,10)
  APPLCTN           PSB=STUDNT04
    DATABASE        DBD=DI21PART
      TRANSACT      CODE=TRANS04,PRTY=(2,6,3),PROCLIM=(5,10)
  APPLCTN           PSB=STUDNT05
    DATABASE        DBD=DI21PART
      TRANSACT      CODE=TRANS05,PRTY=(2,6,3),PROCLIM=(5,10)
  LINEGRP      DDNAME=DD2740S
    LINE          FEAT=POLL,ADDR=032
       TERMINAL   ADDR=E2
          NAME    L2740S2
    LINE          FEAT=POLL,ADDR=033
DFSCTBMT TERMINAL ADDR=E2
          NAME    MASTER
          NAME    L2740S1
    MASTTERM        MASTER
    LINE          FEAT=POLL,ADDR=034
       TERMINAL   ADDR=E2
          NAME    L2740SM1
   LINEGRP        DDNAME=DD2740A,FEAT=(TRANSCTL,SWITCHED),UNITYPE=2740
    LINE          FEAT=AUTOANS,ADDR=035
       TERMINAL   ADDR=E2
          NAME    INQUIRY1
    POOL            FEAT=AUTOCALL,ZONE=1
       SUBPOOL      TELNO=2774211
          NAME    LTERM1
          NAME    LTERM2
          NAME    LTERM3
  PROCLIB      PDS=CLA.PROCLIB
  PGMLIB       PDS=CLA.PGMLIB
  PSBLIB       PDS=CLA.PSBLIB
  DBDLIB       PDS=CLA.DBDLIB
  MACLIB       PDS=ICS.MACLIB,UNIT=2314,VOLNO=IMSLIB
  RESLIB       PDS=CLA.LOAD,UNIT=2311,VOLNO=CLASS1
  MSGQUEUE     REUSE=(YES,150),QCRIN=(INQCR,CLA.INQCR,2311,CLASS),      ,
               QCROUT=(OUTQCR,CLA.OUTQCR,2311,CLASS),                  ,
               MSGOUT=(OUTMSG,CLA.OUTMSG,2311,CLASS),
               MSGIN=(INMSG,CLA.INMSG,2311,CLASS)
         SPACE 2
  IMSGEN       UT1SDS=TEMPSET,ASMPRT=ON,LEPRT=(XREF,LIST)
         END
/*
```

PART NUMBERS IN DATA BASE

02AN960C10
02CK05CW181K
02CSR13G104KL
02JANIN976B
02MS16995-28
02N51P3003
02RC07GF273J
02106B1293P
02250236-001
02250239
02250241-001
0225079
02250796
02250891
02252252-003
023003802
023003806
023007228
023008027
023008838
023009228
023009270
023009280
023013405-002
023013412
023012419-001
023012460-001
023013548-002
0256134-016
0260003-118
02652540-002
02652799
028663-102
0268663-104
0269857-635
027060654P001
027438995P002
027454949P001
027618032P101
027618289P049
027630843P513
027736847P001
02803008035
0282125-056
0282124-640
0282125-869
0284353-456
0290-3033334
0290-3033665
02905537-384
02906028-040
02907021-782
02921125-009
02922294-002
02922399-001
02925363-136
029 25380-101
02930331-102
02930331-123
02930333-001
02945325-086
02950060-006
02954017-001
02958007-180
02960528-067
02968534-001
02974810-010
02975105-001

International Business Machine Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM World Trade Corporation
821 United Nations Plaza, New York, N.Y. 10017
(International)

R20-4142-0

Printed in USA R20-4142-0

International Business Machine Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM World Trade Corporation
821 United Nations Plaza, New York, N.Y. 10017
(International)

Printed in USA R20-4142-0